Decision-Making in a Robotic Architecture for Autonomy

Tara Estlin, Rich Volpe, Issa Nesnas, Darren Mutz, Forest Fisher,
Barbara Engelhardt, and Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
{*firstname.lastname*}@jpl.nasa.gov

Keywords: autonomy, robotics, planning, scheduling, execution

Abstract

This paper presents an overview of the intelligent decision-making capabilities of the CLARAty robotic architecture for autonomy. CLARAty is a two layered architecture where the top Decision Layer contains techniques for autonomously creating a plan of robot commands and the bottom Functional Layer provides standard robot capabilities that interface to system hardware. This paper focuses on the Decision Layer organization and capabilities. Specifically, the Decision Layer provides a framework for utilizing AI planning and executive techniques, which provide onboard, autonomous command generation and re-planning for planetary rovers. The Decision Layer also provides a flexible interface to the Functional Layer, which can be tailored based on user preferences and domain features. This architecture is currently being tested on several JPL rovers.

1 Introduction

NASA recently outlined a new Mars Exploration Program that will have us visit the red planet over six times in the next two decades. At least four of these missions will involve rovers or other robotic craft that will be used to explore the surface of the planet and perform numerous geological and atmospheric experiments. In order to collect a high volume of science data, rovers will require capabilities for long-range traversal and autonomous operation. A key aspect of these capabilities is the generation of rover command sequences. These sequences specify an ordered list of commands that achieve desired science goals while ensuring no rover operation or resource constraints are violated. Sequences must often be changed or enhanced during execution in response to changing science goals or unexpected environment conditions. The model of rover operations used for the Mars-Pathfinder rover, Sojourner, (and the model planned for the Mars '03 twin rovers), is to manually generate sequences on the ground and when necessary, perform additional sequence modifications on the ground based on uploaded data [11]. If something unexpected happens during sequence execution, such as an out-of-range sensor reading or a longer than expected traversal, the rover must be "safed" until further communication from the ground can provide a new command sequence. This procedure often causes hours of lost science time and makes it extremely difficult to take advantage of unexpected science opportunities.

To enable autonomous sequencing onboard a rover, AI researchers have been developing several key pieces of software that interact to provide a valid and desirable set of rover commands. Planning and scheduling systems [3,4,10] input a set of science goals, the current rover state, and a model of rover operations to produce a validated sequence of rover activities. This sequence should achieve as many science goals as possible, while still obeying resource and other operation constraints. Executive systems [9, 15] further expand this activity sequence based on current sensor information into a detailed set of commands and dispatch these commands to low-level rover-hardware controllers for execution at the appropriate time. Planning and scheduling systems typically focus on goal-driven behavior, which enables a robotic system to produce a plan of actions based on a set of high-level goals. Executive systems typically focus on event-driven behavior, which enables a robotic system to quickly react to changes in its environment and modify its actions accordingly.

This paper discusses part of a new robotic architecture called CLARAty (Coupled Layered Architecture for Robotic Autonomy) [17], which is being developed to support autonomous rover operations at the Jet Propulsion Laboratory. In particular, we discuss the top layer of the CLARAty architecture and how it enables the integration of planning and execution functionality, as well as how it provides a flexible interface to the spectrum of functionality built-in to the rover control software, or "Functional Layer" [13]. This top layer is termed the "Decision Layer" since its main objective is to decide what sequence of rover actions should be used to

achieve an input set of goals. Another important objective of the Decision Layer is to provide a framework for using different types of planning and executive systems, and to enable new ways of combining these capabilities.

The rest of this paper is organized in the following manner. First, we introduce the CLARAty architecture and present a brief overview of its two layers. Second, we discuss the first instantiation of the CLARAty Decision Layer, and describe the particular AI planning and executive systems that are being utilized. Next, we describe the important ways the Decision Layer interfaces to the Functional Layer. Finally, we present related work and our conclusions.

2 The CLARAty Architecture

Most mobile robot efforts at the Jet Propulsion Laboratory (JPL) have concentrated on building software infrastructure for navigation, manipulation and control. High-level decision making for these efforts was typically done using very simple execution of linear sequences that were tediously created by ground controllers. However, new missions are looking at rovers that will require significantly more onboard autonomous capabilities to support goals such as long-range traversals, complex science experiments, and longer mission duration. The CLARAty robotic architecture for autonomy is being developed in response to the need for a robotic control architecture that can support future mission autonomy requirements at JPL. CLARAty builds on current work at JPL and in the research fields of robotics and artificial intelligence.

2.1 Review of Three-Level Architecture

Typical robot and autonomy architectures are comprised of three levels – Planning, Executive, and Functional (or Control). These levels are usually organized by the level of abstraction in which they operate. The top Planning level constructs high-level plans utilizing AI planning search techniques. In the past, these algorithms have typically been computationally intensive and required a significant amount of time to respond to new updates or changes. Domain knowledge for this level is encoded in a declarative model, where it can easily be utilized by different search techniques.

The Executive level is responsible for execution of plans produced by the planning level. The executive level typically performs further expansion of planned activities based on current execution context. This level is also responsible for monitoring activities and rover conditions as execution proceeds and for handling exceptions as they arise. This level must quickly react to changes, so its behavior is usually more responsive than that of the planning level. Domain knowledge at the Executive level is typically represented using procedural representations such as looping constructs, conditionals, etc.

The Functional level is responsible for low-level control of the robot. This level typically consists of real-time control loops that directly command the rover hardware, and which tightly couple sensors to actuators.

This three-level approach has been successfully tested on a number of robotic applications [1, 2, 8, 10] but has several significant limitations. One problem is that each level has its own representation and thus several different models of the robot and its environment must be created and maintained. This repetition of information storage is often redundant and causes additional overhead in maintaining the system over time. A second problem is that each level is constrained to work on problems at a certain level of granularity. The planner works at the highest level, the executive at a level below that, etc. This setup often prevents planning or executive techniques from being used on problems where they would be most appropriate. A third problem is that this type of architecture does not account for new research in the areas of planning and execution that blurs the line between the two levels and that has significantly increased the response time of these types of systems. A proposed solution to these problems is discussed in a later section describing the CLEaR system.

2.2 CLARAty Two-Layer Architecture

To correct these shortfalls, CLARAty provides an evolution to a two-tiered architecture, illustrated in Figure 1. This structure has two major advantages: 1) enabling each tier to operate at all levels of granularity (or abstraction) and 2) blending declarative and procedural techniques for decision making.

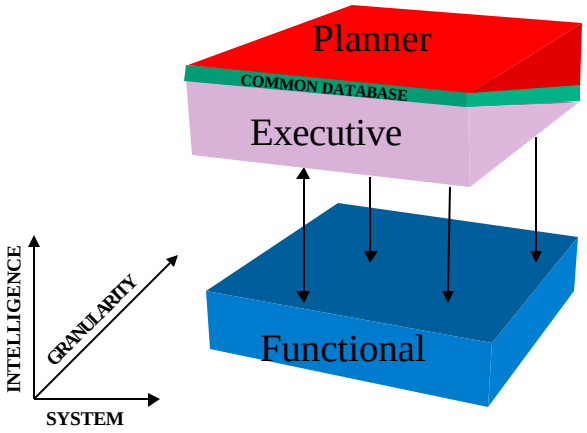

Figure 1: CLARAty Two-Layer Architecture

Adding a granularity dimension to each layer allows for the *de facto* nature of planning horizons in the Decision Layer and for the explicit representation of system hierarchies in the bottom Functional Layer. For the Functional Layer, an object-oriented decomposition describes the system's nested encapsulation of subsystems, and provides basic capabilities at each level of nesting. For instance, a command to "move" could be directed at a motor, appendage, mobile robot, or team of robots. For the Decision Layer, granularity maps to the timescale of the activities the Decision Layer can schedule. Due to the nature of the dynamics of the robot system controlled by the Functional Layer, there is a strong correlation between the Functional Layer system granularity and the timescale granularity of the Decision Layer. However, each layer represents activity and other domain knowledge using different representation formalisms.

The blending of declarative and procedural techniques in the Decision Layer emerges from the trend of planning and scheduling systems that have executive type qualities and vice versa [4, 14, 15]. This merging of techniques has been supported by algorithm and system advances, as well as faster processing capabilities. CLARAty embraces this trend by supporting closely integrated planning and executive systems. It provides a single database to store and interface functionality from both these approaches and enables both declarative and procedural approaches to be applied at different granularity levels.

Next, we develop these concepts by providing an overview of features of the Decision and Functional Layers, as well as the connectivity between them.

2.3 Decision Layer

The Decision Layer breaks down high-level goals into a plan of activities that successfully coordinate Functional Layer capabilities in achieving the goals. The plan must obey any relevant domain or mission constraints, such as resource limitations or instrument operation rules. Specifically, this layer consists of a hierarchical structure that overlays the Functional Layer. As shown in Figure 2, the Decision Layer can be thought of as a triangle that represents the "robot planning space." Here, a set of high-level goals is elaborated into a detailed network of goals and activities that represent the current plan. Goals that are elaborated outside of the triangle usually correspond to higher-level mission goals that are not part of the planning space for the particular robot being controlled by CLARAty.

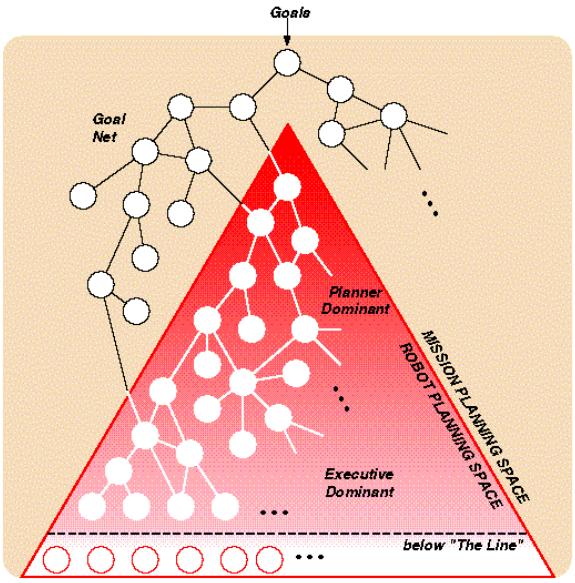

Figure 2: Decision Layer

The darker (top) portion of the triangle is the region of the robot planning space that is handled primarily through planning functions. The lighter (bottom) portion of the triangle is the region of the robot planning space that is handled primarily through executive functions. The line between these two regions is considered fuzzy since executive and planning processes may be tightly coupled and may even share the same representation.

The bottom fringe of this activity network is where the Decision Layer interfaces with the Functional Layer. This interface point is shown by the dashed

black line (called “The Line”). During plan execution, capabilities in the Functional Layer will be called, and results of these actions are monitored to allow the plan to be iteratively modified in response to changing events or conditions. This interface line is flexible and may be moved up or down depending on how much control and elaboration the Decision Layer is responsible for. This floating interface line provides flexibility in the ways the two layers may be connected. At one end of the spectrum is a system with a very capable Decision Layer, and with a Functional Layer that provides only basic services. At the other end of the spectrum is a system with a very limited Decision Layer that relies on a very capable Functional Layer to execute high-level commands. This flexibility enables the user and robot domain to dictate the full capabilities of each layer.

The Decision Layer can also access the Functional Layer to request current state information and resource estimations for future planned activities. Though most three-layer approaches allow the current rover state to be updated in the top levels, none enable the Functional Level to provide predictive information for resources. Instead, this information is usually provided by simple models at the planning level. However, in CLARAty, detailed models and predictive engines for this information are kept in the relevant Functional Layer components for each resource. These models are also needed by the Functional Layer for its control operations. This organization enables more detailed models to be maintained and encapsulates this information in one logical place. The Decision Layer can then query for this predictive information during plan creation. Examples of resource queries are how much battery power is required by an arm operation or how much memory storage is needed to hold data from a science operation. The Decision Layer can also request queries from the Functional Layer in different degrees of resolution. Thus, the level of computation and detailed analysis performed for a resource estimate can depend on factors such as the criticality of the activity using the resource or the amount of time available for planning.

2.4 Functional Layer

The Functional Layer is responsible for providing basic robot functionality using a set of generic components that have predefined behavior. These components attach to their hardware counterparts when the Functional Layer is deployed on a real system. The functionality of components can range from low-level control of a single motor or sensor to system level operations such as traversing a rover to a goal using obstacle avoidance.

The Functional Layer also provides an interface to all system hardware and its capabilities, including nested logical groupings and their resultant capabilities. These capabilities are the interface through which the Decision Layer uses the robotic system. Figure 3 shows a very simplified and stylistic representation of the Functional Layer. Since this paper focuses on the CLARAty Decision Layer, we only provide a brief description of the Functional Layer in this section. For further information on the CLARAty Functional Layer, please see [13, 17].

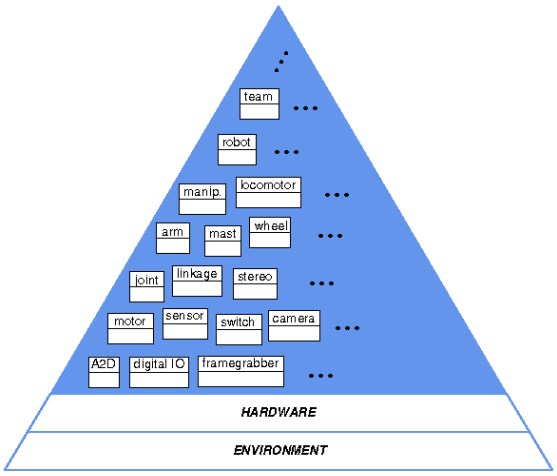

Figure 3: Functional Layer

The Functional Layer has a number of important characteristics. One, it has an object-oriented design that can be structured to directly match the nested modularity of the hardware and allows for basic functionality and state information of the system components to be encoded and compartmentalized. Two, all objects contain basic functionality for themselves that is accessible from within other pieces of the Functional Layer as well as directly from the Decision Layer. Three, the state of the system components is contained in the appropriate Functional Layer object and is obtained from it by query. Thus, the Decision Layer can obtain estimates of current state or predictions of future state, for use in planning and execution monitoring. Four, the Functional Layer may utilize local planners that are part of its subsystems. For instance, path planners and trajectory planners, can be attached to manipulator and vehicle objects to provide standard capabilities without regard to global optimality (which is a Decision Layer concern). Finally, the Functional Layer is also intended to interface to rover simulators as well as actual hardware. The details of this interaction are

hidden from the Decision Layer so that changing between testing on hardware and simulation is seamless for the Decision Layer software.

3 Decision Layer Implementation

We are currently developing the first instantiation of the CLARAty Decision Layer. This section discusses this implementation and gives an overview of the particular systems and techniques being utilized.

3. 1 Utilization of CLEaR System

In keeping with CLARAty's support of integrated planning and executive functionality, the first instantiation of CLARAty is utilizing the CLEaR (Closed-Loop Execution and Recovery) planning and execution system [6, 7]. CLEaR is a hybrid controller system that is built on top of the CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) continuous planner [4] and TDL (Task Description Language) executive system [15]. CASPER provides a soft-real-time capability for performing plan generation, execution, monitoring and re-planning. To increase CASPER's limited executive capabilities, CLEaR integrates CASPER with TDL so that the full spectrum of executive capabilities can be supported. Past versions of CLEaR have been demonstrated for Deep Space Network (DSN) antenna control [6]. It is currently being extended to provide planning and execution support for planetary rovers.

A main object of the CLEaR system is to provide a tightly-coupled approach to coordinating goal-driven and event-driven behavior. Most past approaches have followed the three-level architecture style of separating planners and executives. In this framework each system is treated as a "black box," has its own plan representation, and operates at a particular level of plan granularity. In general, executives provide event-driven behavior that enables a robotic system to quickly react to changes in its environment and modify its command sequence appropriately. Planners provide goal-driven behavior that enables a robotic system to accept high-level goals rather than low-level instructions.

Though separating these capabilities works for some applications, there are many situations (in robotics and other domains) where it would be beneficial to have event-driven capabilities available at a higher activity granularity. For instance, sometimes a conditional reaction or looping behavior may be required in a high-level activity or in an activity scheduled significantly in the future (where these types of activities are typically managed by the planner). The sooner such a behavior is properly reflected in a plan, the sooner information on relevant state and resource usage can be propagated and reasoned about. Furthermore, it would be beneficial to have goal-driven capabilities available on a shorter time scale. For example, there may be low-level activities whose resource usage we want the planner to track and reason about, even when these activities must be quickly modified in response to current state information. If the executive makes a decision about expanding an activity, the planner could influencing that decision in an optimal manner by performing a global-resource analysis on how that expansion affects the overall plan. Without planning-type capabilities available on a shorter time scale, many activity resource and state effects must be handled using a worst-case approximation that can significantly affect plan optimality and flexibility during execution.

The CLEaR approach to how planning and executive behaviors are utilized is shown in Figure 4. Here, the planner and executive operate on the same set of activities and timelines and all capabilities are allowed on both near- and far-term activities. The shaded activity areas of the figure show where the planner and executive are active. The executive is primarily active on a short-term basis but can be used to refine long-term activities. Similarly, the planner is primarily active on a long-term basis but can be used to plan for short-term activities. A separate module in CLEaR decides what functionality is used on what activities and synchronizes the two sets of capabilities.

Currently in CLEaR, CASPER and TDL still maintain separate representations, however plan databases (which hold the current plan for each system) are coupled where changes in one database can be reflected in the other. Thus, if the planner makes a change to the plan, this change can be reflected in the executive database, and vice versa. CLEaR also provides heuristic support for deciding when a plan conflict should be handled by the planner (CASPER) vs. the executive (TDL). For instance, if a rover gets off track during a traversal, both the planner and executive may react and these reactions need to be coordinated. A simple heuristic for this situation is to have the executive react if only the current traversal activity needs to be re-expanded but the overall activity can still be completed within a certain window of the original estimated time. The planner reacts only if global plan changes are required (e.g., the rover is so far off track that other plan activities must be re-arranged).

Future work on CLEaR will tighten this integration. One future step is to enable TDL procedural capabilities to be accessed by CASPER during initial plan generation. This step will enable procedural constructs, such as loops and conditionals, to be easily utilized during planning and re-planning. Currently,

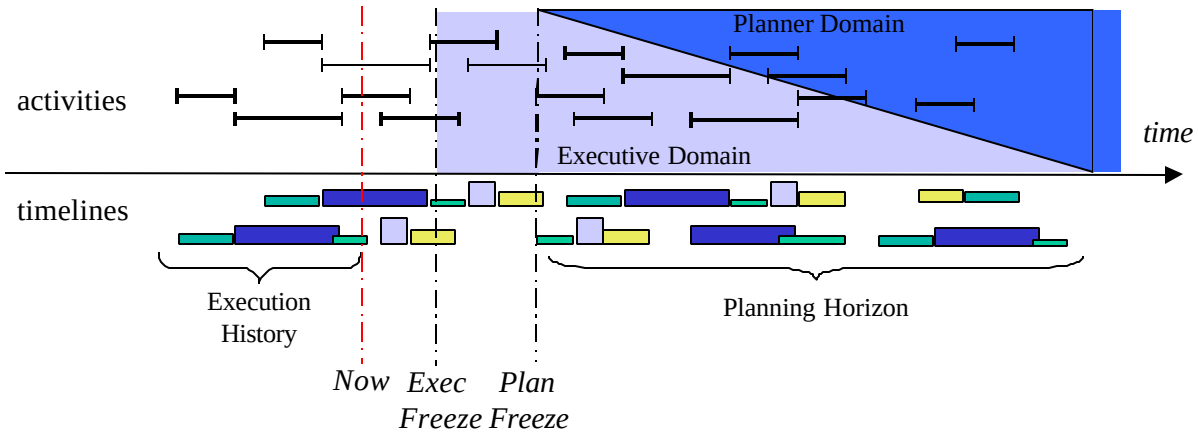

Figure 4: Domains of Planner and Executive in CLEaR System

these types of constructs are difficult to represent in CASPER's declarative representation. Another future step will be to increase TDL's knowledge of resource levels and to have CASPER's global resource analysis affect some TDL decisions. (Currently, TDL offers only limited support for resource management.) Finally, we plan to fully integrate these systems, where both planning and executive functionality use a completely shared representation and operate on one planning database. This will alleviate the need for two different domain models and will enable planning and executive functionality to be easily used at all levels of activity granularity.

3.2 CASPER Planner

Planning in CLEaR is performed by the CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) planning system [4]. Based on an input set of science goals and a rover's current state, CASPER generates a sequence of activities that satisfies the goals while obeying each of the rover's resource constraints and operations rules. Plans are produced by using an "iterative repair" algorithm that classifies conflicts and resolves them individually by performing one or more plan modifications. CASPER also monitors current rover state and the execution status of rover activities. As this information is acquired, CASPER updates future-plan projections. From these updates, new conflicts and/or opportunities may arise, requiring the planner to re-plan in order to accommodate the unexpected events. CASPER has been successfully demonstrated in a number of robotic domains, including command generation for the landed operations part of the ST4 mission (which involved landing a spacecraft on a comet) [4], control of a DSN antenna ground station [6] and coordination of distributed operations for multiple rovers [5].

3.3 TDL Executive

Most executive functionality in CLEaR is performed by the TDL (Task Description Language) executive system [15]. TDL was designed to perform task-level control for robotic control and to mediate between a planner and more low-level rover control software in a robot architecture. It expands abstract tasks into low-level commands, executes the commands, monitors their execution, and handles exceptions. TDL is implemented as an extension of C++ that simplifies the development of robot control programs by including explicit syntactic support for task-level control capabilities. It utilizes a construct called a "task tree" to describe the tree structure that is produced when tasks are broken down into low-level commands. TDL directly support task decomposition, fine-grained synchronization of subtasks, execution monitoring, and exception handling. TDL has been successfully demonstrated on a number of indoor and outdoor robots, including the Nomad robot used for the Antarctica 2000 initiative [12] and the Bullwinkle RWI robot used for Mars autonomy navigation [16].

4 Current Interface to the Functional Layer

In the first implementation of CLARAty, the Decision Layer interfaces with the Functional Layer in several ways. First, after plans are constructed by using the CLEaR integration of CASPER and TDL, low-level commands are relayed to the correct Functional Layer objects. Currently, TDL is responsible for relaying all

commands to the Functional Layer whether or not they were further expanded by TDL. These commands are relayed to generic Functional Layer objects that can break the commands down into more specialized steps for a particular rover. After a command has been executed, the Functional Layer returns an execution status to the Decision Layer reflecting the success or failure of the command. This status is tracked by TDL and in the case of failure, TDL can either attempt to fix the plan itself or signal failure to CASPER so that re-planning can be invoked.

Second, the Decision Layer queries the Functional Layer for state and resource information. A query is for a single time point or for an iterative return of state over a time interval. For instance, during rover traverses, the Decision Layer can instruct the Functional Layer to iteratively (e.g., every second) return the rover's estimate of its current position. Allowing state updates to be done continuously over a certain time interval cuts down on the number of queries performed and ensures that states are only updated when necessary (e.g., rover position is only updated when the rover is moving). Other state and resources that are useful to update include power levels, such as battery, onboard memory capacity, sun angle, and temperature.

Last, when formulating a plan, the Decision Layer queries the Functional Layer for resource prediction estimates associated with particular activities. For instance, when scheduling an arm movement, the Decision Layer will query the Functional Layer manipulator object to determine how much battery and solar power the arm operation will require. For this first implementation, resource querying can only be instigated during planning search. However, future instantiations of this architecture may utilize this capability during executive expansions. Resource queries can also be at different levels of granularity. For demonstrations this year, two levels of queries will be performed. A simple resource query for the amount of power used during traverses will return a simple scalar value. A more detailed power query for traverses will return a vector of values. Queries will also be performed for the memory requirements of science operations. Results of these queries will be used by the planner to better estimate the future plan.

Future demonstrations will also highlight the flexibility of "The Line" between the Decision Layer and the Functional Layer. In the current implementation both the Decision Layer and Functional Layer are responsible for expanding activities at particular granularity levels. Currently the Decision Layer expands goals down into activities that use major rover effectors (e.g., arm, camera, mast) and predicts the resource usage for such operations. The Functional Layer then expands these activities into more detailed steps that handle the low-level control of these effectors. The Functional Layer also interprets sensor data and produces estimates that are mapped to state timelines (e.g., rover position, battery power availability) maintained by the Decision Layer.

5 Related Work

A number of planning and executive systems have been successfully used for robotic applications and have similarities to the CLARAty Decision Layer. Most of these approaches have utilized some form of the standard three-level architecture.

The Remote Agent Experiment [10] (RAX) was flown on the NASA Deep Space One (DS1) mission. It demonstrated the ability of an AI system to respond to high-level spacecraft goals by generating and executing plans onboard the spacecraft. The planner in RAX takes as input a schedule request and produces a flexible, temporal schedule for execution by its executive. Both the planner and executive used different representations and strictly operated on different granularity levels. A major limitation to this approach was that planning was only performed in a batch fashion. If re-planning was required, the spacecraft was "safed" until a new plan had been generated (which could be on the order of hours).

Another approach directed towards rover command generation utilizes a Contingent Planner/Scheduler (CPS) that was developed to schedule rover-scientific operations using a Contingent Rover Language (CRL) [3]. CRL allows both temporal flexibility and contingency branches in rover command sequences. Contingent sequences are produced by the CPS planner and then are interpreted by an executive, which executes the final plan by choose sequence branches based on current rover conditions. In this approach, only the executive is onboard the rover; planning is intended to be a ground-based operation.

Other three-tier approaches include Atlantis [9] and 3T [2], which both utilize a deliberative planner and executive (or sequencing component) on top of a set of reactive controllers. The LAAS-CNRS lab also developed a robot control architecture that contains both a decision and execution level and that balances planning and reactive capabilities [1].

Other systems have also looked at closely integrating planning and execution. The CPEF (Continuous Planning and Execution Framework) [14] is a similar framework to CLEaR for combining planning and execution. CPEF attempts to cull out key aspects of the world to monitor (as is necessary in general open-world domains). CPEF also uses iterative repair to fix plan conflicts under the term "conservative repairs."

6 Conclusions

This paper discusses how intelligent decision-making is performed for the CLARAty architecture for robotic autonomy. Specifically, the top "Decision Layer" of CLARAty is presented. This layer provides support for the new trend in planning and executive systems to closely merge their approaches, which provides more flexibility in creating robot plans. This layer interfaces with a "Functional Layer" that provides robot behaviors and control. The interface between these two layers is flexible so that different instantiations of the architecture can use different levels of Decision Layer and Functional layer capabilities. The architecture is currently being applied to several robotic efforts at JPL and is being directed towards future flight implementations.

7 Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

8 References

[1] R. Alami, R. Chautila, S. Fleury, M. Ghallab, and F. Ingrand, "An Architecture for Autonomy," *International Journal of Robotics Research*, 17(4) April, 1998.

[2] R. Bonasso, R. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack, "Experiences with an Architecture for Intelligent, Reactive Agents," *Journal of Experimental and Theoretical Artificial Intelligence Research*, 9(1), 1997.

[3] J. Bresina, K. Golden, D. Smith, and R. Washington, "Increased Flexibility and Robustness of Mars Rovers," *Proceedings of the 1999 International Symposium, on Artificial Intelligence, Robotics and Automation for Space*, Noordwijk, The Netherlands, June 1999.

[4] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the 5th Intl. Conference on Artificial Intelligence Planning and Scheduling*, Breck-enridge, CO, April 2000.

[5] T. Estlin, A. Gray, T. Mann, G. Rabideau, R. Castano, S. Chien, and E. Mjolsness. "An Integrated System for Multi-Rover Scientific Exploration." In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL, July 1999.

[6] F. Fisher, R. Knight, B. Engelhardt, S. Chien, and N. Alejandre, "A Planning Approach to Monitor and Control for Deep Space Communications," *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, March 2000.

[7] F. Fisher, M. James, L. Paal, and B. Engelhardt "An Architecture for an Autonomous Ground Station Controller," *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2001

[8] E. Gat, "On Three-Layer Architectures," In *Artificial Intelligence and Mobile Robots*, Eds., D. Kortenkamp, R. Bonnaso, and R. Murphy, Boston, MA, 1998.

[9] E. Gat., "ESL: A Language for Supporting Robust Plan Execution in Embedded Autonomous Agents," *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, July 1992.

[10] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith, "Planning in Interplanetary Space: Theory and Practice," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge*, CO, April 2000.

[11] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, B. Wilcox, "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," *Proceedings of the 1998 IEEE Aerospace Conference*, Aspen, CO, March 1998.

[12] S. Moorehead, R. Simmons, D. Apostolopoulous, and W. Whitaker, "Autonomous Navigation Field Results of a Planetary Analog Robot in Antarctica," *Proceedings of the 1999 International Symposium on Artificial Intelligence, Robotics and Automation for Space*, Noordwijk, The Netherlands, June 1999.

[13] I. Nesnas, R. Volpe, T. Estlin, H. Das, R. Petras, and D. Mutz, "Toward Developing Reusable Software Components for Robotic Applications," submitted to *International Conference on Intelligent Robots and Systems*,Maui, Hawaii , Nov 2001.

[14] K. Myers. "Towards a framework for continuous planning and execution." *Proceedings of the AAAI 1998 Fall Symposium on Distributed Continual Planning*, Menlo Park, CA, 1998.

[15] R. Simmons and D. Apfelbaum, "A Task Description Language for Robot Control," *Proceedings of the Intelligent Robots and Systems Conference*, Vancouver, CA, October 1998.

[16] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja and K. Schwehr, "Recent Progress in Local and Global Traversability for Planetary Rovers," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

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