# Planning and user interface affordances

Robert St. Amant
Department of Computer Science
North Carolina State University
EGRC-CSC Box 7534
Raleigh, NC 27695-7534
stamant@csc.ncsu.edu

**ABSTRACT**

This paper takes a first step toward formalizing the concept of affordance in user interfaces. Using a simple example of an AI planning domain, we show how different types of affordance can be described in terms of the costs associated with plan execution. We identify a number of similarities between executing plans and interacting with a graphical user interface, and argue that affordances for planning environments apply equally well to user interface environments. We support our argument with examples of common user interface mechanisms, described in affordance terms.

**KEYWORDS**

Planning, affordances

**INTRODUCTION**

Most of us are familiar with Gibson's notion of affordance: "The affordances of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill" [7, p.127]. An affordance is an ecological property of the relationship between an agent and the environment. For example, we hold a pencil in such a way that it fits comfortably in the hand, ignoring the myriad less appropriate ways that it might be grasped. The pencil affords being held in this way as a result of its length, width, weight, and texture, all with respect to the size, configuration, and musculature of our hand. Further, we can *see* most of these properties and relationships; we can often tell how to interact with an object or an environmental feature simply by looking at it, with little or no thought involved.

The concept of affordance has gained wide currency in the literature of human-computer interaction. Unfortunately, if we are interested in well-founded models (cognitive, perceptual, or even physical models) of user interaction, the intuitive nature of the concept works against us. What exactly *is* an affordance? Several competing answers to this question have arisen, but no consensus [22, 8, 21, 1, 23]. Part of the difficulty is that affordances are closely tied to perception and action in the physical world, and we have no general theories of action and perception powerful enough to give us guidance.

A common formalism in AI and HCI research, the problem space representation, offers a partial solution to this problem. This paper examines a restricted version of affordance in an abstraction of the real physical world. The logic of the paper is as follows. We begin with a description of planning and its roots in a problem space representation of states and actions in the world. We give a brief overview of the various definitions of affordance that have appeared in the literature, and show how they can be interpreted within a planning framework. We then identify a number of striking but not entirely unexpected similarities between research in planning and user interface design, and use this close relationship to argue that our account of affordances for plan execution applies equally well to affordances for interaction with a user interface. We round out our argument with several examples of mechanisms in existing user interfaces and research prototypes that can be interpreted as providing affordances for various types of user interaction.

The work presented in this paper is theoretical, in the sense that we do not present an implementation and we do not show how the ideas contribute to improved design (though we discuss both points briefly in the conclusion.) Instead, we address a more basic point: nowhere in the literature of user interface affordances can we find an unambiguous, concrete definition of the concept. Affordance remains vague, disconnected from the wide variety of theoretical HCI frameworks in areas such as layout design, task analysis, human information processing, and so forth. With this work we have attempted a first step toward our long-range goals of providing a rigorous foundation for modeling, developing, and evaluating affordances in the user interface.

**PLANNING AND PROBLEM SPACES**

Newell and Simon define a problem space as containing a set of elements, $U$, a set of operators, $Q$, an initial state of

```
(:operator pickup
           :parameters (?x ?loc)
           :precondition (:and (block ?x) (location ?loc)
                               (hand-empty) (at ?x ?loc) (at agent ?loc))
             :effect (:and (:not (hand-empty)) (holding ?x)))
```

Figure 1: A sample planning operator

knowledge, $u_0$, a problem with goal $G$, and the knowledge available to solve the problem [17]. A solution to a problem involves reasoning from the state $u_0$ to the state $G$ by constructing a sequence of appropriate operators from $Q$. By reasoning about problems in an abstract formalism, a human or automated system can focus on the logic of problem solving instead of on less relevant details.

Planning is a kind of search appropriate for reasoning in problem spaces [11]. For conventional partial order planners, the set $Q$ contains *STRIPS operators*. STRIPS operators have a precondition and an effect, both of which specify properties (logical predicates) that hold or potentially hold in the world. An operator may be applied only when its preconditions hold in the current state. Its application results in the creation of a new state in which the effects of the operator hold. A plan is a partially ordered sequence of operators that, when applied, lead from $u_0$ to $G$.

Figure 1 shows one operator from a familiar planning domain, expressed in the language of the UCPOP planner [20]. The `pickup` operator gives an agent the ability to pick up a block. The precondition of the operator specifies that it can only be executed when the agent and the object are in the same location, the object under consideration actually is a block, and the agent's hand does not already hold something. The effect specifies that, once the operator has been executed, the agent will be holding the object.

This representation, like most that appear in the theoretical planning literature, abstracts away most of the details of interaction with the real world. For a planning agent, this abstraction can make a large difference between generating a plan of action and actually executing it, even for an agent with all the appropriate sensors and effectors. Bridging the gap between plan generation and execution runs into difficulties for two reasons [2]. First, a planning agent may not have complete control over the environment. Other agents can interfere with the planner's activities; time passes and the environment may change dynamically while the agent acts. Second, the agent may not have enough information to perfectly predict the effects of all actions, notably in the case where actions involve sensing. This uncertainty means that some plans cannot be guaranteed to succeed in advance of execution.

These observations have implications for the cost of executing a plan. We associate with each operator a cost, which may be determined analytically or empirically, and may draw on information not explicit in the symbolic operator representation. This cost could include the duration of the operator, the difficulty encountered, the probability of failure, the quality of the effects, and other factors. For simplicity in the discussion that follows, we will assume that cost is expected duration. Note that we cannot use a deterministic measure of cost; even if an agent has constructed a complete and correct plan for its goals in the problem space, its cost may vary randomly because of the environmental control and uncertainty factors. Our account of affordance will rely heavily on this point: affordances for a planner (or for any problem space reasoning system) are concerned with reducing the cost of translating a problem space solution into the real world.

## AFFORDANCES

Affordance in the physical world is an intuitive notion, easily described and understood through examples. Like many such concepts, however, it is difficult to define in precise analytical terms. Imagine yourself in the act of sitting down in a chair. There are at least four separate affordance-related concepts involved. First are the affordances proper: the seat of the chair is horizontal, flat, extended, rigid, and approximately knee-high off the ground, all relative to your own proportions and position [7, pp. 128-9]. Second is your perception of these properties, the surfaces, distances, areas, textures, relationships between parts, and so forth. Third is the mental interpretation you derive from the perceptions. Fourth and finally is the act of sitting itself. An examination of these positions will give a better understanding of the subtleties involved.

*Affordances are relationships or properties of relationships.* We begin with Gibson's original concept, that affordances are ecological properties, intrinsic properties of the relationship between an agent and its surrounding environment [7, p. 143]. Many ecological researchers rely on this definition, with one notable specialization: an emphasis on affordances for action. Kirlik et al. define an affordance as "a relationship between properties of the environment and properties of an organism's action capabilities" [10, p. 934]. Static relationships between an agent and its environment, such as support or containment, are of less interest than the active behavior that the environment affords the agent, such as walking or sitting [15]. For researchers and designers interested in issues beyond perception, this is a natural shift of focus.

*Affordances are actions.* Some researchers work at a level of abstraction in which the distinction between affordances and actions is blurred. Mark et al., in testing the ability of human subjects to perceive whether a seat can be sat on, define affordances as "those actions which the particular arrangement

of environmental substances and surfaces will support" [15]. In a similar vein, Gaver describes affordances as "potentials for action" [6, p.79]. It is clear that these authors are using "action" and "potential for action" as a kind of shorthand, but it suggests that distinguishing between action and affordance may not always be necessary.

*Affordances are perceived properties.* Baecker and Buxton write, "Affordances are the perceived properties of an artifact that indicate how it can be used" [4, p. 1]. This view, popularized by Norman [18], has become so widespread in HCI circles that now perceptibility is often taken to be an integral part of an affordance. While this results in an important and useful notion—that designed, perceivable affordances can directly influence usability—a complete change of focus is resisted by some ecological researchers. As Flach writes, "[This] confuses the affordances of an object with the information that specifies the affordances" [5].

*Affordances are mental constructs.* A final view is that affordances are subjective in nature. Vera and Simon see affordances as "internal representations of complex configurations of external objects" [22, p. 41]. Gibson's original definition holds that affordances are objective, measurable properties in the external world. Vera and Simon's view does not deny the existence of "external" affordances; for example, Newell and Simon observe that "adaptive devices shape themselves to the environment in which they are embedded" [17, p. 789]. Instead, their position is that for regularities in the world to be perceptible by an agent, some change in mental representation must occur, and it is the resulting mental structure that deserves to be called an affordance. These mental affordances are the internal encodings of symbols denoting relationships, rather than the external situations that evoke the symbols [23].

## AFFORDANCES FOR PLAN EXECUTION

These affordances exist for all agents, not only human ones. Thus we can speak of the affordances an environment provides an autonomous agent such as the one we discussed in connection with Figure 1, executing its plans to pick up blocks. Let's assume that we have a planner with a full complement of operators beyond `pickup`, along with appropriate sensors and effectors, and that it can construct complete plans for its goals. What can we say about the affordances that exist for it? Can we reconcile the views in the previous section?

The planning representation easily captures one view of affordance, which we can express as follows:

> A *simple affordance* for operator *A* exists if the environment can reach a state in which *A*'s precondition holds.

Preconditions determine the applicability of operators. If an environment is to afford a specific action, then by definition it must be possible to execute the action at some time in the environment. This corresponds directly to the view of affordances as actions or potentials for action. In any reasonable example of our blocks domain, picking up the blocks is simply afforded.

If we consider the cost of operator execution, we encounter a version more interesting than simple affordance. Cost can vary for executing operators, due to uncertainty and incomplete control of the environment. Execution affordances address this point:

> A *precondition execution affordance* for operator *A* is a property or mechanism whose presence decreases the cost of operators that establish or preserve predicates in *A*'s precondition, or increases the cost of other operators that do not.

In other words, a precondition execution affordance facilitates an operator by reducing the cost of making it applicable, either relative to other operators or relative to the environment that would obtain in the absence of the affordance. This definition reflects the view that affordances are action-specific relationships between an agent and its environment. In our blocks domain, the agent must be in the same location as the block in order to pick it up. Suppose that our agent is on wheels, and has a tendency to roll randomly away from any location it finds itself in. A braking mechanism would provide a precondition execution affordance for `pickup` by reducing the cost of preserving the `(at agent ?loc)` predicate. For a property that plays a similar role, imagine what would happen if blocks were spheres instead of cubes. The flatness of a block provides an affordance for `pickup` by helping ensure that it remains wherever it is placed, preserving the `(at ?x ?loc)` predicate.

The notion of an execution affordance extends to predicates in the effect of an operator as well:

> An *effect execution affordance* for operator *A* is a property or mechanism whose presence decreases the cost of establishing or preserving predicates in *A*'s effect.

At some level of detail in a problem space representation we encounter physical activity that is not modeled, activity that results in the state change described by the effect of the operator. Effect execution affordances reduce the cost of this unmodeled, within-operator activity. In our example domain, the effect of a `pickup` includes the predicate `(holding ?x)`, where `?x` is a block. Effect execution affordances include the match between the size, shape, weight, and texture of the block and the properties of the agent's gripper. Like precondition execution affordances, effect execution affordances reflect a view of affordances as relationships.

Another important aspect of affordance is its relationship to perception. For our planning agent, some predicates in its operators are meant to be established through other operators, while others have fixed values that must be detected (some can be handled either way.) For example, the agent

| Planning assumptions | User interface design guidelines |
|---|---|
| Deterministic effects. The effects of an action can be guaranteed to hold after its execution. | Consistency [3]. A user action under a well-defined set of conditions should always produce the same effect, the goal being predictable behavior. |
| Extraneous events. The only changes in the environment are due to actions of the planner. (This ideal of user/planner control is so well-entrenched that in both planning and HCI the terms "action" and "event" are used interchangeably [9].) | User control [3]. From the conventional (i.e., non-agent-based) perspective, user control over the interface should be nearly absolute. The interface does not initiate actions, but rather responds like a tool [25]. |
| Passage of time. Plans remain appropriate in the interval between their generation and their execution. | Perceived stability [3]. In conventional user interfaces, the user should be able to depend on stability (e.g., visual layout) as time passes. |
| Reversible actions. When interleaving plan generation and execution, irreversible actions can be heuristically ignored or delayed to avoid dead-ends. | Forgiveness [3]. Actions should generally be reversible through an undo capability. |
| State information. Complete information is available about the initial state of the world. | Continuous representation. Objects and actions of interest should be continuously visible. |
| Concurrent actions. Operators are atomic and non-overlapping; no parallel execution. | Serial execution. User attention and decision-making focuses on one task at a time. |

Table 1: Planning assumptions and user interface guidelines

cannot make an object a block in order to pick it up, but must sense the value of the `block` predicate for the object. This distinction gives rise to another type of affordance:

> A *precondition evaluation affordance* for operator *A* is a property or mechanism whose presence decreases the cost of determining the values of predicates in *A*'s precondition.

In simpler terms, a precondition evaluation affordance makes it easier to decide whether an operator can be applied. In our example, the agent must establish that an object has the `block` property before it can apply the `pickup` operator. While at plan generation time this may be a simple look-up, at plan execution time it entails the cost of sensing and interpretation to establish the status of the object. A precondition evaluation affordance, for some types of robotic agents, might be a radio beacon placed on top of an object, broadcasting a "block" signal. For other agents, visual properties such as shape or perhaps distinctive coloring provide the affordance. This type of affordance reflects the view that affordances are perceived properties.

We can also consider the symmetrical case of evaluation affordances for predicates in the effect of an operator:

> An *effect evaluation affordance* for operator *A* in a given state is a property or mechanism whose presence decreases the cost of determining the values of predicates in *A*'s effect.

Here we encounter a concept already familiar under a different name—feedback. Recall that operators may fail for reasons outside the agent's control. An effect evaluation affordance makes it easier to determine whether a desired goal has actually been reached by the execution of an operator. For the `pickup` operator, an agent might have visual feedback or tactile pressure feedback to indicate that its `holding` predicate is in place.

These four types of affordance, plus simple affordance, allow us to describe the relationship between an agent and its environment in ecological terms. Our example is purposely drawn from a simple domain for the sake of illustration, and because we have a relatively good understanding of the underlying costs of the interactions. To evaluate our definitions we can ask how well they match our understanding of affordances in the physical world. The implications of our definitions of types of problem space affordances generally match our expectations for physical affordances.

*Affordances are about action.* Our affordances are exclusively concerned with plan execution, reflecting the focus of most researchers on affordances for actions rather than static relationships.

*Affordances are relative.* If a physical object or environment affords an activity, it is rarely a binary affair: for example, stairs may be climbable, but with more or less difficulty. This relative property is inherent in our interpretation of affordances in terms of cost.

*Affordances involve tradeoffs.* In our representation, an affordance that reduces the cost of some predicate may increase the cost of operators that depend on conflicting predicates. Similarly, in the physical world, artifacts and tools are tailored for their intended uses, and true cross-purpose generality is less common.

*Almost any feature of the environment affords some arbitrary action.* This generality may be interpreted as

a flaw of our interpretation, but we see it instead as inherent to the general nature of physical affordance. This highlights the need for appropriate modeling decisions in constructing a set of operators. Affordances exist, as Gibson points out, for good or ill. It is our task to build affordances into the environment that facilitate the activities intended by our design.

We take these observations as a general assurance that we are on the right track. Let's now turn to the user interface.

## PLANNING AND THE USER INTERFACE

The fields of planning and user interface design share a common history of research, the most prominent ancestor of both fields probably being Newell and Simon's *Human Problem Solving* [17]. Strong similarities link the two areas, most significantly in their common reliance on problem space representations.

For planning researchers, the differences between a problem space solution and a real world solution are expressed in the assumptions a planner needs to make at plan generation time. The basic forms of most theoretical planning algorithms in use today (e.g. UCPOP, UMCP, PRODIGY, GRAPHPLAN, SATPLAN) ignore the passage of time, uncertainty in the effects of actions, changes in the environment, actions of multiple agents, and other issues [24], as discussed earlier. The assumptions listed in the left column of Table 1 give us some notion of what might constitute an ideal world for an autonomous planner.

What does this ideal world look like? It bears a striking resemblance to the modern graphical user interface. Designers have developed a number of guidelines for building effective user interfaces, and they overlap significantly with planning assumptions, as seen in the right column of Table 1. These correspondences are not accidental. They are derived from the same basic research framework, which holds that a problem space representation is an appropriate and effective means of modeling human behavior. The application of this precept can be seen directly in many areas, such as GOMS research. Even the distinction between plan generation and plan execution has a counterpart in HCI, as the gulf of execution. The gulf of execution refers to the difficulty of acting through the interface, of matching one's intentions to the actions that must be carried out [12, 19]. The related gulf of evaluation describes the difficulty of determining whether goals have been met. This conceptual breakdown is deliberately reflected in our affordance terminology.

Broadly speaking, planning researchers and user interface designers see opposite sides of the same coin. Planning researchers cannot insist that the environment accommodate itself to the limitations of a planning system; they must concentrate on building better planners, reducing their dependence on assumptions. User interface designers, on the other hand, cannot improve human users, but they *can* alter the environment to reduce potential mismatches between its properties and the abilities and limitations of the users. We use the close relationship between planning and user interface design considerations to argue that the types of affordances we identified for a planning agent in the previous section apply directly to the actions of human users.

Figure 2 shows part of a set of operators for interacting with a graphical user interface. Applying these actions, a user can move from one icon to another, clicking for selection, double-clicking for activation, dragging (with an unshown `Drag` operator), and so forth. The types of affordances for the execution of an operator in this domain are identical to those of any comparable problem space representation.

Consider the task of moving the pointer until it is over an icon and then clicking on it. This is a sequence of operators rather than a single one, but it is otherwise very similar to our earlier `pickup` example. Although clicking on an icon is one of the commonest tasks we encounter in a graphical user interface, our performance is surprisingly unreliable over the long term. In a study of the application of Fitts' Law to performance using different input devices, for example, MacKenzie found that the a small but noticeable proportion of their data was unusable due to pointing errors: "Unadjusted error rates for pointing were in the desired range of 4% with means of 3.5% for the mouse, 4.0% for the tablet, and 3.9% for the trackball" [14, p. 164]. A later study found a mean error rate of 1.8% for pointing with the mouse [13]. Worden et al. found a mean error rate of 5% (for both young and old users) in a study that directly examined targeting difficulties with the mouse [26]. We can account for these difficulties, and explain why suggested improvement mechanisms work, in terms of our affordance definitions.

**Precondition and effect execution affordances.** In some cases an icon selection attempt fails because of the difficulty in establishing or maintaining the `pointer-over` predicate in the precondition of the `Mouse down on` operator. A number of different affordances can be added here:

*Icon size:* Fitts' law tells us that increasing the width of an icon will reduce the time to reach it, thus reducing the cost of establishing the `pointer-over` predicate.

*Sticky icons [26]:* When the pointer is over an icon that is sticky, the gain of the mouse is reduced, requiring an increased effort for the user to move the pointer off the icon.

*Haptic feedback [16]:* On predicting the target of a user's movement, one implementation of a haptic mouse activates an electromagnetic field to stop the mouse on the target (and thus the pointer over the icon), eliminating the possibility of overshooting and the need for detailed adjustment.

*Capture effects:* If a mouse down is detected in the neighborhood of a selectable icon, a system can to interpose an additional movement that puts the pointer over the icon, effectively defining an area of "magnetism" around it. Selection of narrow or small objects such

```
(:operator |Mouse down on|
           :parameters (?object)
           :precondition (and (not (mouse down))
                              (pointer-over ?object)
                              (not (obstructed ?object)))
           :effect (and (mouse down)
                        (when (not (eq ?object background))
                          (object-activated ?object))))
(:operator |Mouse up on|
           :parameters (?object)
           :precondition (and (mouse down) (pointer-over ?object))
           :effect (and (not (mouse down))
                        (when (object-activated ?object)
                          (and (not (object-activated ?object))
                               (when (object-clickable ?object)
                                 (object-clicked ?object))
                               (when (not (object-moved ?object))
                                 (object-selected ?object))
                               (when (object-moved ?object)
                                 (object-dragged ?object))))))
(:operator |Move|
           :parameters (?source ?target)
           :precondition (and (not (mouse down))
                              (pointer-over ?source))
           :effect (and (not (pointer-over ?source))
                        (pointer-over ?target)))
. . .
```

Figure 2: Operators

as line segments in graphical drawing interfaces often rely on such a mechanism.

*Area cursors [26]:* The area cursor has a "hot spot" larger than the conventional single pixel, which provides a greater area for intersection with selectable objects.

Of course, a number of static properties already induce precondition execution affordances: the mobility of the mouse, the corresponding mobility of the pointer, the existing size of the icon, and so forth.

Let's consider another potential problem. The `pointer-over` predicate may hold for the `Mouse down on` operator, but fail for the `Mouse up on` (i.e., the user may inadvertently move the pointer away from the icon in focus, indicating a cancelation of the selection gesture.) The affordances above can be implemented for this case, and restricted, potentially less obtrusive versions are also possible:

*Down sticky icons [26]:* When the pointer is over an icon that is sticky *and* the mouse button is down, the gain of the mouse is reduced.

Dragging can also be facilitated by these mechanisms. Dragging is in principle identical to pointing, though differently parameterized models may be necessary to describe the activities accurately [14]. These affordance mechanisms can potentially be much more beneficial, however, because dragging is more difficult than pointing. In the same studies cited earlier, MacKenzie et al. found error rates for dragging with different pointer devices to be 10.8% for the mouse, 13.6% for the tablet, and 17.3% for the trackball [14]. Their later study found a mean error rate 4.2% for dragging with the mouse [13].

One difference in difficulty between pointing and dragging can be attributed to releasing the mouse button accidentally, as can easily happen when the mouse button is held down for an extended period. This slip can be expensive in terms of recovery time: an object may need to be retrieved from a container; one's place may be lost in scrolling through a large document; a large multiple selection may need to be redone. Our operator representation suggests a simple, novel affordance: a mechanism should be provided to allow the user to recover the lost state in which the object was grabbed.

*Unclick:* If a `Mouse up on` operator is immediately followed by a `Mouse down on` operator (i.e., if the interval between the two events is less than some small value), then the sequence should be ignored, and the earlier state restored.

Unclicking is an affordance in that it allows restoration of the `(mouse down)` predicate, a precondition for the `Mouse up on` operator. This must be done as an "undo" mechanism rather than a simple delay to avoid a time penalty for

ordinary drag-and-drop actions, but this is a plausible and potentially useful extension to existing functionality.

For examples of effect execution affordances, consider the `Move` operator, which shares a `pointer-over` predicate with `Mouse down on`, though in its effect rather than its precondition. The affordances above do double duty here as effect affordances. Other effect affordances not specifically related to the `pointer-over` predicate include variable mouse gain, which can reduce the cost of long movements in general, and mouse warping, which in some contexts can eliminate the cost of mouse movement altogether.

Notice that if we change the operators in our planning domain (say, to have a `Single click` operator instead of `Mouse down on` and a `Mouse up on`) the classification of a mechanism as a precondition affordance may change to an effect affordance. This potential ambiguity is an unavoidable aspect of our interpretation of affordance; in general, no representation of the real world has a privileged status.

**Precondition and effect evaluation affordances.** Precondition evaluation affordances are an essential aspect of a visual interface. Consider a button icon in the interface, and a `Press-button-icon` operator with a precondition containing `(button ?object)`. The cost of this operator includes the cost of deciding whether the icon is a button. Is it button-like, with a raised surface and a distinct, generally convex border? If we were to examine a series of unconventional button icons that shared fewer and fewer recognizable properties with standard button icons, we would find that the cost of their recognition would gradually increase, reducing the precondition evaluation affordance for the `Press-button-icon` operator. Other interactive components of the interface support dynamic precondition evaluation affordances: in Windows, for example, moving the pointer over a grabbable window decoration (e.g., a grow region) causes the pointer to change appearance to reflect the operator that becomes applicable.

Effect evaluation affordances, or feedback mechanisms, are ubiquitous. Pressing a command accelerator keystroke may cause the corresponding menu title to blink, indicating that an operator has been executed. The selection of a graphical object changes its appearance. These and other sorts of feedback reduce the cost of determining whether an action has been successful, so that the user can move to the next task.

Evaluation affordances need not depend exclusively on reducing the cost of unmodeled mental activity. An important element of the direct perception view of user interaction is that properties of an artifact be discoverable through exploration [5]. Thus an interface can ameliorate the lack of a "pressability" affordance for an unconventional button icon by arranging for the mouse cursor to change shape when it passes over the icon. To determine the value of of the `button` predicate for such an icon requires some exploratory activity on the part of the user, with its associated costs, but it may contribute to an overall reduction to the evaluation cost for the operator.

## DISCUSSION

We could continue by building a compendium of user interface mechanisms and the operators they afford, but the application of the concepts should be clear. We are currently extending our framework to affordances for plan operators at a higher level of abstraction, for intelligent assistants in the areas of language learning and statistical data exploration. One benefit of the framework, though it is too early to document in detail, is that it has helped focus our attention on opportunities for improvements in the interface. For our data exploration application, for example, we have developed mixed-initiative plans to represent explicit statistical strategies. In order to guide the user along the paths suggested by the assistant, the interface dynamically introduces evaluation and execution affordances for its preferred operators. These mechanisms include the visually highlighting of suggested operators, the autonomous generation and display of intermediate results, annotating a "roadmap" for navigation, and related activities. Simple affordances for the standard user-selectable operations remain unchanged.

So far in this discussion we have neglected one of the most obvious properties of a planning system: the ability to generate and execute plans. Once we develop a set of operators, a planner can execute them in a real or simulated interface to produce an estimate of the quantitative benefit of an affordance. This has a few novel advantages over the obvious alternative of usability testing with human participants, in that the planner as a simulation can be tuned, if necessary, unlike human users, it can be executed at low cost and high volume, it can generate precise and repeatable quantitative measurements, and in some cases it can provide an analytical connection between a user interface mechanism and an explicit representation of knowledge in operator form. We have explored these possibilities, running the operators of Figure 2 in a Garnet-based user interface to evaluate a few simple precondition execution affordances. Results have been promising. One serious drawback, however, is the difficulty of establishing the adequacy of the planner and its set of operators as a realistic model of human behavior.

This leads to our final point, that this work has severe limitations. We have chosen the simplest workable representation for our planner and its user interface operators. Two clear areas for improvement are operator representation and cost modeling. First, users only rarely consider the buttons and other icons in an interface explicitly; instead, they treat the interface as a window onto a larger work environment, flexibly creating abstractions on the fly and shifting between problem-solving viewpoints. Our representation must capture such flexibility in order to move forward. Second, while our cost models are straightforward for execution affordances, they are almost non-existent for evaluation affordances. Assuming the existence of an operator for recognizing a button icon—and even giving it an expected duration—is far too simplistic to work in the long run. Despite these limitations, we believe that this approach can provide clarity

in the application of affordance concepts to the user interface.

**ACKNOWLEDGMENTS**

This research is supported by ARPA/Rome Laboratory under contract F30602-97-1-0289. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copright notation hereon.

**REFERENCES**

[1] P. E. Agre. The symbolic worldview: Reply to Vera and Simon. *Cognitive Science*, 17:61–69, 1993.

[2] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning*. Morgan Kaufman, 1990.

[3] Apple Computer, Inc. *Macintosh Human Interface Guidelines*.

[4] Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, and Saul Greenberg, editors. *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann, second edition, 1995.

[5] John Flach. The ecology of human machine systems: A personal history. In John Flach, Peter Hancock, Jeff Caird, and Kim Vicente, editors, *Global perspectives on the ecology of human-machine systems*, pages 1–13. Lawrence Erlbaum, 1995.

[6] William W. Gaver. Technology affordances. In *Proceedings of the CHI'91 Conference*, pages 79–84. Association for Computing Machinery, 1991.

[7] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[8] J. G. Greeno and J. L. Moore. Situativity and symbols: Reply to Vera and Simon. *Cognitive Science*, 17:49–59, 1993.

[9] James Hendler, Austin Tate, and Mark Drummond. Ai planning: Systems and techniques. *AI Magazine*, pages 61–77, Summer 1990.

[10] A. Kirlik, R. A. Miller, and R. J. Jagacinski. Supervisory control in a dynamic and uncertain environment ii: A process model of skilled human environment interaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:929–952, 1993.

[11] Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[12] Michael Lewis. Designing for human-agent interaction. *AI Magazine*, 97(2):67–78, Summer 1998.

[13] I. Scott MacKenzie and William Buxton. The prediction of pointing and dragging times in graphical user interfaces. *Interacting with Computers*, 6:213–227, 1994.

[14] I. Scott MacKenzie, Abigail Sellen, and William Buxton. A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of the Human Factors Society*, pages 161–166, 1991.

[15] Leornard S. Mark, James A. Balliett, Kent D. Craver, Stephen D. Douglas, and Teresa Fox. What an actor must do in order to perceive the affordance for sitting. *Ecological Psychology*, 2(4):325–366, 1990.

[16] Stefan Münch and Rüdiger Dillman. Haptic output in multimodal interfaces. In *Proceedings of IUI'97*, pages 105–112, Orlando, FL, January 1997. ACM Press.

[17] Allen Newell and Herbert Simon. *Human Problem Solving*. Prentice Hall, 1972.

[18] Donald A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[19] Donald A. Norman. Cognitive artifacts. In John M. Carroll, editor, *Designing interaction: psychology at the human-computer interface*, pages 17–38. Cambridge University Press, 1991.

[20] J. Penberthy and D. S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, pages 103–114. Morgan Kaufmann Publishers, Inc., 1992.

[21] L. Suchman. Response to Vera and Simon's situated action: A symbolic interpretation. *Cognitive Science*, 17:71–75, 1993.

[22] A. H. Vera and H. A. Simon. Situated action: A symbolic interpretation. *Cognitive Science*, 17:7–48, 1993.

[23] A. H. Vera and H. A. Simon. Situated action: Reply to reviewers. *Cognitive Science*, 17:77–86, 1993.

[24] Dan S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, Winter 1994.

[25] D. D. Woods and E. M. Roth. Cognitive systems engineering. In Martin Helander, editor, *Handbook of Human-Computer Interaction*, pages 3–43. North-Holland, 1988.

[26] Aileen Worden, Neff Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use: Area cursors and sticky icons. In *Proceedings of the CHI'97 Conference*, pages 266–271, 1997.