# Adaptive Processes and Control

**Oliver G. Selfridge**
**Media Lab, MIT**
**Cambridge 02138, USA**
and
**Wally Feurzeig**
**BBN Technologies**
**Cambridge 02138, USA**

## ABSTRACT:

> "... software technology has promulgated a series of failed promises.... In fact...software bugs are pervasive, and there are no robust platforms underneath." [1]

The fundamentals of our approach are that every action undertaken by an agent or a subagent is an act of control; and hence is contained in an adaptive loop. Here we present a number of Elementary Adaptive Modules (EAMs), treating them as the basic building blocks of adaptive agent systems, with a discussion of their use, their control, and their behaviors under different conditions; we will also discuss the host of problems that we expect to run into.

We want to explore how to write software that can improve itself, and keep improving itself, far beyond mere simple programmed adaptation of pre-specified parameters. We want to test heuristically how to build and use hierarchical adaptive control structures in software. The basic elements are of course adaptive, and there are numerous expressions of those ideas. The essence of a purpose is choice and evaluation through some form of feedback; it is the marksman who has the purpose, not the trigger. A responsible agent is not to be designed and programmed in the usual way, for even if I think I know what I want now, I really want my agent to be what I am going to want it to be in the future, even as those wants change and evolve.

## INTRODUCTION

The fundamentals of our approach are that every action undertaken by an agent or a subagent is an act of control; and hence is contained in an adaptive loop. Here we present a number of Elementary Adaptive Modules (EAMs), treating them as the basic building blocks of adaptive agent systems, with a discussion of their use, their control, and their behaviors under different conditions; we will also discuss the host of problems that we expect to run into. We also present a taxonomy of control techniques using EAMs, including the highly specialized servomechanism as a very small class.

We want to explore how to write software that can improve itself far beyond mere simple programmed adaptation of a few parameters. We want to build hierarchical adaptive control structures in software. The basic elements are of course adaptive, and they all express purposes. The essence of a purpose is choice and evaluation through feedback: it is the marksman who has the purpose, not the trigger. A responsible agent cannot to be designed and programmed, for even if I think I know what I want now, I really want my agent to be what I am going to want in the future, even as that want changes and evolves.

A vital problem is the measurement of overall performance, at every level. Many other problems will emerge. What will perhaps be apparent is the astonishing amount of what we call blind control that is embedded in nearly all programming.

A chief essence of human intelligence is continuing learning or adapting. Those are what trigger growth and improvement: we do not (or should not) program our children with rigid rules, because they will have to learn and adapt anyhow. Machine learning has always been the most important aspect of AI; meaning the learning not just of particular skills or subroutines; but rather, of everything.

This suggests that AI software should be concerned more with being changeable—and all that that implies—than with being "correct". Machine learning is growing steadily, and in that field we include both neural nets and genetic programming, since both are occupied with dealing with changes. In people, though, learning and adapting take place at many levels simultaneously and continually, and we in Machine Learning must do so too. "Find a bug in a program, and fix it, and the program will work today. Show the program how to find and fix a bug, and the program will work forever."

> "We humans have purpose on the brain. We find it hard to look at anything without wondering what it is 'for,' what the motive for it is, or the purpose behind it. ... Show us almost any object or person, and it is hard for us to resist the 'Why' question—the 'What is it for?' question." [2]

It has long been understood in the fields of neurophysiology and psychology that our point of view here is the essence of human control. Nearly twenty years ago, Dick Pew wrote:

> "The human motor system should be viewed as a hierarchically distributed processing system. The hierarchical property has always been accepted,... [T]he various levels in the hierarchy operate more autonomously than previously assumed. " [3]

## THE NATURE OF CONTROL

Control is by definition an adaptive process; that is to say, to exercise control, one must make choices about actions (including doing nothing): one must do that action chosen, and then observe and assess the results, and then modify (or not) one's plans or future actions.

In our model, control has many levels. We start by differentiating the three kinds of features available to the components: actions, responses, and evaluations. Each of these can be described by either discrete or continuous variables. Also, each may be subject to interference of many kinds; in observables, for example, this is often termed noise. The environments within which the controls perform are also important, sometimes even prescriptive. We need to classify agent components according to these features and their characteristics

We will be concerned with control levels in information agents, not physical agents. However, even muscular or robotic control involves extensive information communication. Wiener points out, in *Cybernetics*, that vertebrate control of muscle is not performed directly by the motor neurons in the cerebral cortex; rather, the motor neuron changes the gain of the muscle-proprioceptor-spinal loop [4].

The levels of control in agent systems can range from setting the gain of a simple adaptive loop, all the way up to developing contingent operational plans in complex areas like logistics scheduling. The performance of agents at each level is the functional consequences of adaptive information processing at lower levels, as well the orders or instructions from above.

## CLASSICAL CONTROL THEORY

Adaptive control is not a new field. It traditionally began with James Watt in 1784 (see [5]), and the theory of it was mathematically analyzed by Clerk Maxwell in the middle 1800s [6]. Classical control theory is mostly concerned with a limited range of servomechanism-like processes, restricted by linearity and Gaussian noise. In many instances it is very powerful. But the applicability of such techniques is limited when it comes to learning, fine tuning, tackling new situations, and so on. As an example, the gain of a servo cannot be adjusted by means of a servo.

There has been some recent work on systems with embedded with learning and adaptive processes. For example, TACAIR-SOAR is an rule-based system that generates believable humanlike behavior for large-scale distributed military simulations. It has been used to explore how systems can acquire expert-level knowledge through learning. But that is often hard to do:

> "Some of the most advanced system for computer-generated agents used in military simulations are based on the Soar architecture (Laird et al., 1987). ... It provides a learning mechanism called chunking, but *this option is not activated in the current military simulations...*" [8, our stress]

A more general example can be found in the Optimal Control Modeling of human operators (e.g., [9]); here the model processes are linear and continuous, and they work only in environments susceptible to servo analysis.

In most work, the purposes of the agents are specific. They have to be explicitly programmed. Often the specification of purposes is narrow and limited. Yet any search process (e.g., in planning) is purposeful in our sense. Of economic agent systems in a market-based environment, Boutilier says: "Often overlooked...is the fact that an agent must plan not only for its domain-level objectives but also for its plan-suitable economic behavior." [10]

In the last half century the adaptive control field has blossomed beyond servos: neural nets (now termed connectionism), genetic programming, and other approaches have been seriously studied. They have been extensively applied to adaptive control problems in robotics, for example.

Connectionism and genetic programming both emphasize the roles of continuing change. Connectionism is perhaps the broader of the two; the source of the inspiration is rational indeed, but the implementation is less so.

What is missing in nearly all these approaches to modeling adaptive processes is a drive to discuss the structures of purpose and learning as a central theme. Biologists have been concerned with purposes for some time. In a recent and fairly comprehensive anthology of such work, "Nature's Purposes", it is argued that:

> "Because the account we give of the role of teleology in biology will have consequences for our conception of the nature of biology itself, the goal of accounting for 'Nature's purposes' is arguably the most important foundational issue in the philosophy of biology ... it is unsurprising [that there have been many] attempts to come to terms with teleology." [11]

Both connectionist and genetic programming methods are controlled by purposes and goals. But these approaches share a profound limitation. In both, the feedback for adaptive modifications is chiefly through a single information channel that derives from a single-level propagation approach to the overall goal, without intervening sub-goal structures.

At present there is considerable and widespread work on complex adaptive systems. A general characteristic in nearly all this work is that plans and behaviors are specified by built-in actions. Impasses and obstructions therefore have to be handled *ad hoc*, with pre-specified remedies. Those planning cells are non-adaptive. We plan, in contrast, to design subagents that will continually adapt to the changing world at every level, while doing what they are assigned.

## THE EAMS INDIVIDUALLY

First we deal with individual EAMs and their control parameters and behaviors.

**Control and the EAM: Blind Control.** In the every standard course on control theory, the concept of control implies a loop. However, for human experience, the loop aspect is often ignored, although it is nearly always present.

We suggest that the clearest instances of control that do not work as loops are to be found in computer software. We term such control "blind". Computer hardware these days is usually so reliable that blind control works—most of the time. When it doesn't do what the programmer wanted it to do , that is when the trouble starts, with systems' crashes and so on.

We do not argue that software uses only blind controls; far from it. For example, an IF statement usually requires some checking, and so may be an example of a real control. But software algorithms are set up without the universal checks and balances that we are referring to here.

**Types of EAMs.** Every EAM has one of two fundamentally different types of behavior: the first is passive, where the module makes direct changes in response to the behavior of the world outside of it; the second is active, where the model makes changes in how it is behaving towards the outside

world, and uses that world's responses to change its behavior.

**Passive EAMs**. The standard example is the servomechanism:

- A servo's action is to make additions to a continuous 1D scalar variable; by an amount D (linearly) proportional to the error E; D is then multiplied by the negative of the gain G. This means that E must be able to be both positive and negative.
- The overall behavior of E enables us to assess the performance of the EAM.
- The observations that enable computation of E are usually corrupted by an noise, so that E is some kind of recent average of observed deviations. And those deviations are with respect to a desired value of the variable, the *setpoint*.
- The additive noise is nearly always considered to be Gaussian with 0 mean.

In the passive EAMs we are considering, most of those details are not followed. The first obvious one is that the action need not be 1D, though it does have to be more or less continuous—that is, it does not merely take on a small number of discrete possibilities. Next, D does not have to be linearly proportional to E, though there is usually something equivalent to G, which sets the overall size of the correction. Finally, the noise does not have to be Gaussian.

Passive EAMs are not immediately applicable to discrete action ranges.

**Active EAMs**. Here the EAM makes changes experimentally, to see if (small) changes improve or harm the performance. An obvious way to do this is to make a change in the variable that is being controlled; if things get better, keep going; if worse, go the other way. We have termed such operation RT, for Run and Twiddle. We regard it as the archetype of active EAM systems. The RT procedure is akin to the determination of the gradient in mathematically well-behaved systems, but is far more general.

RT is an extraordinarily powerful EAM. It is often inefficient, but it is very flexible. There are many variations on its behavior. It reflects the fundamental practicum of adaptive (and learning) behavior—keep on doing what works.

**Features of the EAMs.** Any adaptive unit receives information, and does something, often on a continuing basis; the ways in which this is done are the features of an operating EAM. Our initial taxonomy of EAMs describes some of these:

- External action: this may be discrete or continuous; it may be applied sporadically or continually.
- External action: its size is a crucial factor. It might correspond to the *gain* of a servo, or to the *step size* of an RT module.
- External response to action: this may be discrete or continuous; it may occur temporarily or continually.
- The Evaluation Function (EF): this is a function of the external responses, and measures the fitness of the EAM. The EF of a servo EAM is called its *setpoint* and is a *control parameter*. The EF of an RT EAM, its *purpose*, is also a control parameter.

Features of the EF include Integration Time, the length of time over which we measure the responses. It is one of the *control parameters*. It will often not be constant. The particular balancing of the responses may be discrete or continuous, as noted above. .

Note that the EF is itself a high-level control parameter, and may be termed the *purpose* assigned to the EAM. That is a very general point . The EF will be the primary control parameter. The other parameters function are more in the nature of supporting controls.

Initial settings of the control parameters will be an interesting technical problem that we discuss later. There are of course other kinds of control parameters, including the following.

- For active EAMs, what kinds of steps or changes in the controlled variables should be tried? Notice that the steps can be undertaken in variable spaces of more than one dimension.
- How often should changes be made or assessed? This is especially relevant when the EAM is controlling a discrete variable.
- When and how often should the EAMs be exercised, and how should those intervals be assessed?

  There are often cautions that need to be applied and that are not obvious: if we are controlling a gain G of a servo near its best position, a fair-sized increase in it may easily lead to divergence and a consequent system crash.
- There are often limits to the possible values of parameters: a vehicle usually has a maximum possible (or safe) velocity; that parameter may be both inherent, as from available power, or imposed externally, e.g., by a speed limit.

**Timing: permanent and ephemeral EAMs** Some kinds of control are working most of the time: controlling the speed of a vehicle is an obvious example at the low end; expressing values, at the high end. But the detailed assessment of speed and its controls aren't used when the vehicle is at rest. Nevertheless, we can term that kind of control as *permanent*, in that we would plan to design it so as to be in place permanently.

There are, however, some agents of control that are exercised only momentarily: turning on the light with a binary switch is one of them—yes there is a loop there, for if the light doesn't go on in response to the switch, then something different must happen. We term these controls and their EAMs *ephemeral*. It is a real question of design whether to build these controls on the fly, so to speak, and then to dump them after their purposes have been fulfilled (or not!).

**Timing: other questions.** Many, if not most, of the adaptive controls in any complex structure are not doing positive adaptation at any particular time. Using the previous example, we would ideally not adapt the gain of a throttle control when the vehicle is parked. Similarly, the fitness functions of the overall system and of its substructures need to be evaluated during intervals in which they are performing, and not otherwise. This suggests that a control needs something like its own clock for assessment measuring.

**Difficulties in setting up the EAMs,** Here are some more or less independent problems that can arise in the real world, having to do with the environments that the EAMs may work in.

- The first question is how to determine what kind of EAM should be used to handle a particular control problem. It

seems obvious that control of a binary variable, like a light switch, needs a different kind of EAM from one running ailerons on airplane wings. Some of these choices are easy, others not .

- How do we set up the initial control parameters (at every level)?

## CONTROL OF THE EAMS

Agents and subagents and their control components are real-time adaptive and quasi-autonomous units driven by purpose structures. A key problem in the design and development of our adaptive agent design is that purposes, and their evaluation functions, must be expressed precisely and accurately, in order to support analysis and processing. The nature of the controls that must be exercised is shown schematically in Figure 2.

For a low-level component that may seem to be almost trivial: for example, to maintain an adequate local supply of some resource such as fuel, the amount of fuel on hand may serve as the scalar basis for the evaluation function. For a high-level component, it will be much harder, involving combining conflicting purposes, estimating the risks and future contingencies. We describe our approaches to that in the next section.

In the most simple or elementary adaptive modules, there are several important considerations, and several different forms of behavior; they all need to be constructed as essentially different structures; a hill-climber does not work or behave in the same way as a servomechanism, and yet we will need both types of modules in our agent-based systems. Some of the important considerations in choosing a basic set of modules are:

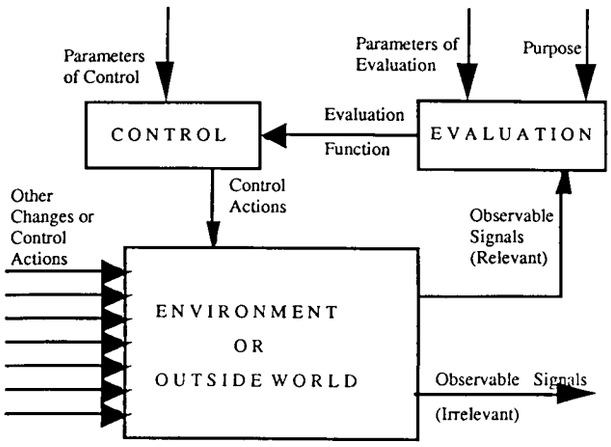


**Figure 1** **Essence of Control**

- How many fundamentally different structures will be needed, and with what controls?
- Is the responsibility of the agent to make discrete and irreversible decisions (like to fire a weapon), or to modify a variable (like the gain or setpoint of a lower level control)?
- Are the adaptive traces of the system merely the current values of the control parameters, or are there separate facilities for recording history—for case-based reasoning, for example.
- When any system of agents has been working for some time, they will have spawned new agents and subagents; how should these be incorporated in a library of structures, for further utilization? How should they be classifed?

Those considerations and features are not rules. Instead they suggest the order in which different structures are tested and evaluated, reflecting the working of one of the highest purposes, to search or adapt fast and effectively.

- Elementary hill climber: with continuous action and continuous evaluation function; we require neither that the evaluation function be a continuous function of the state, nor that it can be negative. We call this controller "RT".
- Elementary servo: with standard conditions, save that the noise need be neither linear nor Gaussian. We call this controller "servo".
- Discrete controller: in the analysis of plans, this will be of primary use, since most elements in them consist of discrete choices. We term this controller a "chooser".
- The Null Controller: at the beginning most of the controllable parameters are likely to be fixed (by the researcher), and it is important for the software that changes be allowed for *ab ovo*. The null controller will be a piece of software does nothing, but can enable later substitution of active controllers. We denote it by lambda, "$\Lambda$".

Every elementary adaptive controller will have the following parameters for controlling it:

- Action? The unit must be able to undertake some action, even if it is a null action. This parameter denotes that kind of action, each describable by a different number or character. actions are countable, and often even binary. Note that a unit with a continuous action can be used to provide a binary action, but not the other way around.
- Action Target? What unit or exterior performing parameter is the unit controlling? The kind of Action Target is denoted by T.
- Evaluation function? The unit must use observable inputs to compute the evaluation function that enables it to work out the next action(s). Example: have I won the game, or lost it? Some evaluation functions are continuous. The kind of Evaluation is denoted by E.
- Purpose, or setpoint, or goal? This parameter describes the process by which the evaluation function is used for control. Thus the purpose of playing a game of chess may be to win the game. The kind of Purpose is denoted by P.
- Gain? This parameter describes the nature of the changes on the actions to be taken. It has no significance, of course, when A is binary. The Gain is denoted by G, and may be a scalar or some other descriptor.
- Name or identification number? Every unit structure must have a name in order to specify the connections between them. The Name is denoted by N.

Thus the elementary units are each denoted by the set of descriptors {A , T , E , P , G, N}; note that only the elementary units have this few descriptors. Initially, we will set most parameters by hand, while making sure to include the capability of their control by the system.

The example of chess shows one of the traps in many theories of adaptation and learning. In simple games, like

tic-tac-toe, the evaluation function can be simply have I won or not? But in beginning chess, that is just not enough; fundamentally that is because winning or losing can provide at most one bit of information (and usually much less) per game. The use of subgoals, however, can provide up to and more than one bit of information per move. The subgoal, of course, must be relevant; however, if we have already derived a subgoal (e.g., from a simpler game), that may be much easier. Two examples: tic-tac-toe provides an easy environment in which to discover the idea of a double threat; and for children, playing with other children provides an easy environment for discovering the idea of the value of one's pieces—for who likes to have one's toy s taken by someone else? It is hard to imagine that that idea could be easily derived from the information in one bit per game of chess. And once there is the idea of pieces having values, it is a short conceptual leap to confirm the idea of relative values of different pieces in different situations.

## INITIAL SETTINGS

These are complicated and difficult problems. To illustrate:

We want an agent to control the speed of a vehicle by changing the (throttle) setting. That agent we decide should be a servo EAM with its associated control parameters. Initially, the setpoint V0 will be 20 m/sec, and it will not change.

We suppose the fixed operational period to be 100 ms. That is, every 0.1 seconds E ( = V - V0) is computed from continuing speedometer observations, DQ is computed and applied to the throttle setting Q

$$Q = Q + DQ\ (= E * (-G))$$

If the road is at all rough or if the speedometer is noisy, then E must be computed by some kind of average of recent speedometer and error settings. An initial setting might be for example

$$E(t) = 0.9 * E(t-1) + 0.1 * (V(t) - V0)$$

That choice of procedure and the parameters (0.9, 0.1) are control parameters to be tuned later; in the example below those numbers are replaced by R and (1 - R).

Notice also that Q may not be matched to the output velocity, depending for example irregularly on the slope of the terrain or the weather in the air; too, the throttle may have an unknown action of the actual speed of the vehicle.

## THE BEGINNINGS OF STRUCTURES

**Vehicle control** For our first example we consider an agent responsible for the control of a vehicle. Part of the overall agent structure is shown in figure 2.

Each of the boxes in that figure is quite complicated. A first stab at one of them—inside the dashed lines—is shown in figure 3.

**Throttle control** It should be remembered that figure 2 is ideal in many engineering respects. In fact, any action taken, like a change in throttle setting, takes time to have effect. In fact the mathematical theory of servos is very complicated, and we will not delve into that at this point; for our interests are more in adapting to the complications of real life and realistic controls. We will, however, discuss the various complexities of real-life control systems later.

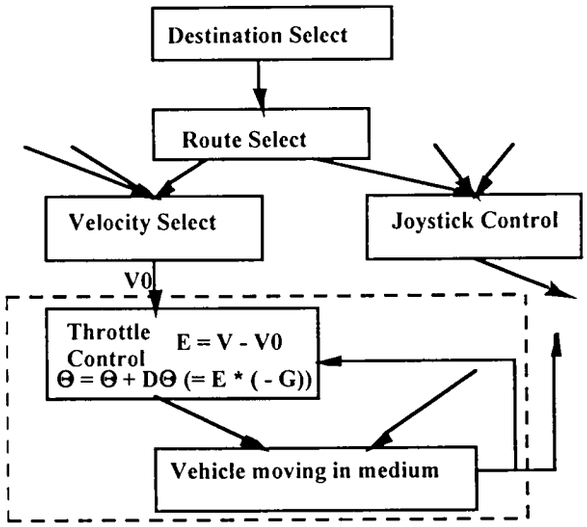


**Figure 2** **Overall structure**

But there are some assumptions worth noting. We assume that the speedometer is accurate; or if not accurate, then with an error that averages out to 0; or else the performance will be skewed. We assume that the gain is set at a reasonable level: if G is too high then the velocity may oscillate out of control; if too low, then convergence will be too slow. Other assumptions will be obvious to those who have taken a course in control theory.

In general, furthermore, in what we are proposing, the gain G and ratio R are also set by control loops; but such control loops cannot be servos; for one thing, the fitness function, or how well the system is doing as a whole, is just some departure from a (possible changing) ideal—and such a departure must always be greater than zero.

In real life systems, G is usually established by the system designer, both from analysis and from experience. In ordinary systems, like cruise control, that value is never changed. In what we are proposing the need for learning new tasks and for handling differing environments will deny us that luxury.

A straightforward control for the gain G uses the RT EAM; as shown in figure 3. However, we must allow for certain other considerations (which may apply very widely throughout purpose structures):

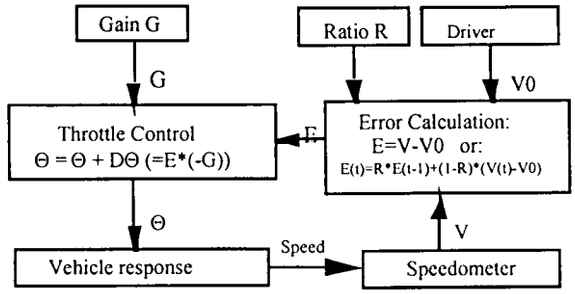


**Figure 3** **A substructure from figure 2**

It would be easy to go hog wild about adding control structures to control structures: for example, in figure 3, it will no doubt occur to the reader that the selection of a good step size DG may require its own diagram very like that of figure 3.

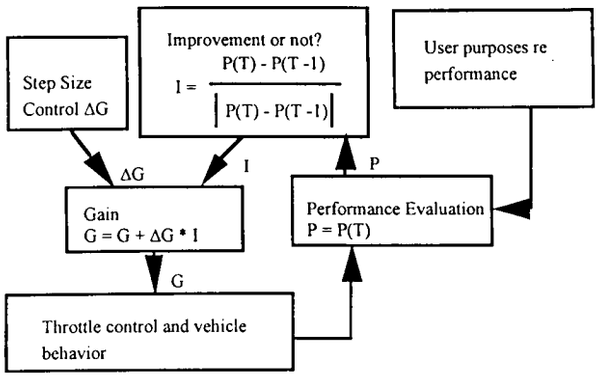


**Figure 4** **Structure for discrete control**

Moreover, every adaptive control needs some attention while it is operating; that carries some cost. For people, if some performance is good enough, then many of those controls are often treated as ephemeral.

**Steering.** It is clear that while the throttle control may in some circumstances be treated as an ephemeral control;, that is not possible with steering, which always requires constant attention.

## DIFFICULTIES IN RUNNING STRUCTURES

The most obvious difficulty in running structures is simple: conflict. How does an agent handle two opposed subgoals?

- Purpose structures at some level above the lowest often have the responsibility of trying to satisfy several purposes at once. Two differing goals may not be directly in opposition to each other, but they areclearly competing for resources.
- Different purpose structures often have different *styles* of behavior:
- A crucial setting for an RT EAM is the step size; for a servo it is the gain G; for a discrete control, it is how often we test the other possibilities.
- Relative priorities of different purposes will change as the environment changes.

## BUILDING NEW AGENTS

On the surface we may be deceived by the following argument: if we need a new agent for some task, copy some agent doing a similar task and then let it adapt to the new one. People can no doubt do this fairly well, since it has been done in programming for decades. But our goal is to enable an agent itself to build other agents.

Some easy cases are clear. If we have a competent agent that is piloting a vehicle, then a similar vehicle will call for first copying that pilot agent. Note that we use the term *similar* instead of *identical* because two vehicles cannot be identical intrinsically; for one thing they have different identification numbers. But a ground-based vehicle 'pilot' (or driver) should probably not be used as a model for an agent pilot in an air-based vehicle.

Note further that that argument does not rule out the copying of certain subagents to apply. For example, the throttle control discussed above will probably be close enough to extrapolate from ground-based vehicles to airborne ones—or, for that matter, to marine ones!

## CONCLUSIONS

The assembly of an agent from the purpose structures is a complicated task indeed—just as is writing a program for that agent to do the same things. But the former agent will inherently have greater and more powerful capabilities for changes and improvements,.

We are trying to bring out into the open a large number of considerations and assumptions that programmers usually write into programs without regarding them as controls at all. Hidden assumptions are often a bane to understanding. For example, machine learning software usually makes assumptions about what is to be learned and how it ought to be learned without the programmers' awareness in writing it.

We have also concluded that the actual design of true adaptive control along the lines of what we are proposing leads to enormous nets of control loops. Many of these loops function adaptively only sporadically, of course, and the control and utilization of that fact has not been examined in any general way. Of course, ordinary programming techniques use control loops all the time, although the programmers do not usually call them that.

## REFERENCES


[1] Brown, W.J., Malveau R.C., McCormick, H.W.III, and Mowbray, T.J. **AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.** John Wiley, New York. 1998, p 132.

[2] Dawkins, Richard. **River out of Eden: A Darwinian View of Life.** BasicBooks. 1995., p.98.

[3] Pew, R.W. "A Distributed processing view of human motor control". In W. Prinz and A. F. Sanders, (Eds) **Cognition and Motor Processes.** Sprintger-Verlag. 1984, pp. 19-27.

[4] Wiener, N., **Cybernetics**, Wiley.,1948.

[5] Willems, J. C. **The analysis of feedback systems.** Cambridge, MA. MIT Press, 1971.

[6] Maxwell, J.Clerk, "On Governors", **Proc. Royal Soc., 16**, London, 1868, pp. 270-283.

[7] Laird, J.E., Newell, A., & Rosenbloom, P.S., "Soar: 'An architecture for general intelligence' ", **Artificial Intelligence, 33**, pp. 1-64.

[8] See [3] above, pp.19-27.

[9] Pew, R.W. & Mavor, A.S. **Modeling Human and Organizational Behavior.** Commission on Behavioral and Social Sciences and Education, National Research Council. National Academy Press. 1998, p. 37.

[10] Boutilier, Craig. "Multiagent System"*s*. **AI Magazine, 20**, 4, Winter 99, p. 42.

[11] Allen, C., Bekoff, M., & Lauder, G., eds. **Nature's purposes; analyses of function and design in biology**. MIT Press, 1998, p. 2.