

Opportunistic Control of Action in Intelligent Agents

Barbara Hayes-Roth
Knowledge Systems Laboratory
Stanford University
Palo Alto, CA 94304



Abstract for Correspondence

An agent should adopt different control modes in different situations. Depending on the predictability of its environment and the constraint imposed by its goals, the agent should modulate its sensitivity to run-time events and its commitment to specific actions. We propose an opportunistic control model that supports this flexibility.

Abstract

Given its multiple goals, limited resources, and dynamic environment, an intelligent agent must decide which of many possible actions to execute at each point in time. Planning and reactive models embody two different *modes of control*. By contrast, we characterize a two-dimensional space of control modes, each of which maximizes the quality of run-time behavior in the corresponding region of a two-dimensional space of control situations. The situation space is defined by dimensions representing the predictability of the agent's task environment and the constraint imposed by its goals. The space of control modes is defined by corresponding dimensions representing the agent's sensitivity to run-time events and its commitment to specific actions prepared in advance. The corners of the mode space are anchored by extreme modes that an agent rarely uses: *open-loop plan execution*, *goal-specific reactivity*, *dead reckoning*, and *reflex* modes. The large interior is occupied by a space of more practical *strategic* control modes that modulate the balance of sensitivity and commitment for situations embodying different combinations of predictability and constraint. To support behavior throughout the space of control modes, we propose an *opportunistic control model*: run-time conditions trigger a subset of possible actions, strategic plans constrain intended actions, and the match between possible actions and strategic plans controls action execution. The model is operationalized as an agent architecture and an associated representation language for actions and plans. We show how the model allows an agent to move continuously throughout the space of control modes and report preliminary empirical results.

1. The Problem

An intelligent agent has multiple goals, limited resources, and a dynamic real-time task environment. Because it has many predictable and unpredictable opportunities to act, the agent must “control” its actions--it must decide which of many possible actions to execute at each opportunity over a period of time. In a given situation, the agent’s control decisions determine whether it achieves its goals, what resources it consumes, and what side effects it produces. More generally, the agent’s approach to control determines the range of situations it can handle, its flexibility in responding to unanticipated conditions, and its coherence and comprehensibility to observers. How should an agent control its actions in order to achieve a high global utility?

Under the well-known “planning model” [11, 34, 40, 41, 45, 46, 48], an agent plans its actions ahead of time. Given its goals, potential actions, and current and expected states of the environment, the agent constructs a logical sequence of specific executable actions, perhaps with some conditionality, whose execution will achieve the goals. Then it executes the planned actions. To guard against failures, the agent monitors plan execution and, if necessary, replans. Although there are different methods for constructing plans [2, 16, 38], all operationalize plans as executable programs of action. For present purposes, they are equivalent. The planning model has several strengths. An agent can construct plans that have desirable global properties regarding goals, resources, and side effects, as well as coherence and comprehensibility. While following a plan, it can conserve resources that would otherwise be spent perceiving the environment [32, 47], deciding among alternative actions, or instantiating specific executable actions; as a result it can execute planned actions promptly at run time. The agent can make different plans for different situations and change plans when necessary. Weaknesses of the planning model include the high computational cost of planning [7] and the inflexibility of planned behavior under unanticipated run-time conditions [1, 39]. In the worst case, an agent encounters a critical condition and undertakes a computationally expensive replanning task as time runs out.

Emphasizing the planning model’s weaknesses and ignoring its strengths, some researchers firmly reject it in favor of the “reactive model” [1, 5, 33, 35, 39]. Instead of planning, an agent reacts to environmental events as they unfold. Given its goals, potential actions, and relevant possible states of the environment, the agent (or more often, a programmer or knowledge engineer working with an automatic programmer) constructs a set of goal-specific perception-action rules and stores them in a computationally efficient

form. On each iteration of a perceive-act cycle, the agent perceives the current environmental state, triggers the associated rule, and performs its action. Thus, the agent can respond flexibly to many run-time conditions, in some implementations with bounded cycle time. Weaknesses include the infeasibility of anticipating, distinguishing, and storing all run-time contingencies [15], precipitous failure when unanticipated conditions do occur, the infeasibility of perceiving complete relevant environmental state in bounded time, and the uncertain global effectiveness, efficiency, and coherence of locally determined behavior. In the worst case, an agent encounters a critical condition and thrashes about executing locally appropriate, but globally uncoordinated actions, as time runs out.

Ironically, the debate between proponents of planning and reactive models reprises a similar debate among psychologists--but in the opposite order. During the first part of this century, psychology was dominated by the "stimulus-response (S-R) model" [31, 37, 42]. Like the reactive model, it excludes internal representations and cognitive processes as mediators of goal-directed behavior. Instead, it models behavior as a sequence of immediate reactions to sensory information. Despite evidence that such reactions occur, most psychologists eventually concluded that the S-R model alone could not account for all of the experimental data, let alone all of human behavior [3, 10, 14, 23, 24]--or even all of animal behavior [25, 43]. Proponents of the "cognitive model" [6, 29, 30] firmly rejected the S-R model:

(1) All psychologists have recognized [the selective central] factor in behavior. It undoubtedly exists. (2) Recognizing it is really a denial that behavior is only a series of responses to environmental stimulation. [23]

... behavior in general, and human behavior in particular, is not a chain of conditioned reflexes. ... the effect an event will have upon behavior depends on how the event is represented in the organism's picture of itself and its universe. ... any correlations between stimulation and response must be mediated by an organized representation of the environment, a system of concepts and relations within which the organism is located. A human being ... builds up an internal representation. [26]

To make sense of this recurring debate, we must distinguish between *control modes* and *control mechanisms*. Control modes are behaviors that differentially emphasize two orthogonal qualities: sensitivity to run-time events and commitment to specific actions. Control mechanisms are data structures and processes used by an agent to realize particular control modes. We argue that an intelligent agent should be able to operate in different control modes in different situations--and, therefore, needs a control mechanism that can support different modes and smooth transitions among them.

Figure 1 characterizes theoretical spaces of control situations and control modes, superimposed on one another.

First consider the theoretical space of control situations. The left vertical dimension represents the uncertainty of the task environment, including both exogenously produced events and events influenced by the agent's own actions. At one extreme, the environment is minimally uncertain, allows only predictable relevant events to occur, and demands minimal run-time monitoring. At the other extreme, the environment is maximally uncertain, allows many unpredictable relevant events to occur, and demands maximum run-time monitoring. The bottom horizontal dimension represents the degree of constraint imposed by the agent's goals. At one extreme, the goals are minimally constraining, allow many alternative successful courses of action, and therefore demand minimum commitment to specific actions. At the other extreme, the goals are maximally constraining, allow only a single successful course of action, and therefore demand maximum commitment to specific actions. We assume continuous gradation of both dimensions of the situation space.

Now consider the theoretical space of control modes. The right vertical dimension represents sensitivity to run-time events. At one extreme, the agent is minimally sensitive and executes prepared actions open-loop. At the other extreme, the agent is maximally sensitive and conditionalizes every action on immediate environmental conditions. The upper horizontal dimension represents the agent's commitment to specific actions. At one extreme, the agent's commitment is minimal and it is willing to perform any of a very large class of actions during the run-time interval. At the other extreme, the agent's commitment is maximal and it is willing to perform exactly one action at each point during the run-time interval. We assume continuous gradation of both dimensions of the mode space.

PLEASE INSERT FIGURE 1 HERE

The two spaces are superimposed to indicate that control modes in particular regions of the mode space produce the best performance in corresponding regions of the situation space. We assume that adequate monitoring and preparation resources are available and that, other things equal, an agent prefers to spend resources on preparation rather than on monitoring in order to achieve the best possible run-time performance. Alternatively, if we assume variations in availability or cost of monitoring and preparation resources, the superimposed spaces show how run-time performance may be degraded in order to conserve those resources. We use the domain of skiing to illustrate the proposed mapping between

control situations and modes for the four theoretical extreme modes: *open-loop plan execution*, *goal-specific reaction*, *dead reckoning*, and *reflex*. However, because intelligent agents rarely encounter the extreme situations represented along the borders of the situation space and rarely operate in the associated extreme control modes, our skiing examples merely approximate these extreme modes and, therefore, are positioned well inside the borders of the space. We use the term *strategic* mode to refer to the large interior region of more practical control modes that modulate the balance of sensitivity and commitment for situations embodying intermediate degrees of uncertainty and constraint.

When an agent has maximally constraining goals to achieve in a minimally uncertain environment, *open-loop plan execution* produces the best run-time performance. The agent can prepare a detailed, globally coordinated plan in advance and execute it reliably and efficiently with minimal run-time monitoring. For example, consider an Olympic downhill skier:

In the days before the race, after walking the course and skiing two time trials, you will have run the race in your head dozens if not hundreds of times. Memorizing the course is crucial since at speeds averaging 70 miles an hour there is little opportunity for adjustment and you need every precious instant to react to the unknown and unforeseeable, primarily imperfections in the skiing surface. [4]

The skier achieves maximum race speed by severely restricting run-time sensitivity to the environment and conditionality of actions.

When an agent has maximally constraining goals to achieve in a maximally uncertain environment, *goal-directed reaction* produces the best run-time performance. The agent can commit in advance to perform specific actions under specific conditions, but must determine which of these conditions hold and, therefore, which of these actions to execute based on run-time monitoring. For example, consider the same Olympic skier participating in an impromptu race on an unfamiliar trail. He or she prepares specific racing actions for adjusting speed and course based on relevant gross features of the environment, as well as subtler adjustments to surface imperfections. At run time, the skier focuses on monitoring just the relevant features of the environment in order to decide which of its small set of race-relevant actions to execute and when to execute them. Presumably the skier will not achieve Olympic speeds because of the time required for additional run-time monitoring and

adjustment of actions and because of possible weaknesses in the global properties of the action sequence.

Notice that in both planning and reaction modes--which correspond to the planning and reactive models in the literature--an agent commits to specific executable actions in advance in order to achieve highly constraining goals. Planning mode exploits predictability in the environment to eliminate most monitoring actions, to restrict action to a single globally coordinated sequence, and thus to streamline run-time performance. Without this predictability, reaction mode prepares a larger number of actions for a larger number of contingencies. In intermediate modes along the planning-reaction border, the agent modulates the amount of run-time monitoring it performs and the associated conditionality of its actions. In all modes on this border, however, the agent expends substantial computational resources in advance to determine the set of specific conditions it will monitor and the set of specific actions it is willing to perform. The agent is maximally committed and unlikely to respond to an unanticipated event or to perform an unanticipated action.

But what about situations in which the agent has more general goals and does not need to constrain its actions so severely? What about situations in which the agent wishes to behave more flexibly at run time or to conserve resources spent on preparation?

When an agent has minimally constraining goals to achieve in a minimally uncertain environment, *dead reckoning* mode produces the best run-time performance at less preparation cost than planning mode. The agent can make a very general plan and, at run time, execute any sequence of actions that constitutes a valid instance of the plan. For example, consider our skier when practicing basic skills on a very familiar trail. For example, he or she might decide to practice tight turns down the fall line at the top of the trail, which is known to be narrow and steep, and big carving turns at the bottom, which is known to be broad and shallow. The skier does not bother to specify the exact course or speed, the number of turns of each type, or when the transition should occur. He or she confidently instantiates this general plan at run time in any of many equally valid sequences of specific turns, adjustments, etc. with little need to monitor the environment.

When an agent has minimally constraining goals to achieve in a maximally uncertain environment, *reflex mode* produces the best run-time performance at less preparation cost than reactive mode. The agent is sensitive to many classes of run-time conditions and is

willing to perform any actions the actual conditions elicit. For example, consider our skier when performing “extreme skiing” on a treacherous untracked trail never before encountered. He or she will be happy to reach the bottom of the trail without serious mishap. It is essential that the skier remain sensitive to a very broad range of possible conditions and be willing to perform an unusually broad range of actions, including for example stopping, walking, climbing, and radioing for help. At every point on the trail, the skier executes whatever actions are necessary to survive the immediate situation.

Notice that in both dead reckoning and reflex modes, the agent is positioned implicitly to perform a much larger number of specific actions and action sequences, compared to planning and reactive modes, respectively. Dead reckoning mode exploits predictability in the environment to commit to large classes of actions, which can be instantiated top down as alternative acceptable courses of action at run time. Without this predictability, reflex mode relies heavily on run-time monitoring to detect conditions that elicit actions. In the intermediate modes along the dead reckoning-reflex border, the agent modulates the amount of run-time monitoring it performs and the balance of top-down versus bottom-up control of actions it executes. In all modes on this border, however, the agent expends minimal computational resources in advance by identifying arbitrarily large classes of events to monitor or actions to perform. It does not commit to monitor any specific events or to perform any specific actions. In that sense, and in contrast to planning and reaction modes, the agent is always responding to “unanticipated” events or performing “unanticipated” actions.

As mentioned above, we refer to modes in the large interior region of the mode space as *strategic modes*. Depending on the uncertainty of its environment, the degree of constraint imposed by its goals, and the availability of resources for preparation and monitoring, the agent makes a plan of intermediate specificity that more or less constrains the actions it intends to execute. The agent works toward its goals by executing actions that happen to be triggered at run-time and happen to match its plan. For example, when skiing fast down a somewhat familiar trail, our skier makes a general plan to perform tight turns down the fall line, but monitors the trail at run time to decide exactly which turns to make where, whether any unplanned classes of actions are required, etc.

Notice that, in strategic mode, the agent does not waste resources making plans that are unrealistically or unnecessarily specific, but remains open to the possibility of executing alternative actions and action sequences that only become apparent at run time.

On the other hand, it is not unduly sensitive to the environment, but organizes its monitoring and actions within the general structure of its strategic plan. Strategic mode produces good run-time behavior in a broader range of more realistic situations than any of the extreme modes along the borders of the space.

Given the situation-specific effectiveness of alternative control modes and their differential demands for advance and run-time resources, we need a control mechanism that will allow an intelligent agent to operate in any mode throughout the two-dimensional space, at its own discretion. The mechanisms of planning and reactive models reported in the literature are mode-specific; they basically support control modes only in the correspondingly named extreme regions of the two-dimensional mode space. While these mechanisms might be useful for special-purpose applications, they do not provide the range of control modes required of intelligent agents. On the other hand, both mechanisms contain elements of a comprehensive control architecture. Integrating and extending these component processes, we propose the *opportunistic control model*. Section 2 describes the model conceptually. Section 3 describes an implemented architecture and representation language that realize the model. Section 4 explains how the opportunistic model supports the two-dimensional space of control modes. Section 5 discusses evaluation of the model.

2. The Opportunistic Control Model

The *opportunistic control model* comprises three component processes. Preserving important properties of the reactive model, an *event-based triggering process* identifies a subset of possible actions at run time. Preserving important properties of the planning model, a *strategic planning process* dynamically constructs and modifies plans that constrain intended actions. Integrating the outputs of these two processes, a *match-based control process* selects for execution possible actions that satisfy strategic plans. To illustrate the following discussion, we turn to a new domain, a hypothetical “errand robot” as it drives east on Jackson Avenue, one of the streets in its environment. Outputs of the robot’s triggering, planning, and control processes appear in Figure 2.

PLEASE INSERT FIGURE 2 HERE

2.1 The Event-Based Triggering Process

An agent’s detection of run-time events triggers a subset of its possible actions.

As an agent operates in its environment, it detects a stream of perceptual and cognitive events (changes), which trigger a stream of possible actions. By “trigger,” we mean notice that an action’s enabling conditions are satisfied, bind its parameters in the current context, and produce an executable instance of the action’s execution code. Thus, a “possible action” is a fully instantiated action, whose execution code the agent could execute “now” if it chose to do so. For example, the errand robot might know the action:

Name: Evaluate-posted-sales
 Trigger-condition: Detect-sale-sign at @site.
 Action: Evaluate-sale at @site.
 Execution-code: ...

While driving east on Jackson Avenue (during step P2-B in Figure 2), the robot might detect sale signs at three stores, potentially triggering three possible actions:

Evaluate-sale at Truc.
 Evaluate-sale at Manny’s.
 Evaluate-sale at Artifactory.

Depending on run-time conditions, many actions may be logically applicable--that is, their trigger conditions may be satisfiable, given the state of the environment. But, an agent has goals to achieve and limited resources; so it typically notices only a subset of run-time conditions, triggers only a subset of possible actions, and executes only a subset of triggered actions. For example, at “Now” in Figure 2, the errand robot is executing a single possible action, it has triggered but not yet executed three other possible actions, and it has not triggered two other actions that are logically applicable. The robot may or may not subsequently execute the three triggered actions or trigger the two logically applicable actions, depending on what other events occur, how it responds to them, etc. Naturally, an agent is biased to trigger and execute actions that are relevant to its goals. This bias may be strong or weak, depending on the specificity of the goals and the availability of computational and real-time resources. For example, in Figure 2, the robot’s goal in plan P2-B (“Survey-sales at clothing stores”) makes it more likely to notice sale signs at clothing stores and to trigger and execute associated “Evaluate-sale” actions. By contrast, if the robot were late for an important appointment, it would be more likely to notice conditions and trigger actions related to parking; it might not notice the sales at all. But in all cases, the bias affects only the probability, not the possibility of triggering and executing logically applicable actions.

Triggering possible actions based on run-time events preserves a key strength of the reactive model: it recognizes that, in most cases, specific opportunities to act are not explicitly anticipated, but only become apparent in the course of an agent's interactions with the environment. This may be because the agent need not, cannot, or simply does not anticipate them. For example, the errand robot need not anticipate the opportunity to evaluate a sale at Manny's; it can recognize that opportunity when it detects the sale sign. In fact, the mechanism of the reactive model offers only a limited form of this capability because it conflates enabling and goal conditions to invoke exactly one "optimal" action for execution at each point in time. For example, under the reactive model, the robot triggers and immediately executes the action "Evaluate-sale at Manny's" if and only if: (a) it detects a sale sign at Manny's and (b) evaluating a sale at Manny's is a goal. Because the reactive model hard-wires an agent's goal hierarchy, run-time events control the agent's actions. However, the agent is not capable of executing possible actions that have not been prepared explicitly to serve predefined goals in the context of anticipated run-time events.

By contrast, the opportunistic model uses only enabling conditions for triggering and may identify several possible actions at each point in time. For example, the errand robot can trigger the action "Evaluate-sale at Manny's" whenever it detects a sale sign at Manny's--either because it has looked for one in an effort to achieve its goals or because it happened to notice one. Depending on its goals, the robot may or may not execute the triggered action. Thus, an agent can change its goals (e.g., in response to run-time events) without changing the way it triggers actions. And the agent is logically capable of noticing and executing actions that lie outside of its current goals. This is an extreme form of responsiveness to the environment that neither the planning nor reactive model allows, but one that can be useful, for example in exploratory behavior, in making use of perishable slack resources, or in adapting goals themselves to run-time demands and opportunities. Thus, in the opportunistic model, *run-time events enable, but do not control an agent's actions.*

Excluding goal conditions from triggering has other consequences. It expands the set of conditions in which a given action is possible, increases the number of distinct condition-action pairs that can be represented in a space-bounded agent, decreases the agent's chance of encountering unanticipated conditions, and increases the number of unique sequences of actions the agent can execute. It simplifies the knowledge engineering task because, with goal-related strategic knowledge factored out, the system builder only needs to catalogue the

conditions that logically enable each action, not all of the possible goal-specific instances and sequences of actions. Finally, allowing a subset of run-time conditions to trigger a subset of possible actions, rather than exhaustively perceiving the environment to discriminate among all possible actions, presents a more realistic objective for bounding perception time [8, 9, 47].

On the other hand, triggering actions by enabling conditions fails to alleviate--in fact, exacerbates--the reactive model's uncertain global effectiveness, efficiency, and coherence of behavior. And it does not provide a mechanism for selecting one of several possible actions to execute at each point in time. As discussed below, "strategic plans" give an agent's behavior its global properties and the "match" between triggered actions and strategic plans controls the selection of specific actions to execute.

2.2 The Strategic Planning Process

An agent dynamically constructs strategic plans that constrain its intended actions.

An agent dynamically constructs, refines, and modifies strategic plans for its own actions. For example, given a goal to "Perform the day's errands," the errand robot might construct and follow the strategic plan P2 in Figure 2. P2 is comparable to the sort of abstract plan that might be generated and refined top-down by a traditional planning system. Unlike the traditional model, we assume that planning occurs dynamically at run-time. An agent may or may not construct a complete plan prior to beginning execution. It may refine or change already constructed parts of the plan at any time prior to completing execution of their associated actions. For example, in Figure 2, the errand robot has completed step P2-A and is "now" in the process of executing actions under P2-B. For the time being, it is free to refine or modify any of P2-B--P2-E or to introduce additional plan elements.

By "strategic plan," we mean an abstract description that more or less constrains the kinds of actions an agent intends to perform during some time interval. Unlike traditional plans, a strategic plan ordinarily does not specify any directly executable actions--that is, any fully instantiated actions that have associated execution code. For example, plan step P2-B specifies a class of actions: "Survey-sales at clothing stores." It does not specify any directly executable actions, such as those discussed above: "Evaluate-sale at Truc," "Evaluate-sale at Manny's," "Evaluate-sale at Vera's." Instead, it implicitly allows the robot to execute any of these or other executable actions that belong to the designated class,

such as “Look-for-sale at Manny’s” or “Go-to-sale at Manny’s”. Conversely, P2-B implicitly excludes actions that do not belong to the designated class, such as “Admire the acacia trees near the curb.” Miller, Galanter and Pribram anticipated our concept of strategic plans:

We call them “plans” without malice--we recognize that you do not draw out long and elaborate blueprints for every moment of the day. You do not need to. Rough, sketchy, flexible anticipations are usually sufficient. [26]

Strategic plans of different specificity differentially constrain an agent’s actions. For example, in P2-B, the errand robot plans to “Survey-sales at clothing stores.” Because there are three clothing stores on Jackson Avenue (Truc, Manny’s Shoe Store, and Vera’s Dress Shop), the robot implicitly plans to survey sales at any or all of them if possible--that is, to the degree that run-time conditions and resources allow. Alternatively, the robot might plan to “Survey-sales at good clothing stores.” If Manny’s and Vera’s are the only good clothing stores, the robot implicitly plans to survey sales only at Manny’s or Vera’s. Alternatively, the robot might plan to “Survey-sales at stores.” In this case, it implicitly plans to survey sales not only at the three clothing stores, but also at any other stores on Jackson Avenue. Notice that, in all of these cases, the robot need not even know in advance what kinds of stores exist on Jackson Avenue. It can instantiate its strategic plan as appropriate for whatever stores it encounters there at run time.

Strategic planning preserves a key strength of the traditional planning model: it recognizes that an agent cannot be at the mercy of its sensors, but must be able to construct and follow a goal-oriented course of action. The errand robot need not exhaustively evaluate the environment and reassess all possible actions at every moment on its journey; it can focus attention on activities that help it follow the goal-oriented plan it previously adopted. In fact, the traditional planning model offers only a limited form of this capability because it conflates goal and enabling conditions to plan correct and complete sequences of directly executable actions. For example, under the planning model, the errand robot would look for a sale at Manny’s if and only if it had anticipated the run-time conditions under which it would be able to do so (e.g., while stopped for a pedestrian in the nearby crosswalk or while driving past Manny’s) and had specifically planned to perform that action at one of those times. Because the planning model makes strong predictions of run-time conditions, a directly executable plan controls an agent’s actions. However, the agent is not capable of executing possible actions whose run-time applicability was not explicitly anticipated.

By contrast, the opportunistic model emphasizes goal conditions in abstract plans that permit alternative sequences of action, only one of which an agent eventually executes. For example, the errand robot's step P2-B permits many alternative selections and sequences of specific "Survey-sales" actions--only some of which may turn out to be logically possible. Thus, the agent can reuse a given plan in many different situations, despite the fact that very different events are likely to occur. Moreover, in many situations an agent need not reason out detailed logical sequences of actions because, to a large degree, they follow regularities in the run-time events that trigger them [36]. For example, the errand robot can plan to survey sales at clothing stores without planning the exact sequence of executable actions. Detection of a clothing store will trigger the action to look for a sale. Detection of a sale will trigger the action to evaluate it. And a good evaluation will trigger the action to go to the sale. The robot's plan to survey sales at clothing stores will favor execution of these actions when they are triggered and execution of these actions will produce events that trigger their logical successors. On the other hand, it is possible for an agent to find itself with no triggered actions that serve its plan. In that case, it may attempt to perform actions whose outcomes will trigger plan-relevant actions. For example, if the robot happened not to notice any sale signs while driving east on Jackson Avenue, it might redirect its sensors in an effort to find a previously overlooked sign. Depending on run-time conditions, resource constraints, etc., an agent executes exactly one of the sequences of possible actions allowed by its plan. Thus, *strategic plans constrain, but do not control an agent's actions.*

Strategic planning preserves other features of the traditional planning model. An agent can construct globally effective plans, focus its perception of the environment, modify its plans in light of unanticipated events, and follow a coherent, comprehensible course of action. Methods for generating traditional executable plans can be used to generate strategic plans too. However, because strategic plans have a smaller number of larger "steps," the cost of planning and replanning decreases and can be managed to fit the available resources. Because plans constrain classes of intended actions, rather than specifying complete and correct sequences of executable actions, flexibility increases. Together, these two properties make for more robust plans that work for a wider range of unanticipated conditions and are easier to repair when necessary. In some cases, an agent has to balance competing properties of its behavior. For example, achieving global effectiveness and coherence may require more detailed plans that decrease run-time flexibility and increase the cost of planning and replanning. Strategic planning allows an agent to balance competing properties by planning at an appropriate level of specificity.

On the other hand, strategic plans do not provide a mechanism for triggering and selecting specific actions to execute. As discussed above, specific actions are triggered by an agent's detection of their enabling events. As discussed below, the "match" between triggered actions and strategic plans controls the selection of specific actions for execution.

2.3 The Match-Based Control Process

An agent controls its behavior by executing possible actions that match its strategic plans.

An agent controls its behavior by executing, at each opportunity, a currently triggered possible action that matches its currently active strategic plan--to the degree that doing so is practicable. By "match," we mean satisfy the constraints in the plan. Other things being equal, the agent prefers to execute actions that match its plan better than others. For example, at "Now" during P2-B in Figure 2, the errand robot plans to: "Survey-sales at clothing stores." It chooses to execute a possible action that perfectly matches its plan: "Evaluate-sale at Manny's." This is the right action class at the right store class. The robot prefers this action over three others that do not match its plan as well. "Admire the roses at Manny's" is the wrong action class at the right store class. "Evaluate-sale at Artifactory" is the right action class at the wrong store class. "Admire the acacia trees near the curb" is wrong on both counts. On the other hand, if no perfectly matching action had been triggered and slack resources were available, the robot might choose to execute one of the two partially matching actions, or even the non-matching action, rather than no action at all.

An agent's current triggered actions and, to a lesser extent, its current strategic plan, are inherently dynamic and adaptive to run-time conditions. Triggered actions can change whenever the agent detects new conditions. The strategic plan changes--progresses to the next step or undergoes modification--whenever the agent changes it. Like everything else an agent does, changing its strategic plan involves executing an action that is triggered by detection of an enabling condition and selected for execution when it matches a current plan. For example, the errand robot has an action for advancing to the next step of an active plan whenever the goal of the current step has been met:

```
Name: Advance-plan
Trigger-condition: Goal of @current-step of @plan is true
Action: Advance @plan to step after @current-step
Execution-code: ...
```

To insure prompt execution of this action whenever it is triggered, the robot's strategic plan (Figure 2) includes a persistent plan, P1: "Maintain the plan." Any triggered instance of the "Advance-plan" action will perfectly match P1 and so be favored for execution.

PLEASE INSERT TABLE 1 HERE

An agent follows its strategic plan to the degree that doing so is practicable. Its actual course of action depends on the interaction between its current strategic plan and its current triggered actions--both of which depend on its detection of and response to run-time conditions. For example, under step P2-B, if only Manny's has a sale sign and few demanding driving conditions occur, the errand robot is quite likely to look for and find the sale sign and then to evaluate the sale. However, if Truc and Manny's and Vera's all have sale signs and many demanding driving conditions distract the robot (under persistent plan P0), it may not have enough resources to survey all three sales while driving. At the other extreme, if there are no sale signs at any stores and few special driving conditions, the robot will look for sale signs, but find none to evaluate. In that case, given its perishable slack resources, the robot might execute a triggered action that does not match its plan at all, such as admiring the landscape. Table 1 summarizes a few run-time situations that would produce different courses of action under P2-B. Thus, *run-time events enable possible actions, strategic plans constrain intended actions, and the match between possible actions and strategic plans controls action execution.*

3. Architecture and Representation for the Opportunistic Control Model

The opportunistic control model can be implemented in different ways, with different performance implications. To retain flexibility in exploring the space of design decisions, our implementation has two levels, architecture and representation. Some of the details at both levels are parameters and subject to revision. In particular, the numerical functions used for rating the match between possible actions and strategic plans could be replaced by qualitative functions. Nonetheless, the present implementation illustrates one way in which the opportunistic control model has been implemented for use in building intelligent agents. The architecture is described in previous articles, so we only summarize the relevant features here. We discuss the representation language and the associated match function in more detail.

3.1 The Dynamic Control Architecture

As illustrated in Figure 3, the *dynamic control architecture* [17, 18] comprises three kinds of subsystems, which operate concurrently and asynchronously. One or more perception systems filter sensed data under dynamic instructions from the reasoning system. A single reasoning system interprets perceived events, performs reasoning and problem solving, and decides what physical actions to perform. Action systems control the execution of physical actions by effectors. As summarized below, the reasoning system operationalizes the three basic processes of opportunistic control.

PLEASE INSERT FIGURE 3 HERE

On each run-time reasoning cycle, the agenda manager triggers possible actions based on detection of their enabling conditions. Its inputs are: (a) a set of cognitive and perceptual events (*Es*) (changes to memory or the environment) that have occurred during the last cycle; (b) a set of active control plans (discussed below); and (c) a set of *knowledge sources* (*KSs*), which define known actions as:

Name: Unique identifier
 Triggering-condition: Predicates on an event
 Action-parameters: Assignment of context-specific values to parameters
 Potential-action: Parameterized description of an executable action

For every E-KS pair in which E satisfies KS's triggering conditions, it binds the KS's parameters to their context-specific values, creates a corresponding *knowledge source activation record (KSAR)* :

Name: Unique identifier
 Possible-action: Instantiation of the executable action

and places it on the *agenda*. As mentioned above, limited resources cause an agent to notice only a subset of run-time conditions and to trigger and execute only a subset of logically applicable actions, typically those most relevant to its goals. The architecture uses control plans to support this focus in two ways. First, perception systems filter and prioritize perceptual events based on both their relevance to the current control plan and their intrinsic importance [47]. Second, the agenda manager uses a "satisficing algorithm" to order and interrupt its consideration of events and KSs based on both their relevance to the control plan and their intrinsic importance [9]. Thus, at any point in time, the agenda

contains a subset of the most relevant and important of the agent's possible actions--actions it can execute now if it chooses to do so.

The control plan is a data structure in which an agent dynamically constructs strategic plans to constrain its intended actions. The control plan may contain several concurrent plans, each comprising a graph of related *control decisions*:

Decision-Name: A unique identifier
 Strategic-plan: Description of intended actions
 Activation-interval: (Initiation-Time Termination-Time)
 Current-status: Active or inactive
 Goal: Predicate on events or states in memory
 Weight: Scaled numerical measure of importance
 Relations: Precedes, Sub-Goal-Of, ...
 Expansion-method: Method for generating subordinate decisions

Control knowledge sources, which are triggered, scheduled, and executed like other knowledge sources within the reasoning cycle, dynamically construct and modify control plans at run-time. Control KSs are quite general and apply to many specific control plans. For example, one control KS is triggered by the appearance of any new control decision that has an *expansion-method*. When executed, it uses the decision's expansion-method to generate a sequence of more specific subordinate decisions. Another control KS is triggered whenever the *goal* of any active control decision becomes true. When executed, it deactivates that control decision and activates the one that succeeds it in the superordinate plan. An agent can use these and other control KSs to change its control plan on any reasoning cycle. But at any point in time, the control plan's currently active strategic plans constrain the kinds of actions the agent intends to execute now if it can do so.

On each cycle, the scheduler controls action by choosing the KSAR (possible-action) that best matches the active control plans. It uses a *match function* that takes as inputs: (a) the current agenda of KSARs; and (b) a list of active terminal control decisions that have no subordinates. For each KSAR and each control decision, the match function rates the degree to which the KSAR's possible-action satisfies the constraints in the control decision's strategic-plan. If there is more than one active control decision, the scheduler computes some function that integrates the KSAR's ratings across decisions, for example the sum of weighted ratings or the highest single weighted rating. It chooses the KSAR with the highest integrated rating. An *executor* executes the chosen KSAR, producing changes to the global memory, which may be to produce an inference related to domain reasoning, to place a

physical action description in the buffer for transfer to an action system, or to change the control plan itself. In all cases, the action produces associated cognitive events.

3.2 The Representation Language

3.2.1 Possible Actions and Strategic Plans

Our representation language reifies actions and action classes as conceptual entities about which an agent can have knowledge and about which it can reason. As illustrated by the excerpt from the errand robot's action class hierarchy in Figure 4, action hierarchies are defined with "is-a" and "can-be-a" links. Other links represent other kinds of relations with other actions, events, and states. For example, terminal actions have "is-triggered-by" links to the classes of events that can trigger them and "causes" links to the classes of events they produce when executed. (Events and states are defined in similar class hierarchies.) Selected attributes and link relations can be inherited over specifiable chains of link types, as well as over the traditional "is-a" link.

Possible actions and strategic plans are represented by instantiating action templates with appropriate parameter values. Each action's *template* lists its formal parameters in an English-like sentence:

@action {noise words} @parameter-1 {noise words} @parameter-2 ...

Example: Look-for-sale at @store.

For example, the template for the action "Look-for-sale" lists the action "Look-for-sale," followed by the noise word "at," followed by the formal parameter "@store." *Parameter-constraints* restrict legal values for parameters to concepts in the specified classes (see Figure 5), preceded by any number of modifiers defined for those classes. For example, legal values for the parameter "@store" in the action "Look-for-sale" must be concepts related to the concept "store" by a chain of one or more "is-a" links, preceded by any number of modifiers defined for any of the concepts along that chain. *Modifiers* define procedures for making discrete or scaled evaluations of particular instances of actions. For example, the modifier "Reactively," defined for the action "Survey-sales," returns a value of 100 for instances of the actions "Evaluate-sale" and "Go-to-sale," but a value of 0 for instances of "Look-for-sale." The modifier "Quickly," defined for a higher-level action not shown in Figure 4, returns a value scaled 0-100 that is inversely related to the value of an

action's "Average-time." As these examples illustrate, most parameter-constraints and modifiers are defined for higher-level action classes and inherited by their subordinates.

Only terminal actions have executable *code*, which is parameterized with the same formal parameters used in their templates. For example, the errand robot has no code for and cannot execute a "Survey-sale" action except by executing instances of its terminal subordinates, "Look-for-sale," "Evaluate-sale, or "Go-to-sale." On the other hand, all actions have *execution-properties* that describe actual properties of terminal actions and, for action classes, typical properties of their terminal subordinates.

Finally, a *cross-reference table* (not shown in Figure 4) specifies which formal parameters in different templates semantically correspond to one another. Obviously, all actions correspond semantically to one another. To take a less obvious example, the parameter "@store" in templates for "Survey-sale" and its subordinates and the parameter "@location" in templates for "Look-around" and its subordinates all semantically correspond to one another because all represent the location of the object of the action in their respective templates.

PLEASE INSERT FIGURE 4 HERE

The formal parameters in any action template can be instantiated with legal phrases:

```
{Action-A-Modifiers} Action-A
{noise words} {Concept-C-Modifiers} Concept-C
{noise words} {Concept-D-Modifiers} Concept-D
...
```

Example: Reactively survey-sale at good expensive clothing-store.

Actions and modifiers for "@action" phrases come from the action hierarchy discussed above. Concepts and modifiers for other parameter phrases come from a similar concept hierarchy. As shown in Figure 5, the errand robot's concept hierarchy defines concepts relevant to its errand task, such as different kinds of stores. *Modifiers* define procedures for making discrete or scaled evaluations of concepts. For example, the modifier "Good," defined for "Store," returns 100 for "Manny's" and "Vera's," but 0 for other stores. The modifier "Expensive," defined for "Clothing-store," returns a value scaled 0-100 that is positively related to the value of a clothing-store's "average-purchase-price." Only terminal concepts exist in the world. For example, "clothing-store" does not exist in the

errand robot's world, except in the form of its terminal subordinates, "Manny's," "Truc," and "Vera's." However, all concepts have *properties* that describe actual properties of terminal concepts and, for concept classes, typical properties of their terminal subordinates.

PLEASE INSERT FIGURE 5 HERE

Possible actions are represented by instantiating the formal parameters in a terminal action template with terminal concepts. That is the only way to express a directly executable action operating on objects that actually exist in the world. For example, the template for the terminal action "Look-for-sale" can be instantiated as different possible actions by instantiating its formal parameter "@store" with the names of particular stores:

Look-for-sale at Manny's.
 Look-for-sale at Truc.
 Look-for-sale at Vera's.

Each terminal action template can be instantiated as a different possible action for each combination of terminal concepts that legally instantiate its formal parameters. Within the dynamic control architecture, the agenda manager determines which possible actions actually are instantiated on the agenda at each point in time by triggering associated knowledge sources and binding their formal parameters in the context of particular run-time events.

Strategic plans are represented by instantiating the action or formal parameters of any template with any legal phrases. The specificity of the instantiating phrases determines how much the strategic plan constrains the intended actions. For example, the following strategic plans impose successively more constraints on the errand robot's intended actions:

Survey-sale at store.
 Survey-sale at good clothing-store.
 Reactively survey-sale at good clothing-store.
 Evaluate-sale at Manny's.

The first plan is quite general, allowing the errand robot to "Look-for-sale," "Evaluate-sale," or "Go-to-sale" at any store--a large space of actions. The second plan is more constrained, allowing the robot only six actions, "Look-for-sale," "Evaluate-sale," or "Stop-for-sale" at Manny's or Vera's. The third plan is still more constrained, allowing

only four intended actions, “Evaluate-sale” or “Go-to-sale” at either Manny’s or Vera’s. The fourth plan, which instantiates both the action and formal parameters with terminal values, allows only the single intended action it explicitly describes, “Evaluate-sale at Manny’s.” Each action template in the hierarchy can be instantiated as a different strategic plan for each combination of phrases that legally instantiate its action and formal parameters. The degree of constraint imposed by a given strategic plan is inversely related to the number of distinct intended actions it allows, which is simply the product of the number of legal terminal instantiations for its constituent phrases. Within the dynamic control architecture, the execution of control knowledge sources determines which strategic plans actually are instantiated on the control plan by instantiating action and parameter phrases in the context of particular run-time events.

3.2.2 The Match Algorithm

The match algorithm rates a possible action against a strategic plan by quantifying the degree of class membership and modifier satisfaction between their corresponding parameter values. Different match algorithms are possible. We present one based on hierarchical averaging of numerical ratings, scaled 0-100. The overall rating for a possible action against a strategic plan is the average rating for its parameter values against the corresponding parameter phrases in the plan. The rating for a parameter value against a parameter phrase is its average rating against each significant term in the phrase. The rating of a parameter value against a term is determined by evaluation of is-a relations and the procedural definitions of relevant modifiers, all of which return discrete or scaled values 0-100.

For example, given this strategic plan:

Strategic Plan: Quickly survey-sale at good clothing-store.

this possible action:

Possible Action: Look-for-sale at Manny’s.

rates: 60 on its “@action” parameter because (is-a look-for-sale survey-sale) returns 100 and (quickly look-for-sale) returns 20; and 100 on its “@store” parameter because (is-a Manny’s clothing-store) and (good Manny’s) both return 100. Averaged together,

these give an overall rating of 80. On a scale 0-100, this is a pretty good match and a high rating. By contrast, this possible action:

Possible Action: Look-for-sale at Truc.

rates: 60 on the “@action” parameter because (is-a look-for-sale survey-sale) returns 100 and (quickly look-for-sale) returns 20; but only 50 on its “@store” parameter because (is-a Truc clothing-store) returns 100 and (good Truc) returns 0. Together these give an overall rating of 55. Other things being equal, the robot prefers the higher-rated action: “Look-for-sale at Manny’s.”

Within the dynamic control architecture, the scheduler uses the match algorithm to rate the possible action in each KSAR on the agenda against the strategic plan in each active control decision. It chooses to execute the possible action with the highest weighted sum of ratings (or highest value on another integrative function) against all active strategic plans.

4. Support for Alternative Control Modes

We hypothesize that the opportunistic control model allows an agent to operate under control modes throughout the theoretical space of control modes and to make smooth transitions among them by modulating two key properties of its control plan: the specificity of the classes of planned actions and the degree of sequential organization in the plan. Figure 6 summarizes our hypotheses, which are discussed below.

To operate in planning mode, an agent constructs a control plan that describes an explicit sequence of specific executable actions. As the agent works its way through the plan, its perceptual preprocessor severely filters perceived events to focus on just those--if any--that are necessary to trigger each next planned action. Its agenda manager severely restricts its consideration of events and KSs in order to trigger only each next planned action. Assuming the plan is valid, on each cycle the agenda manager immediately triggers and executes each planned action in turn. Overall, the agent executes its planned sequence of specific actions very fast (within the limits of the environmental data rate and its action execution rate), with minimal monitoring and high global coherence (determined by the quality of its plan). However, its control plan will work only in the situation for which it is constructed and will fail if its predictions of environmental conditions fail.

PLEASE INSERT FIGURE 6 HERE

To operate in reaction mode, an agent constructs a control plan comprising a set of specific condition-action rules, with no explicit sequentiality. Its perceptual preprocessor evaluates all events that could trigger any of the rules. Its agenda manager considers any events and KSs whose combination could instantiate any of the conditional actions, ordered by rule importance. Assuming the control plan is valid, on each cycle the agent executes the first action it instantiates. Overall, the agent executes some more or less coherent sequence of its conditional actions rather fast, although not as fast as if it had planned that particular sequence. On the other hand, its goal-directed behavior is robust over a broader range of run-time situations.

To operate in dead reckoning mode, an agent constructs a control plan that describes an explicit sequence of a few very general action classes. As the agent works its way through the plan, its perceptual preprocessor filters perceived events to admit any that might trigger any actions in the current planned action class. Its agenda manager restricts its consideration of events and KSs in an effort to trigger any action within the current planned class. Assuming the plan is valid, on each cycle the agenda manager triggers and executes a single action within the current planned class, updating its position in the plan as appropriate. Overall, the agent executes some sequence of specific actions within the planned sequence of action classes. Its choice of successive actions is slower than it would be in planning or reaction mode because it cannot focus its perception or constrain its agenda manager as well. But its behavior is robust over a very broad range of run-time situations and has a moderate amount of global structure and coherence.

To operate in reflex mode, an agent constructs a control plan comprising a set of general condition-action-class rules, with no explicit sequentiality. The agent's perception preprocessor weakly filters perceived events to admit any that might trigger any of the rules. Its agenda manager considers any events and KSs whose combination could instantiate an instance of any of the action classes, ordered by importance. Assuming the control plan is valid, on each cycle the agent executes the first action it instantiates. Overall, the agent executes some sequence of actions. Its choice of successive actions is slower than in reactive mode because it cannot focus its perception or constrain its agenda manager as well. And its behavior has minimal global structure. But its control plan is extremely robust over an extremely broad range of run-time situations.

To operate in strategic mode, an agent constructs a control plan that has some intermediate degree of both specificity and sequentiality. The illustrative control plan in Figure 2 is a good example. As the agent works its way through the plan, its perception preprocessor moderately filters perceived events to focus on those that might trigger actions in the currently planned action class. Its agenda manager moderately constrains the order in which it considers events and KSs to trigger actions in the planned class quickly. Overall, the agent executes a somewhat variable sequence of specific actions within a fairly well constrained sequence of moderately specific action classes. Its behavior is similarly intermediate among the extreme control modes with respect to the speed of choosing successive actions, global structure, and robustness.

5. Evaluation of the Opportunistic Control Model

We have begun to analyze and empirically evaluate some predictions of the model's performance. For example, we have made preliminary studies of predicted effects of increasing the specificity of an agent's strategic plan. (a) The agent will spend more time matching possible actions to strategic plans on each reasoning cycle, but it will require fewer reasoning cycles to complete its task. Depending on the magnitude of match time vs. other cycle functions, this relation entails a quantitative prediction of changes in total task time [13]. (b) The agent will ignore a greater number of possible actions, but more reliably execute those favored by its plan. Depending on the distribution of run-time events vs. the class of possible actions favored by the plan, this relation entails a quantitative prediction of changes in the global utility of the agent's behavior [20]. (c) The agent can find possible actions that match its strategic plan more quickly. This entails quantitative predictions of improvements in the agent's reasoning speed, its ability to meet deadlines, and, depending on the correctness of the plan given run-time events, changes in the global utility of the agent's behavior [9]. Two other studies characterize the opportunistic control model's intrinsic trade-off of speed for flexibility of run-time performance compared to other control models [13, 28]. In all of these studies, the actions being controlled include a large number of cognitive actions and a smaller, but not insignificant number of communication actions. The latter include actions for exchanging information with human beings and, in some of the studies, actions for real-time control of simulated physical systems. Studies of other aspects of the model's behavior are ongoing.

Our implementation of the opportunistic control model has been used in many experimental applications (for example, errand planning [21], protein structure modeling [19], intensive care monitoring [22], tutoring [28], layout design [44]). This experience is useful not so

much to confirm that the model behaves as predicted, but rather to confirm that diverse real-world domains benefit from the services the model provides: triggering possible actions based on dynamic run-time conditions; dynamically constructing and modifying strategic plans that constrain intended actions; and controlling actions based on the match between possible actions and strategic plans. In all of these domains, the need for opportunistic control arises partially from uncertainty about the effects of the agent's own actions and, in the case of tutoring and intensive care monitoring, uncertainty about the occurrence of exogenously generated real-time events. In on-going work, we are continuing evaluation of the model in the intensive care application and in real-time monitoring applications in two engineering domains, semiconductor manufacturing [27] and power plant maintenance. Because building experimental applications is expensive in time and effort, concluding that they benefit from the services of a particular control model is based largely on subjective judgments made by a small number of people--we retain a proper degree of skepticism. However, we also believe that experience with challenging applications provides essential data regarding the ecological validity and ultimate utility of any proposed control model. Our experience with the opportunistic control model in the several domains mentioned above has been promising.

References

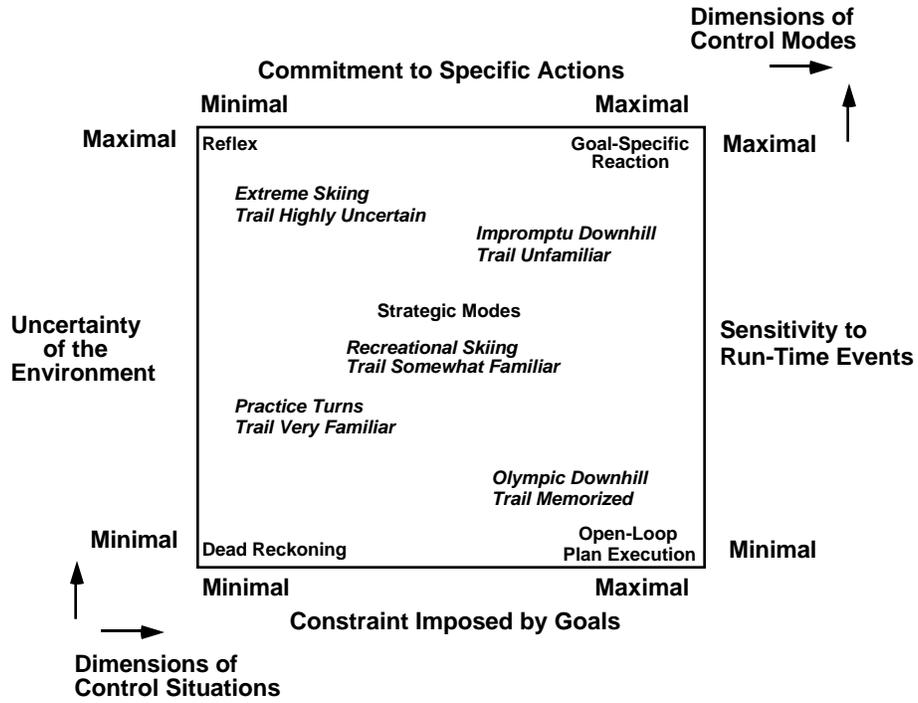
1. P. E. Agre and D. Chapman. "Pengi: An implementation of a theory of activity." In Proceedings of the National Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1987.
2. R. Alterman. "Adaptive planning." Cognitive Science, 12, 393-421, 1988.
3. F. C. Bartlett. Remembering, A Study in Experimental and Social Psychology. Cambridge, Boston: Cambridge University Press, 1932.
4. J. Bowermaster. "Uphill Racer." The New York Times Sunday Magazine, February 2, 1992.
5. R. A. Brooks. "Intelligence Without Reason." In Proceedings of the International Joint Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 569-595, 1991.
6. J. S. Bruner, J. Goodnow, and G. Austin. A Study of Thinking. New York: Wiley, 1956.
7. D. Chapman. "Planning for conjunctive goals." Artificial Intelligence, 32, 333-378, 1987.
8. L. Chrisman and R. Simmons. "Sensible planning: Focusing perceptual attention." In Proceedings of the National Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1991.
9. A. Collinot and B. Hayes-Roth. "Real-time performance of intelligent autonomous agents." In Proceedings of the Third European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW), Kaiserslautern, Germany, 1991.
10. W. K. Estes. Modern Learning Theory. New York: Appleton-Century-Crofts, 1954.

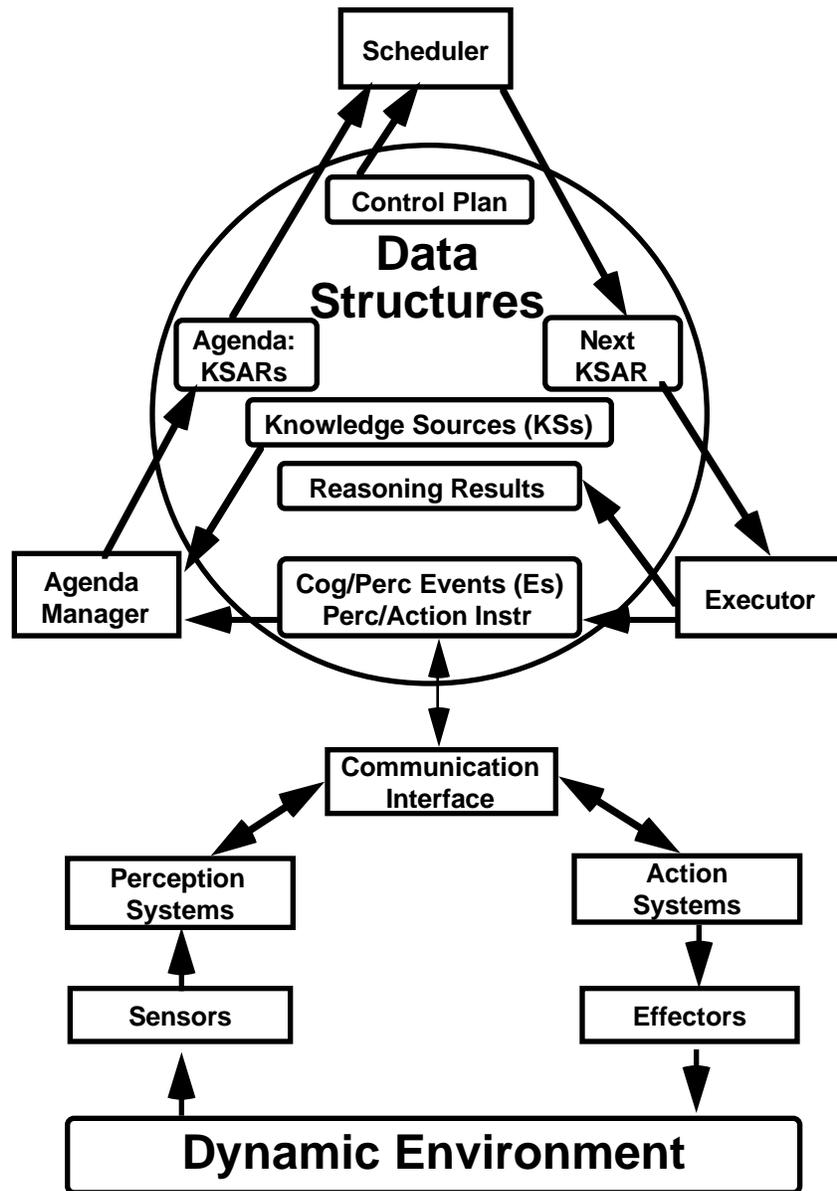
11. R. E. Fikes and N. J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." 2, 198-208, 1971.
12. A. Garvey, C. Cornelius, and B. Hayes-Roth. Computational Costs versus Benefits of Control Reasoning. In Proceedings of the National Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1987.
13. A. Garvey and B. Hayes-Roth. An empirical analysis of explicit vs. implicit control architectures. In V. Jagannathan and R.T. Dodhiawala (Eds.), Current Trends in Blackboard Systems, Academic Press, 1989.
14. J. J. Gibson. "A critical review of the concept of set in contemporary experimental psychology." Psychological Bulletin, 781-817, 1941.
15. M. L. Ginsberg. "Universal planning: An (almost) universally bad idea." Artificial Intelligence Magazine, 40-44, 1989.
16. K. J. Hammond. "Chef: A model of case-based planning." In Proceedings of the National Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1986.
17. B. Hayes-Roth. "A Blackboard for Control." Artificial Intelligence, 26, 251-321, 1985.
18. B. Hayes-Roth. "Architectural foundations for real-time performance in intelligent agents." Real-Time Systems: The International Journal of Time-Critical Computing Systems, 2, 99-125, 1990.
19. B. Hayes-Roth, B. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan, and O. Jardetzky. "PROTEAN: Deriving protein structure from constraints." In Proceedings of the National Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1986.
20. B. Hayes-Roth and A. Collinot. "Scalability of real-time reasoning in intelligent agents." Stanford University: Report KSL-91-08, 1991.

21. B. Hayes-Roth, F. Hayes-Roth, S. Rosenschein, and S. Cammarata. "Modelling planning as an incremental, opportunistic process." In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, pp. 375-383, 1979.
22. B. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, and A. Seiver. "Intelligent Real-Time Monitoring and Control." Proceedings of the International Joint Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1989.
23. D. O. Hebb. The Organization of Behavior. New York: Wiley, 1949.
24. R. E. Hilgard and D. G. Marquis . Conditioning and learning. New York: Appleton-Century, 1940.
25. W. Kohler. The Mentality of Apes. London: 1927.
26. G. Miller, E. Galanter, and K. Pribram. Plans and the structure of behavior. New York: Holt, 1960.
27. J. L. Murdock and B. Hayes-Roth. "Intelligent Monitoring and Control of Semiconductor Manufacturing." IEEE Expert, 6, 19-31, 1991.
28. W. Murray. "Dynamic instructional planning in the BB1 blackboard control architecture." In V. Jagannathan, R. Dodhiawala, and L. Baum. Current Trends in Blackboard Systems, (Eds.). Morgan Kaufman, 1989.
29. U. Neisser. Cognitive Psychology. New York: Appleton, 1967.
30. A. Newell and H. A. Simon. "The logic theory machine: A complex information processing system." IRE Transactions on Information Theory, 61-79, 1956.
31. I. P. Pavlov. Conditioned Reflexes. Oxford: Humphrey Milford, 1927.
32. M. Pollack. "Computers and Thought Lecture." In Proceedings of International Joint Conference on Artificial Intelligence, 1991.

33. S. J. Rosenschein and L. P. Kaelbling. "The synthesis of digital machines with provable epistemic properties." In Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge, Morgan Kaufman, pp. 83-98, 1986.
34. E. D. Sacerdoti. "The non-linear nature of plans." In Proceedings of the International Joint Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1975.
35. M. Schoppers. "Universal plans for reactive robots in unpredictable environments." In Proceedings of the International Joint Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1987.
36. H. Simon. The Sciences of the Artificial. Cambridge, MA: MIT Press, 1982.
37. B. F. Skinner. The Behavior of Organisms. New York: Appleton-Century-Crofts, 1938.
38. M. Stefik. "Planning with constraints." Artificial Intelligence, 111-140, 1981.
39. L. Suchman. Plans and Situated Actions. Cambridge University Press, 1987.
40. G. J. Sussman. A computational model of skill acquisition. MIT, Artificial Intelligence Laboratory, AI TR-297, 1972.
41. A. Tate. "INTERPLAN: A plan generation system that can deal with interactions between goals." Machine Intelligence Research Unit, University of Edinburgh, MIP-R-109, 1974.
42. E. L. Thorndike. Human Learning. New York: Century, 1931.
43. E. C. Tolman. Purposive behavior in animals and men. New York: Century, 1948.
44. I. Tommelein, R. Levitt, B. Hayes-Roth, and A. Confrey. "SightPlan Experiments: Alternate strategies for site layout design." Journal of Computing in Civil Engineering, 5, 42-63, 1991.

45. S. Vere. "Planning in time: Windows and durations for activities and goals." IEEE Transactions Pattern Analysis and Machine Intelligence, 246-267, 1983.
46. R. Waldinger. "Achieving several goals simultaneously." Machine Intelligence, 94-136, 1975.
47. R. Washington and B. Hayes-Roth. "Managing input data in real-time AI systems." In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, San Matao, Ca: Morgan Kaufmann, 1989.
48. D. E. Wilkins. "Domain-independent planning: Representation and plan generation." Artificial Intelligence, 22, 1984.





Name: Survey-sale
 Is-a: Gather-information
 Can-be-a: Look-for-sale, Evaluate-sale, Go-to-sale
 Relations: (...)
 Template: Survey-sale at @store.
 Parameter-Constraints: (Is-a @store 'store)
 Modifiers: (Reactively, ...)
 Execution-properties: (Average-time, Average-complexity, ...)

Name: Look-for-sale
 Is-a: Survey-sale
 Relations: ((Is-triggered-by Is-detected sign) (Causes Is-detected sale) ...)
 Template: Look-for-sale at @store.
 Code: (Look-for-sale-code @store)
 Execution-properties: (Time, Complexity, ...)

Name: Evaluate-sale
 Is-a: Survey-sale
 Relations: ((Is-triggered-by Is-detected sale), (Causes Is-evaluated sale...))
 Template: Evaluate-sale at @store.
 Code: (Evaluate-sale @store)
 Execution-properties: (Time, Complexity, ...)

Name: Go-to-sale
 Is-a: Survey-sale
 Relations: ((Is-triggered-by Is-evaluated good sale), ...)
 Template: Go-to-sale at @store.
 Code: (Go-to-sale @store)
 Execution-properties: (Time, Complexity, ...)

Name: Look-around
 Is-a: Gather-information
 Can-be-a: Admire, Observe
 Relations: ..(
 Template: Look-around at @thing in @location.
 Parameter-constraints: ((Or (Is-a @thing 'object) (Is-a @thing 'person)) ...)
 Modifiers: (Aesthetically, Socially ...)
 Execution-properties: (Average-time, Average-complexity, ...)

Name: Admire
 Is-a: Look-around
 Relations: ...
 Template: Admire @landscape at @location.
 Code: (Admire @landscape @location)
 Execution-properties: (Time, Complexity, ...)

Name: Observe
 Is-a: Look-around
 Relations: ...
 Template: Observe @occupants in @vehicle.
 Code: (Observe @occupants @vehicle)
 Execution-properties: (Time, Complexity, ...)

Name: Store
 Can-be-a: Clothing-store, Art-store, Furniture-store, ...
 Modifiers: (Good, Expensive ...)

Name: Clothing-store
 Is-a: Store
 Can-be-a: Shoe-store, Dress-store, Adult-clothing-store ...

Concept-Name: Shoe-Store
 Is-a: Clothing-store
 Can-be-a: Manny's, ...
 Properties: (...)

Name: Manny's
 Is-a: Shoe-store
 Properties: (Size, Average-purchase-price, ...)

Name: Dress-store
 Is-a: Clothing-store
 Can-be-a: Vera's, ...
 Properties: (Average-purchase-price, ...)

Name: Vera's
 Is-a: Shoe-store
 Properties: (Size, Average-purchase-price...)

Name: Adult-clothing-store
 Is-a: Clothing-store
 Can-be-a: Truc, ...

Name: Truc
 Is-a: Adult-clothing-store
 Properties: (Size, Average-purchase-price, ...)

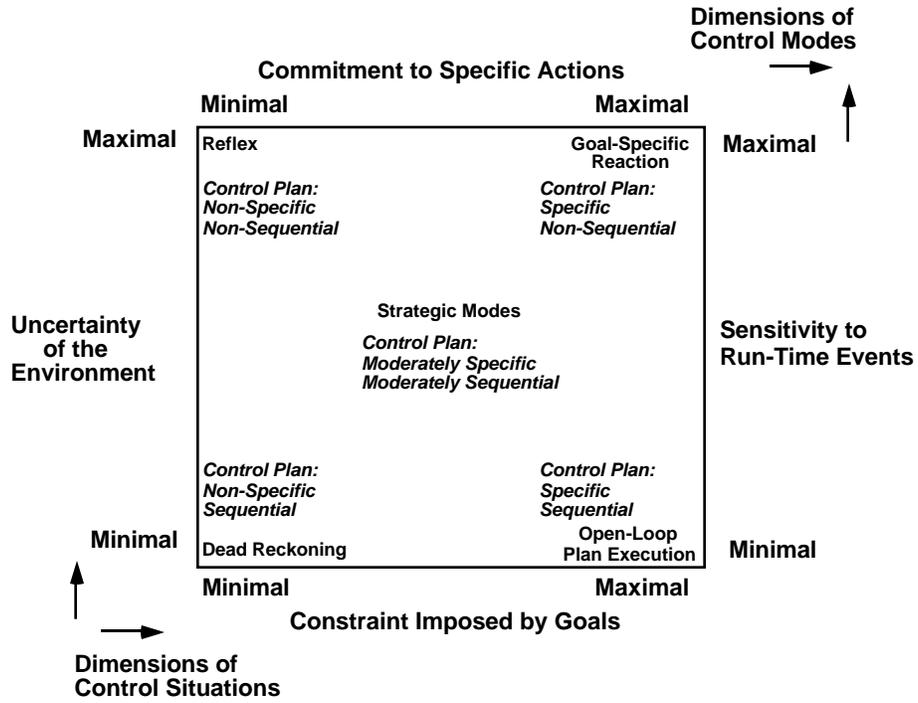


Table 1. Illustrative Run-Time Conditions and Associated Courses of Action under Plan P2-B

<i>Driving Conditions</i>	<i>Sale Signs</i>	<i>Likely Sale-Survey Actions</i>
Few	Manny's	Manny's
Many	Truc & Manny's & Vera's	Any or None
Few	None	None -- Admire the roses at Manny's
Few	All stores	Truc & Manny's & Vera's & Others?
Few	Stationery Store	Stationery Store

Figure Captions

Figure 1. Two-dimensional spaces of control situations and control modes.

Figure 2. Excerpt from the errand robot's control plan.

Figure 3. Overview of the dynamic control architecture.

Figure 4. Excerpt from the errand robot's action hierarchy.

Figure 5. Excerpt from the errand robot's concepts hierarchy.

Figure 6. Effecting different control modes by modulating the specificity and sequentiality of control plans.

Footnotes

1. Manuscript received ...

1. This research was supported by NASA contract NAG 2-581 under DARPA Order 6822 and by subcontract No. 71715-1 from Cimflex Teknowledge under DARPA contract No. DAAA21-92-C-0028, and AFOSR grant AFOSR-91-0131. We thank Richard Washington, Michael Wolverton, and three anonymous reviewers for helpful comments on an earlier version of the manuscript. We thank Ed Feigenbaum for sponsoring the research in the Knowledge Systems Laboratory.

Author's Biography

Dr. Barbara Hayes-Roth is Senior Research Scientists at the Knowledge Systems Laboratory, Stanford University. Her research concerns adaptive intelligent systems that must perform multiple knowledge-based reasoning tasks while interacting with a dynamic environment under real-time constraints. She received the PhD in 1974 and the MS in 1973, both in cognitive psychology from the University of Michigan, and her BA in 1971 from Boston University. She is a AAAI Fellow and Council Member. She may be contacted at: Knowledge Systems Laboratory, 701 Welch Road Building A, Palo Alto, CA 94305.