

CLEaR: Closed Loop Execution and Recovery – A Framework for Unified Planning and Execution

Forest Fisher, Tara Estlin, Daniel Gaines, Steve Schaffer, Caroline Chouinard, Russell Knight

Abstract

Intelligent behavior for robotic agents requires a careful balance of fast reactions and deliberate consideration of long-term ramifications. The need for this balance is particularly acute in space applications, where hostile environments demand fast reactions, and remote locations dictate careful management of consumables that cannot be replenished. However, fast reactions typically require procedural representations with limited scope and handling long-term considerations in a general fashion is often computationally expensive.

1 Introduction

Robotic agents performing under hard resource and time constraints in uncertain environments require careful balancing of both deliberative and reactive reasoning [Knight, et al, 2001]. As in most domains with uncertainty, a task may fail or produce unexpected results leading to plan failures. If the robot is also under hard time deadlines and resource constraints, a task requiring a different time or resource allocation than planned could cause failure at future points in the plan. In some cases, the robot may be able to retry a failed task, use more time or take up more of a resource without causing a problem.

In our discussions, we will focus on the higher-level autonomy issues of balancing reaction and deliberation, and declarative and procedural representations in particular as they relate to planning and execution.

2 Issues in Planning and Execution

We define autonomy as the ability of a system to handle complex tasks involving environmental feedback without external intervention or supervision. Examples include the operation of free-flying robotic spacecraft for mapping and science observation missions, the operation of surface rovers on extra-terrestrial missions, and the tracking of spacecraft using ground communications stations. While there are many aspects of autonomy, we focus on issues arising from automated planning and scheduling in the context of real-time task execution. Aspects of autonomy we explore include system plan and task representation, system control, plan generation, plan execution, monitoring, and error handling.

2.1 Knowledge Representation

A complete autonomous system consists of hardware, software (engines) and models. For the purposes of this article, we will focus on software and models. Software and models may be defined either declaratively or procedurally. Software (e.g., a planning engine or task execution engine) provides the control information necessary to use the information provided by models. A model is the knowledge about a domain or situation.

Procedural models represent domain knowledge by embedding it in a control stream. This is sometimes referred to as “arbitrary code.” It is often quite difficult to separate the control information from the domain information in procedural models. Declarative models are static representations about events, objects, and their relationships. The software required to use a declarative model may be quite complex; this is the topic of considerable Artificial Intelligence research.

The tradeoff between procedural models and declarative models is that while procedural models can be quickly encoded for specific domains, conventional wisdom is that (compared to declarative models) procedural models are brittle and difficult to change. Declarative models do not commit to any particular

control path, and thus in theory can be more flexible with respect to uncertain and unknown events. However, the software that uses these models can be slow and often requires more computational resources than software using procedural models.

2.2 Plan Generation and Execution

Plan (and/or schedule) generation is the act of devising a set of actions (the plan) to realize a task or set of goals. In order to realize a task or achieve a goal, a system may use deliberation and/or reaction. Deliberation is the process of producing a collection of executable sub-tasks that when executed result in the realization of tasks/goals. This collection of executable sub-tasks is referred to as a plan. Deliberation is the search required to find an appropriate plan and typically cannot guarantee finding a solution.

Reaction is the act of producing the next executable sub-task required to achieve a task. Therefore, reactive systems generate a plan one step at a time, only generating the next step in the plan and waiting for the results before continuing. Reaction performs little or no search, and thus is able to provide a bounded time response. Reaction is a very powerful technique that deals well with real-time issues. However, reaction has the weakness that it may take harmful actions that result in suboptimal performance or even unnecessary failure because it does not perform look-ahead. Even though it makes its decisions locally without look-ahead, reaction still has many powerful capabilities such as the ability to respond to execution errors in a timely fashion through generic error handlers and task trees as well as the ability to synchronize multiple execution threads.

The tradeoff between deliberation and reaction is that deliberation may require more resources (time and computation) than reaction, resulting in missed deadlines. Reaction may waste resources trying to achieve a task, and might not solve the problems associated with achieving a task at all. Further deliberation allows for plan optimization, and the ability to solve the problem within the global context of the robotic system and/or mission desires. We see the use of both techniques to achieve robust autonomy.

Plan execution is the act of realizing a task given a plan. Technologies used for plan execution include (among others) execution software and mode identification. Execution software takes the representation of a plan and controls the hardware such that the tasks in the plan are achieved. Mode identification is the act of using sensor information and plan context to determine the state of the environment and the state of the autonomous system. The combination of execution software and mode identification allows plan execution to achieve tasks of a plan and to know that the tasks have been achieved. Likewise, failure to achieve a state can be known and reported back to the plan generation system. Reactive plan generation immediately provides a new task (probably very similar to the failed task) to be executed. If plan generation is deliberative, a new plan or revised plan is produced.

We believe that declarative and procedural representations of models as well as deliberative and reactive plan generation are required for intelligent robotic systems.

3 A Unified Planning and Execution System for High-Level Robotic Automation

To address the issues outlined in the previous section, we have developed a system for high-level decision-making capabilities for future autonomous robotic missions. CLEaR (Closed-Loop Execution and Recovery) is currently comprised of two major components with hooks for a third:

- **A Continuous Planner** that provides capabilities for initial command sequence (plan) generation and continuously updating of that plan (i.e., re-planning) based on changing operating context and goal information.
- **A Reactive Task-Level Executive** that provides task-level control capabilities for a robotic system, including execution and monitoring of the plan, as well as mediation between a planner and low-level robotic functionality.

- **Domain Specific Solvers:** For example in the rover domain, a **Global Path Planner** that provides global path planning information about predicted routes to both the Planner and Executive is plugged into the framework.

We will begin by first introducing the CLEaR framework for unified planning and execution, then give a more detailed description of the individual components the framework is comprised of, and then discuss how the framework addresses many of the issues presented. In the next section we will briefly describe how this framework has been used.

3.1 CLEaR Framework

The CLEaR unified planning and execution framework (Fisher, et al., 2002) was developed to pursue a tight integration of planning and execution capabilities. Currently, CLEaR is a hybrid controller system that is built on top of the CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) continuous planner and the TDL (Task Description Language) executive system. CASPER provides a soft-real-time capability for performing plan generation, execution, monitoring and re-planning. Versions of the CLEaR framework have been demonstrated for Deep Space Network (DSN) antenna control (Fisher, et al., 2001), and planning and execution support for planetary rovers (Estlin, et al., 2002).

CLEaR's primary objective is to provide a tightly coupled approach to coordinating goal-driven and event-driven behavior. Many past approaches have followed a three-level architecture style where the planning and executive processes are treated as *black box* systems. This is in contrast to how CLEaR enables the planner and executive to interact with each other and more effectively share the responsibility for decision making. In part this is managed through shared plan information and continual updates of state being made available to both the planner and executive. CLEaR also provides heuristic support for deciding when certain plan conflicts should be handled by the planner vs. the executive. For instance if a rover gets off track during a traverse, the reaction of the planner and executive need to be coordinated. If the executive believes it can resolve the navigation delay within the original allotted time it will manage the plan changes, but once the executive identifies that the repair will require more time or resources than allotted by the planner, it will then allow the planner to use its global perspective to fix the problem.

In Figure 1, we graphically depict the concept of shared responsibility in the plan modification process. In the bottom half of the figure the timelines depict states and resource being affected by the plan activities. In the top half of the figure the I-bars represent planning activities with the left edge of the bar representing its start-time and its length/size depicting the duration of the activity. In this figure, time is advancing from left to right and is marked by the *Current Time* marker. As you move into the future (the right) the changing thickness of the wedges are depicting the decreasing responsibility of the executive and the increasing responsibility of the planner.

3.1.1 CASPER Planner

Planning in CLEaR is provided by the CASPER system (Chien, et al., 2000). 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 relevant 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 plan activities. As this information is acquired, CASPER updates future-plan projections. Based on this new information, new conflicts and/or opportunities may arise, requiring the planner to re-plan in order to accommodate the unexpected events.

3.1.2 TDL Executive

Most executive functionality in CLEaR is performed by the TDL executive system (Simmons and Apfelbaum, 1998). TDL was designed to perform task-level control for robotic control and to mediate between a planning system and low-level robot control software. It expands abstract tasks into low-level commands, executes the commands, and monitors their execution. It also provides direct support for exception handling and the fine-grained synchronization of subtasks. 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 uses a construct called a *task tree* to describe the tree structure that is produced when tasks are broken down into low-level commands.

3.1.3 An Instantiation of the CLEaR Framework

Figure 2, depicts an instantiation of the CLEaR framework for a rover domain. In this instantiation the CASPER planner selects planning activities to be executed by the TDL executive, which in turn commands the lower level robotic control software (CLARAty Functions Layer). As CLARAty carries out the commands, command status (*in progress, completed, etc.*), and vehicle state and resource updates are provided. This enables TDL to perform execution time monitoring of the tasks being performed. Similarly any of these updates that pertain to information being tracked in the planner include planning activity status is provided to the planner. CASPER then uses these updates to project the state of the agent forward in time. If these updates create inconsistencies with the projected state, operations constraints, or mission planning goals then the plan is modified through replanning. In the rover domain the planner and executive both query a Path-Planner for information like distance between to locations. This information is used to sequence targets and provide estimates on how long traverses will take, which in part impacts the number of activities that can be placed into the plan.

3.2 Unified Planning and Execution

In general, it seems to be agreed that for automation tasks involving tight time constraints and hard resource and state constraints that both deliberative and reactive types of reasoning are necessary. To date, many approaches have combined the deliberative planning process and the reactive executive in a *black-box* fashion. This makes the tracking of planning constraints difficult for the executive to do during reactive execution of the plan. By unifying the planning and execution process the passing or access to constraint information is simplified. Although less work has been done to date on unifying the representation, this unified approach also reduces the need to duplicate system model information (hopefully reducing development time, modeling errors and difficulty of validation) in the planning and executive processes because both access the same runtime information.

In section 2 we identified that a reactive approach can be short sighted, while a deliberative approach can be computationally expensive (time consuming). The use of the continuous planning approach enables CLEaR to replan more frequently thus solving smaller problems enabling the deliberative process to occur more quickly. CLEaR also attempts to identify the need to replan sooner by coupling the execution monitoring with the plan updates, and by doing simple reasoning in the executive to predict a likelihood of success or a need to replan.

4 Applied Domains

The CLEaR framework has been applied to primarily two domain areas: onboard decision-making for Mars surface exploration rovers, and Deep Space Network (DSN) ground station communication antenna station automation. To date both of these applications of the CLEaR framework have taken place under research/technology development efforts here at JPL.

4.1 Rovers:

In conjunction with the CLARAty (Coupled-Layer Architecture for Robotic Autonomy) task (Volpe, et al., 2001), the CLEaR framework has been applied to provide the first instantiation of the CLARAty Decision-Layer (DL) (Estlin, et al., 2001). In this application CLEaR provides command sequence generation, execution, monitoring and onboard replanning of a rover, its science instruments and resources in an attempt to maximize science return based on a set of high-level goals (objectives) selected by a mission/science team and the model describing the capabilities of the rover along with the operations constraints, which could include flight rules. This is in contrast to the current state-of-the-art practice where command sequences are generated on the ground and uploaded to the rover for execution, but there is little ability to adapt the sequence in response to unexpected events. In Figure 3, we show a picture of Rocky-7 and Rocky-8, the research rovers that our software has been applied to. In Figure 4, we show a set of science targets, rocks, a planned path and the actual path traversed to the science targets used during the 2001 year end demonstration of the CLEaR software running as part of the CLARAty level one milestone.

4.2 DSN:

As a component of the Deep Space Station Controller (DSSC) or Common Automation Engine (CAE) technology demonstration effort, CLEaR is used to provide antenna station subsystem command sequence generation, execution, monitoring, and replanning to provide robust execution of downlink communication passes by configuration and commanding the appropriate subsystems. CLEaR generates and modifies the command sequence by dynamically piecing together smaller sequences in order to achieve the desired equipment state. This is in contrast to the current state-of-the-practice of executing static rigid sequences, which are not well adapt to unexpected situations like subsystem resets. In this application, CLEaR monitors the progress of the command sequence with the aid of a very capable fault detection and isolation (FDI) component that provides system wide monitoring and contributes to state estimation. In figure 5, we show a picture of a cluster of 34 meter Beam Waveguide antennas at Goldstone, CA. One of these antennas was usually the station used during the validation phases of the DSSC task.

5 Acknowledgements

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

6 References

Chien, S., Knight, R., Stechert, S., Sherwood, R., and Rabideau, G., "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," Proceedings of the 5th Int'l Conference on Artificial Intelligence Planning and Scheduling, Breck-enridge, CO, April 2000.

Estlin, T. Fisher, F., Gaines, D., Chouinard, C., Schaffer, S., and Nesnas, I., "Continuous Planning and Execution for an Autonomous Mars Rover," 3rd International NASA Workshop on Planning and Scheduling for Space, Houston, Texas, October 2002 (to appear).

Estlin, T., Volpe, R., Nesnas, I., Mutz, D., Fisher, F., Engelhardt, B., and Chien, S. "Decision-Making in a Robotic Architecture for Autonomy," Proceedings of the Intl Symposium, on AI, Robotics and Automation for Space, Montreal, Canada, June 2001.

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

F. Fisher, D. Gaines, T. Estlin, S. Schaffer, C. Chauinard, "CLEaR: A Framework for Balancing Deliberation and Reactive Control," Proceedings of the AIPS On-line Planning and Scheduling Workshop, Toulouse, France, April 2002

Knight, R., Fisher, F., Estlin, T., Engelhardt, B., and Chien, S., "Balancing Deliberation and Reaction, Planning and Execution for Space Robotic Applications," In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Maui, Hawaii, Oct 2001.

Nenas, I., Volpe, R., Estlin, T., Das, H., Petras, R., and Mutz, D., "Toward Developing Reusable Software Components for Robotic Applications," Proceedings of the Int'l Conference on Intelligent Robots and Systems, Maui, Hawaii, Nov 2001.

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

Volpe, R., Nenas, I., Estlin, T., Mutz, D., Petras, R., Das, H., "The CLARAty Architecture for Robotic Autonomy," Proceedings of the 2001 IEEE Aerospace Conference, Big Sky, Montana, March 2001.