Traded Control with Autonomous Robots as Mixed Initiative Interaction

David Kortenkamp, R. Peter Bonasso, Dan Ryan and Debbie Schreckenghost
Metrica Inc. Robotics and Automation Group
NASA Johnson Space Center – ER2
Houston, TX 77058
kortenkamp@jsc.nasa.gov

Abstract

This paper describes a problem domain that lends itself to mixed initiative interaction. The domain is traded control with an autonomous robot. Traded control is a situation in which a human wants to control a robot during part of a task and the robot is autonomous during other parts of a task. A significant problem in traded control situations is that the robot doesn't know how the environment has been changed or what parts of the task have been accomplished when the human has been in control. Because of this, errors can occur when the human relinquishes control back to the robot; these errors can cause potentially dangerous situations. Our solution is to use an intelligent software architecture designed for *autonomous* robot control and modify it to work in concert with human control. Using an architecture designed for autonomy allows us to use the monitoring functions designed to track the actions of the robot to monitor the actions of human agents for the same tasks. The intelligent software architecture includes a mixed initiative planner, an execution monitor, robotic skills and a user interface. This paper describes the problem domain and our initial attempts at defining a software architecture that operates in the domain.

Introduction

We want to establish effective human/robot teams that accomplish complex tasks. As members of human/robot teams, robots must be equal partners with humans in performing those tasks. The software systems controlling such robots must allow for fluid trading of control among team members, whether they be humans or robots. This is the essence of mixed initiative interaction. We believe that the problem domain of traded control of an autonomous robot, which is a subset of the human/robot teaming domain, is an interesting one for mixed initiative interaction. We have just begun a research project in this domain and in this paper we outline some of the issues that we are facing. Our research is framed by an intelligent control architecture that we use for controlling robots, so we will also present this architecture and our preliminary ideas on how to use it in mixed initiative situations.

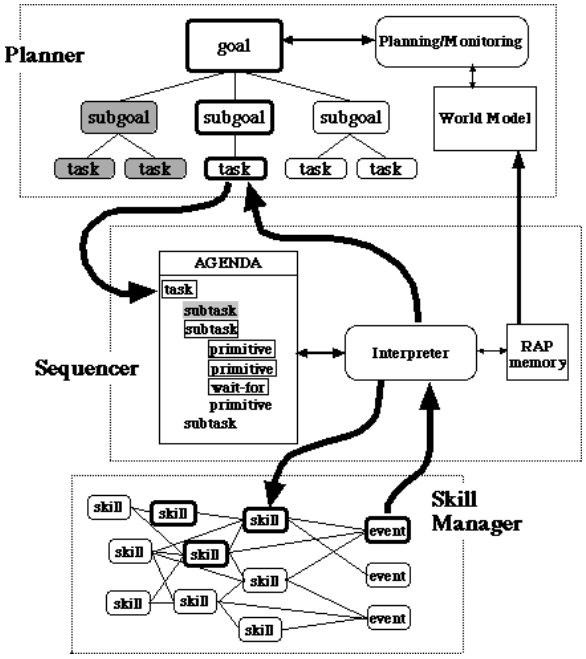

Figure 1: The 3T intelligent reactive control architecture.

An intelligent robotic software control architecture

Over the last several years, we 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 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*.

- 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).

Scale of control

Figure 2 shows a progression of control from complete teleoperation to full autonomy. The figure also maps these onto our intelligent software architecture. Here is a brief explanation of each:

1. **Teaming**: Robots and humans work as a team in which each has full autonomy, but they communicate to accomplish complicated tasks. Interaction with our architecture is at the planning level with goals given to the robot just as they are given to the other team members.

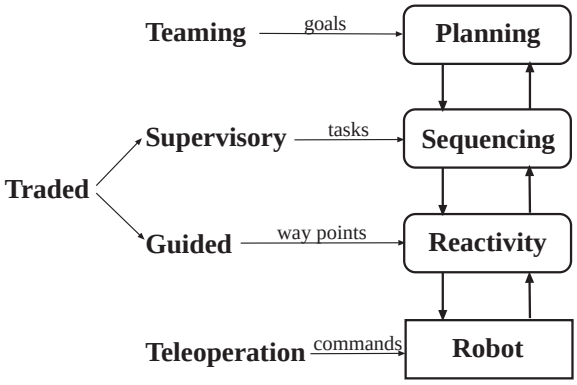

Figure 2: Levels of semi-autonomous control and how they map to the 3T architecture.

2. **Supervisory**: Robots work nearly autonomously, but a human is constantly watching and can stop or correct the robot at any time. Interaction is at the task (sequencing) level and the human has the opportunity to rearrange the robots task plan or to stop the robot completely.
3. **Traded**: Robots perform most tasks completely autonomously, but sometimes a human takes complete control to perform some difficult subtask or to extract the robot from a dangerous situation. The human then relinquishes control back to the robot to perform autonomously. Interaction with our architecture is at both the task (sequencing) level or through skills. This is really a mixture of supervisory and guided control.
4. **Guided**: A human is always guiding the robot through a task although the robot has some autonomous capabilities, such as obstacle avoidance or grasping, that allow for safer and faster operation. Interaction with our architecture is at the skill level.
5. **Teleoperation**: The human is in complete control of all robot movements at all times. The robot has no autonomy. Interaction is with the mechanical robot servos directly, bypassing the architecture completely.

While the ideal mixed initiative system will support all of these levels of control, we have only seriously developed a robot system that aims for the middle of this progression of control, namely *traded control* or number three above.

Issues in traded control

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 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 and 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 replan to account for the human intervention. Because our architecture has the ability to perform multi-agent planning and replanning with its top tier and the ability to recognize off-nominal situations with its middle tier, it is perfectly suited for traded control situations, as will be shown below.

Using 3T to resolve traded control issues

Our proposed solution to the problems outlined in the previous section is three-fold. First, we want to use the multi-agent, mixed initiative planner in the 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. Second, we want to use our architecture's sequencing and monitoring capabilities to allow the robot to "mentally" check off steps of the task as the human performs them. We can do this using *the same monitoring functions used in autonomous operations*. In essence the robot is watching and thinking, but not doing. Third, we would 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. Finally, an integrated graphical user interface will be needed to manage interaction with the human. We will examine some of these in detail in the next several subsections.

Mixed initiative planning for traded control

Planning for traded control confronts the planning system with a number of difficult challenges that revolve around the issue of the robotic system maintaining an accurate and coherent view of the world, other agents' views of the world, and status of pending tasks, as performed either by itself or its collaborators, what has been termed the *context registration* problem. Requiring that even a single human and single robot be able to engage in a traded control execution scenario this immediately moves us into a multi-agent domain of much higher complexity than that occupied by the typical autonomous robot. The system must not only plan for collaboration, it must also replan in case of unanticipated sequencing. We are already seeing evidence for this assertion even in our initial traded control implementations: the possibility of unknown and/or unanticipated actions by a human leads to the need to carefully specify at RAP design time, which steps when executed may or may not lead to a correctly updated world state. Utilization of a planner that can generatively produce tailored recovery plans would lessen the difficult burden on system builders to anticipate required recovery steps, and lessen the need to constantly perform general but resource intensive scans of the environment.

For the most part our planning focus in the past has been from a multi-agent perspective: the view that the planner plans *globally* for a set of agents in order that they achieve a common objective. In addition to purely autonomous multi-agent planning, however, we are now beginning to investigate the planning task from a different perspective; that of mixed-initiative planning in which agent roles are opportunistically negotiated amongst the agents in order to best solve the problem at hand (Ferguson, Allen, & Miller 1996; Ferguson 1995). Rather than central coordination, the mixed-initiative perspective is one of collaboration among equals. In practice, of course, the semi-autonomous system is rarely, if ever, on an equal footing with the human. What is meant is that each contributes in an interactive nature what each does best

in deriving plans of action.

We feel there are some advantages to moving from the multi-agent to mixed-initiative planning perspective for our traded control tasks, including:

- Increased flexibility of the overall solutions as the system and human agents collaborate synergistically on solving problems, each doing what it does best – e.g., the human performing sensing where it is superior such as in vision tasks, and the planning system rapidly evaluating many courses of action.

- More robust system behavior as it can participate in an interactive dialogue with the human agent for filling in unknowns about the domain, thus alleviating the context registration problem which we've found is so problematic for a traded control system.

- More tractable planning as the planner can rely on human agents to aid in constraining particularly unwieldy and combinatorially explosive parts of a plan tree through operator selection, variable instantiation commitment, and revised goal ordering.

- Increased user control and involvement in the planning process which can result in plans customized to the preferences of different classes of users.

Related to the idea of mixed-initiative planning is that of advisable planning systems (Myers 1996), where an advice-taking interface is developed for the planner. Such an interface expands the range of possible interactions between human and agent and thus shares many of the benefits of mixed-initiative planning. An advisable planning system may be viewed as a point on the planning perspective dimension which is approaching a true mixed-initiative approach. Currently, we have not begun applying the planning layer of the architecture to traded control situations.

Sequencing traded control tasks

Our research to date has concentrated on the conditional sequencing layer. This is the crucial layer in our architecture as it serves as a "differential" between the continuous, fast-paced real world and the symbolic, discrete, and deliberative planning world. Our conditional sequencer is based on the RAP system (Firby 1989), although we have, over the years, made many changes to the basic RAP system that made implementation of traded control much easier. For example, we extended the RAP system to allow for control of multiple agents (i.e., multiple skill managers). This enhancement allows a human to be one of the other agents that could be used by the RAP system.

Preliminary work All of our preliminary implementations have focused on the sequencing layer of our architecture. We have two domains in which we apply traded control. First, we have a simulated "StationWorld" domain in which a robot can move around a space station and perform tasks. Second, we have an upcoming flight project in which certain Shuttle Remote Manipulator System (SRMS) operations will be automated using our architecture. The key aspects of traded control for man-machine teams in both of these domains are the following:

- **Changes in the level of autonomy** Robot operation at three levels of autonomy is desired: (1) autonomous; (2) semi-autonomous (for traded control); and (3) tele-operated (fully manual). All changes in level are initiated by the human.

- **Human-robot task allocation** Our implementation of traded control allocates to the human the choice of an agent to perform a task. While in semi-autonomous mode, the robot can perform single primitive actions without human intervention, but after each action the robot must be authorized by the human to proceed.

- **Maintaining robot awareness of situation during human operations** The robot maintains awareness of human activities and their effects by monitoring those activities. We have implemented two approaches. In one approach, the robot monitors tele-operated actions using the sensor feedback events provided by the skill manager for autonomous execution. In the second approach, the robot uses alternative sensing techniques to monitor the action. For the servicing scenario, the alternative sensing technique is visual monitoring of human activities.

- **Reconfiguration of the robot at the resumption of autonomous operations** Tele-operated actions can change the state of the robot's subsystems and the state of the environment. When resuming autonomous operations, the robot must update the RAP memory with the most recent state information. To make this update tractable and efficient, the robot only checks states that are likely to have been affected by the actions performed while in semi-autonomous mode. Thus, if an arm-grasp RAP has been executed, the robot will update and verify the states of all the manipulators.

An example from the StationWorld problem domain is useful to demonstrate these key aspects. To begin a task – for example, the loading of a light – one first selects the level of autonomy. When the "Set LOA"

button is selected in the initializations menu, a pop-up window allows the choice of three levels of autonomy. The autonomous level is the normal mode of operation; the robot control is exercised by the appropriate RAPs enabling and disabling sets of skills to achieve a task. Both the RAPs agenda and the updates to RAPs memory are hidden from the user and are available only upon request.

In the tele-operate LOA, the RAPs methods guide the choice of which skills to use and when, but their actual execution is carried out by the human user. However, the RAPs enables essentially the same robot sensing events to follow the actions of the human. In some situations, the sensing may be for the same purpose but uses a different sensor, as when the electrical on-off toggle of a device is disabled and the human must use a manual switch to turn the device on. In this case, rather than use the electrical signal from the device, the robot may use a camera to watch the human throw the switch. In this mode, to give the human as much context as possible, both the agenda and the RAP memory updates are displayed to the human.

The selection of a semi-autonomous LOA – perhaps the most useful mode – causes the robot to query the human before each primitive action as to whether the human wishes the robot or the human to carry out the action. The mode selected is exercised as in either the autonomous or tele-operated LOAs, but only for that primitive action. The user also has the option at the query of reverting to either of the other two modes, thus providing a "robot takes over" or "human takes over" capability. Since this is a mode in which the human may want to only control certain steps of a task, only the RAPs agenda is made available to provide task context to the user.

Let's assume the situation is that the human wishes to control the gripper of the robot during the first half (the pick up) of the load task. In this case, the human selects the semi-autonomous mode. The robot selects RAPs to carry out the task and, since the left-arm is already at the light to be loaded, decomposes them down to the primitive level of the grasp operation. The user selects tele-operate at the pop-up and the RAPs system tells the user what to do and begins to monitor for the grasp event.

The user will use the low level interface to the simulator to carry out the grasp operation. At this point, however, let us assume that the user mistakenly commands the right arm to grasp. If the right arm is at fire extinguisher #2, it grasps that item. Since the RAPs continue to monitor for the left-arm grasp action, the robot waits for the human to execute the correct action. Seeing the mistake, the human now commands the left arm to grasp, RAPS notices the correct event, and continues with the task.

When RAPs decomposes the next phase of the task – putting the light in the ORU-pouch – it bottoms out at a primitive arm-move action and again asks the user which LOA he or she prefers. The user selects "robot takes over", which sets the LOA to autonomous, thus preventing any further queries from the system, and the robot successfully stows the light in the ORU pouch.

Now however, the right arm is holding an item. The robot knows about this change because a RAP method for using the human to carry out the grasp primitive has been added to the arm-grasp RAP definition. That method also contains eye-scan actions for the other arms that could have been commanded to grasp anything at that point in the task. These eye-scans will update the RAP memory as to the new location of the fire extinguisher. So the user now commands the arm-empty RAP and the robot deposits the fire extinguisher in the ORU-pouch, the preferred put down place for held items other than tools.

From our preliminary implementations we have identified several requirements for the sequencing layer of the architecture:

- The level of autonomy (LOA), which resides in the sequencing layer, must be adjustable both from the planning layer and by the user.
- The LOA must be adjustable during sequence execution.
- Additional sensing tasks and queries must be added to conditional sequences to determine inadvertent changes made to the world during tele-operate mode.
- Sequence failures must be recognized (i.e., cognizant failure) and special methods written to ask the human to intervene in these cases.
- Periodic RAP monitors must be used to update the RAP memory in order to notice changes to the world not expected by the specific traded control tasks.
- The sequencing layer must be able to accept updates in beliefs from the human at a change in level of autonomy. This capability is needed to inform the robots of changes that cannot be sensed.
- The sequencing layer must allow for selecting groups of tasks for autonomous execution, avoiding repetitive queries to the user at each step in the task. Ability to insert breakpoints in autonomous sequences where the sequencer requests if change is LOA is desired.

Skills for traded control

If skills are written correctly, semi-autonomous control can be easily implemented without major changes to the existing autonomous control skills. For example, a skill that flies the end effector of a manipulator while avoiding obstacles could have inputs coming from another autonomous skill that is generating set points or from a human that is flying the end effector. Independent of where the inputs come from, the skill just does its job.

The skill level of the architecture also easily supports traded control in the way that skills are partitioned into two classes: blocks and events. Blocks are skills that accomplish actions in the world. Events are skills that monitor the world through the robot's sensors and report changes in state back to the sequencing layer. Because control and monitoring are separate it is possible for the monitoring to continue even when the robot is under human control. This is a key contribution of our architecture to traded control.

User interfaces for traded control

An important capability for traded control is effective human-machine communication. The control software must be designed to make required information available as well as communicate this information effectively to a human. The human and machine must (1) exchange information on the machine's status, goals, beliefs, and intentions, (2) coordinate during joint/shared tasks, and (3) update world views at task hand over. This exchange includes both user-initiated information queries and user-volunteered information updates. These information updates can also be requested by the robot. The user can (1) change what the robot "intends" by altering goals, resource availability or allocation, or constraints, or (2) change what the robot "believes" by altering its perceived situation. Such updates help establish a common understanding of events by the human and intelligent machine (Roth, Malin, & Shreckenghost 1996) and are particularly useful at transitions between manual and autonomous task execution during traded control.

Key issues in designing for information exchange is assisting the human in knowing what to tell the robot and how to effectively communicate this information. This requires providing the human a view of what the robot currently understands (the current state of robot memory) and knowledge about what the robot is "interested" in or "needs to know." For example, verifying that the success clauses of planned manual activities hold true may be used to guide information queries at the transition to autonomy. An activity history/log that characterizes what robot observed about manual activities can assist in identifying and resolving inconsistencies, errors, and omission in RAPs memory.

Conclusions

We have just begun implementing traded control on our robots. We have a simulated robot domain and a real flight project. To date, all of our work has concentrated on the middle layer of our architecture, but we expect to begin addressing traded control at the planning level soon. At this point we expect to begin researching more complex issues related to mixed initiative interaction including having the planner decide when control shifts from robot to human, adjusting the plan as a robot is taken off of a task and coping with unexpected human intrusion. We see the domain of human/robot teams as a rich problem space for mixed initiative efforts and we see our autonomous control architecture as a framework for successfully implemented such human/robot teams.

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).

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.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS-96).*

Ferguson, G. 1995. *Knowledge representation and reasoning for mixed-initiative planning.* Ph.D. Dissertation, University of Rochester.

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

Myers, K. L. 1996. Advisable planning systems. In Tate, A., ed., *Advanced planning technology.* Menlo Park, CA: AAAI Press.

Roth, E.; Malin, J.; and Shreckenghost, D. 1996. Interfaces to intelligent systems. In Helander, M.; Landauer, T.; and Prabhu, P., eds., *The Handbook of HCI 2nd Edition.* Amsterdam: North Holland.