System Design for End-User Robots: sharing work amongst research programs

Marc G. Slack
The MITRE Washington AI Center, Mail Stop z401
7525 Colshire Drive, McLean, Virginia 22102-3481
slack@starbase.mitre.org

Abstract

In the last 5 years or so, new perception and control techniques have been developed and demonstrated which allow a to robot perform tasks in natural environments. Unfortunately, most efforts have resulted in one-of-a-kind robots, limited to research laboratories or carefully circumscribed venues. A user who would like to apply robotic technology to a particular real-world problem usually starts from scratch. The problem is that the robotic technology (i.e., the hardware and software) which might apply to the user's domain exists in a diverse array of formats and configurations. For intelligent robotics to become a practical reality, an effort to standardize some aspects of robotic technology must be made. This paper proposes that this technology be thought of in ways which are similar to the standardization of personal computer technology. I envision a "plug and play" system in which a person can buy robotic hardware components from one vendor and intelligent application software from another. A final system would then be constructed by assembling "snap together" hardware and installing the software, without requiring that the user understand the inner workings of the system. This work presents some system requirements (hardware and software) and one implemented design for end-user robots. We conclude that such technology is possible today and necessary for the sharing of ideas as well as the rapid advancement of the field of intelligent robotics.

Robot Capabilities

Robotics is often thought of as the black science of creating info-mechanical systems. The state of the art to date has resulted in divers paradigms for creating autonomous robots [Fikes *et al.*, 1972, Brooks, 1986, Kaelbling, 1987, Georgeff *et al.*, 1987, Schoppers, 1987, Payton, 1988, Connell, 1992]. As the science has developed, we as a community are beginning to construct agents which perform tasks which are useful beyond the laboratory. However, if we are ever going to reach a user community[1] with our technology we need to take some pointers from the evolution of the personal computer market and use these pointers to step into the future.

Buying a Robot

If one imagines pulling a new robot out of a box, and throwing a power switch, one can envision some basic skills with which the robot should come equipped. Assuming the example of a home user interested in the robot which will clean his floor, we expect the robot to navigate safely around the house, to have a basic mission sensor such as a barcode reader, a basic mission actuator such as a vacuum, and a user friendly interface. The robot will also its own set of self-monitoring routines which go into effect upon being powered up, so that the robot will "know" when to recharge or request maintenance. Assuming these capabilities, the user will guide the robot in a semi-autonomous mapping of his home, during which the areas to be cleaned will be given names, and a daily routine dictated. Given nothing else, the robot will remain docked at its battery charger, its clock ticking, waiting for the appropriate time to carry out its next cleaning tasks.

Adding Capability

Now assume our home user wants the robot to perform sentry duties as well as vacuum the floor. Ideally the user will purchase a "guard" sensor package which includes motion and heat sensors. In addition, the package will come with a set of software skills which allows the robot to use the new sensors (e.g., motion tracking or heat segmentation). The package will also

[1]Funding for this field will eventually become driven more by application than philosophical interest. This is not to say that there is nothing of philosophical interest in constructing systems with specific applications or application areas in mind, only that research sponsors are increasingly more interested in solving specific problems.

include a mission disk which which will support the integration needed to control the base and the new sensors for its job as a guard robot. Such disks will be available for three or four basic variety of end-user robot platforms. The software will be added to the mission and skill libraries of the robot, augmenting its previous capabilities. The new sensors will plug into the peripheral bus of the robot chassis. Once the package is installed, the user will again designate to the robot through the user interface which areas of the floor map need to be patrolled and at what times. The robot will then resolve any conflicts while consulting with the user and generate a schedule of cleaning and patrolling to follow.

Research

Consider the robotics researcher's situation. For a researcher concerned about the issues of intelligent agents, there are many potential levels of interest available. Many of the issues requiring research have nothing to do with the low-level control of the robot. For such researchers a robot which comes equipped with some primitive capabilities for their experimental domain would greatly facilitate the ability to focus the research. For example, one can imagine a robot which comes with some basic capabilities for operation in your basic office environment (e.g., `follow-hall`, `enter-doorway`, `look-for-office-x`, `wait-for-elevator`, etc.). These capabilities have been demonstrated and shown to be within the ability of the state of the art. Such a robot would allow researchers interested in issues such as map-learning to begin working on their research issue immediately upon arrival of their robot. All that they would require to begin their research is knowledge of the software environment with which their development must interface.

A Standardized Interface

There have been a number of attempts at standardizing the
interface to robotic systems [Mazer and et. al., 1991, Miller and Lennox, 1990]. However these have been directed mainly at teleoperation or assembly operations. We have chosen to take an approach which reflects some of the current efforts in robotic intelligence to integrate both Artificial Intelligence planning and the reactive approach to robot construction. To this end, there have been a number of systems which have to different degrees described, attempted, accomplished this integration [Firby, 1989, Simmons, 1990, Kaelbling and Rosenschein, 1990,
Connell, 1992, Gat, 1991]. This work standardizes this integration between planning and reaction while attempting to capture and reuse much of the low-level code that people have written for controlling robotic agents [Slack, 1992, Elsaesser and Slack, 1994].

Requirements

There are a number of considerations that must be taken into account when defining a generalized architecture for system development.

1. Semantic Abstraction. Knowledge acquisition and reasoning about that knowledge can take many different forms. There is the process of acquiring the information from the sensors and the process of interpreting that information within the context of the situation in order to make sense of the information. Action as well has different levels of abstraction. Consider the act of going to college as compared to the act of maneuvering through a hallway. An architecture which captures both planned actions as well as reactive interaction with the world would benefit from the ability to have different syntactic mechanisms for different levels of problem abstraction.
2. Reuse. The robotics community is full of code developed for controlling the actions of robots (e.g., inverse kinematics for Puma class robots, occupancy grids for mapping, navigation algorithms for real-time control of mobile robots, etc.). In addition, there are always new sensors and control algorithms being developed. An architecture which attempts to standardize the interface between the developer/user and the robot must be able to capture and easily build upon this body of work.
3. Hardware Portability. In order for an architecture to be general it must be flexible in order to mate with the different and diverse hardware architectures of robotic systems.
4. Extendibility. The real test of a general approach is the ability to extend the system as new developments are made in the field of robotics. This includes the ability to add new hardware and control software as well as the ability to adapt to hardware changes.

An Implementation

We have constructed an architecture to handle this process of mediating between AI Planning and reactive control. In addition, we have developed a number of tools to facilitate the development within this paradigm. The architecture's structure is depicted in Figure 1. While this particular diagram or ones like it have appeared in many references [Firby, 1989, Connell, 1992, Gat, 1991] we have attempted to provide a robust implementation of the concepts embodied in this particular structure. Each of the layers runs asynchronously with respect others communicating

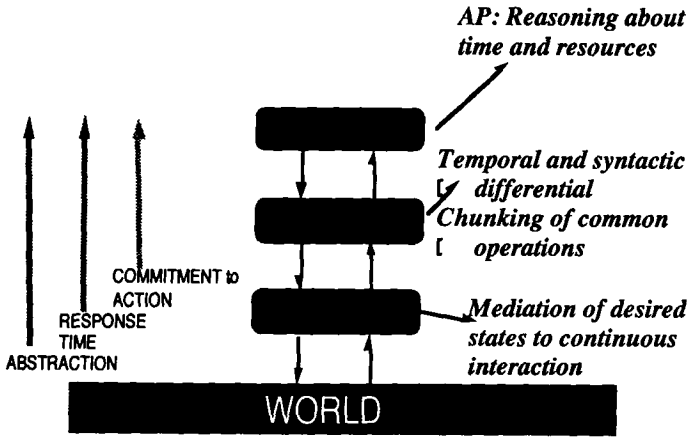

Figure 1: Intelligent Agent Architecture

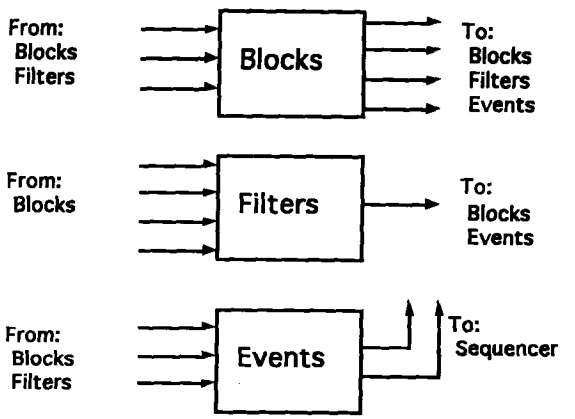

Figure 2: Computational elements of a Skill Network

with each other through events. Below is a thumbnail sketch of how each layer functions[2].

The Reactive Layer Is responsible for communicating with the robot's hardware and providing the immediate level of interaction with the environment. This layer is implemented as a dynamically configurable network of asynchronous computational skills. There are three types of skills used to form this layer of the architecture: blocks, events, and filters (see Figure 2). Blocks accept information from other blocks and filters performing a transformation over their inputs and providing their outputs to other skills. Filters are used to collect particular types of inputs from multiple blocks and perform mediation over their values and providing their outputs to other skills [3]. The event skill type is used to communicate information up to the sequencing layer of the architecture. Events accept inputs from blocks and filters to determine if a particular state in the world has been accomplished. If it has an asynchronous event is sent to the sequencing layer indicating that something of interest has occurred.

Implementation of the skills in this layer is accomplished using a development environment which packages the computational transform (written in your favorite language) into a canonical form. This allows developers to focus on the desired computation or interface of the system and ignore the hows associated with the inter-skill communications as well as the communications between layers of the architecture.

The Sequencing Layer Is implemented in a modified version of the Reactive Action Package (RAP) system [Firby, 1989]. The modifications made to the system allow it to execute tasks in parallel. The sequencer interacts with the Reactive layer by dynamically activating and deactivating skills based on the task and situation at hand, in essence, dynamically selecting the reactive agent which is necessary for the task at hand. In order for the sequencer to remain synchronized with the world it has special constructs which allow it to query state information from the reactive layer as well as block task execution until some event in the world occurs. Both of these mechanisms communicate with the reactive layer by activating the event skill type. These constructs allow the RAP system to perform method selection based upon the currently perceived situation by querying the reactive system and to determine task success or failure when messages are returned by the reactive system. The location of the skills is immaterial to the sequencer thus making it transparent whether the reactive portion of the architecture is implemented on a single processor or on multiple processors.

The Planning Layer The sequencing layer embodies a collection of tasks that the robot can routinely perform. There is however, no ability to project the consequences of current actions in the future in order to determine the impact of a proposed action or to reason about current resource utilization on future task requirements. These are the problems at which traditional AI planning has been directed. The planning layer of the architecture is dealing with a greatly simplified view of the world. The sequencing and reactive layers of the architecture allow the planner to reason at the grain size of the RAP library. This significantly reduces the planning space that the planner is concerned about making it possible to reason about and act on plans of moderate complexity in real-time. The planning system that we have chosen to use in this system is the AP (Adversarial Planner) system. It is a fairly typical least commitment planner with the ability to perform execution monitoring and least

[2]There is a forthcoming paper which details this architecture

[3]This is similar to the channel concept used in Gat's the Alpha programming language

impact replanning if the plan is determined to have a flaw. The planner is connected to the world through the sequencing system. It dynamically sends RAP tasks to the sequencer for action and monitors their progress by monitoring the memory state of the sequencer. In this way the planner can dynamically add and delete tasks from the RAP system in order to accomplish its high level objectives.

Current State of the Architecture

Today, we have a robust and tested implementation of the reactive and sequencing layers of the architecture. They are up and running on two separate Macintosh-based machines controlling our research platform "Uncle Bob".[4] We have also transitioned this architecture and development environment to a UNIX-based robot Mortimer at the NASA Johnson Space Center. The planning system is implemented and we are in the process of working out the final details of integration between the planner and the sequencer.

We have begin testing our architecture on a number of problems. Questions we plan to address include the following:

1. What activity is appropriate for planning or sequencing layers?
2. For which domains is reactivity sufficient?
3. Are there domains where sequencing is sufficient?
4. In which domains is deliberation highly useful?
5. To what measure is the architecture beneficial over more ad-hoc approaches?

Conclusions

Working with this system we have observed that once a robust set of skills have been created for the robot's domain of operation it is possible to extend the task set within the domain without making modifications to the skill set. This is a large win for robotics as it allows developers extend the robot's capabilities by symbolic manipulation of skills. We believe that further development will result in the ability to perform development strictly at the planning layer of the architecture, venturing to the lower layers only when the task domain is being extended.

If such an environment is viewed in its ideal sense we can easily imagine a system where the results of different research and development programs can be combined in using a "plug and play" approach. I do not believe that this is the answer to all research and integration but a step in a direction which will allow the community to benefit from synergistic activities. As a community we rebel against any standardization of technology as it "restricts our intellectual freedom". I argue that without any standard to work towards, that we will continue to reinvent the wheel at each turn and that the delta gain from each new robotic project will remain small as there is little opportunity to stand on the shoulders of others.

[4]From the Terminator 2 Movie

Acknowledgments

I'd like to thank Pete Bonasso, Jim Firby and David Miller for the many interesting conversations which helped form the ideas contained herein.

References

R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

J. H. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1992.

C. Elsaesser and M. G. Slack. Integrating delibrative planning in a robot architecture. In *Appears in the proceedings of the 1994 AAIA/NASA conference on Intelligent Robots in Field, Factory, Service and Space*, June 1994.

R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 23:251–288, 1972.

R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds.* PhD thesis, Yale University Department of Computer Science, January 1989. see Technical Report 672.

E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.* PhD thesis, Virginia Polytechnic Institute Department of Computer Science, April 1991.

M. P. Georgeff, A. L. Lansky, and M. J. Schoppers. Reasoning and planning in dynamic domains: An experiment with a mobile robot. In *Proceedings of the NASA Conference on Space Telerobotics*, pages 27–39, January 1987. JPL Publication 87-13, Volume I.

L. Kaelbling and S. Rosenschein. Action and planning in embeded agents. *Robotics and Autonomous Systems*, 6:35–48, 1990.

L. P. Kaelbling. An architecture for intelligent reactive systems. In *Proceedings of the Workshop on Planning and Reasoning About Action*, pages 395–410. Morgan Kaufman, 1987.

E. Mazer and et. al. Act: a robot programming environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1427–1432, April 1991.

D. J. Miller and R. C. Lennox. An object-oriented environment for robot system architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 352–361, 1990.

D. W. Payton. Internalized plans: A representation for action resources. In *Proceedings of the Workshop on Representation and Learning in an Autonomous Agent*, November 1988.

M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of American Association of Artificial Intelligence Conference*. Morgan-Kaufman, 1987.

R. Simmons. An architecture for coordinating planning, sensing, and action. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292–297, San Diego, CA, November 1990.

M. G. Slack. Sequencing formally defined reactions for robotic activity: Integrating RAPS and GAPPS. In *Proceedings of the SPIE Conference on Sensor Fusion*, November 1992.