User interface affordances in a planning representation

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

*St. Amant is a computer scientist with an interest in artificial intelligence and complex interactive systems; he is an Assistant Professor in the Department of Computer Science at North Carolina State University.

Abstract

This article shows how the concept of affordance in the user interface fits into a well-understood artificial intelligence (AI) model of acting in an environment. In this model AI planning research is used to interpret affordances in terms of the costs associated with the generation and execution of operators in a plan. We motivate our approach with a brief survey of the affordance literature and its connections to the planning literature, and then explore its implications through examples of common user interface mechanisms described in affordance terms. Despite its simplicity, our modeling approach ties together several different threads of practical and theoretical work on affordance into a single conceptual framework.

Contents

1 INTRODUCTION

An affordance is an ecological property of the relationship between an agent and the environment: "The affordances of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill" (Gibson, 1979, p.127). 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 deliberation.

The concept of affordance has gained wide currency in the literature of human-computer interaction. Interface designers are encouraged to provide cues in their environments that indicate how objects can be used, in order to improve ease of use, reduce the need for instructions, and enhance familiarity with the interface (Dix et al., 1998; Eberts, 1994; Preece et al., 1994). Unfortunately, current explanations of user interface affordances are incomplete. Most rely almost exclusively on examples in their description of the concept, appealing to our intuitions about well-designed artifacts in the physical world. While an informal understanding may be sufficient to motivate the development of user interface guidelines, such understanding is no substitute for a well-founded model (whether cognitive, perceptual, or even physical) of affordances in the user interface.

Our long-term goal is to develop a rigorous foundation for modeling, designing, and evaluating affordances in the user interface. The goals of the work presented in this article are considerably more modest: we show how the concept of affordance fits into a traditional, well-understood model of acting in an environment, and we examine the implications of such descriptive models for the development and evaluation of low-level user interface affordances.

Section 2 gives a brief overview of the various definitions of affordance that have appeared in the literature. There has been considerable disagreement about the nature of affordances, partly because affordances are closely related to perception and action in the physical world, phenomena for which we have only incomplete and conflicting theories. The user interface, however, is not the physical world. User interaction has long been modeled as search in a problem space, a significantly less complex environment (Card et al., 1983; Newell & Simon, 1972). In adopting a problem space representation of interaction, we are led to a relatively simple interpretation of affordances in terms of the costs associated with the execution of operators and the evaluation of relevant features of the environment. This interpretation has strong connections to research in artificial intelligence planning. Our exploration of these relationships in Section 3 leads to a small taxonomy of types of user interface affordances that represent common practice in user interface design and provide a reasonable match with our understanding of physical affordances.

We offer support for our modeling approach in two ways. In Section 4 we describe a number of common user interface mechanisms in terms of our affordance definitions. The restricted scope of these affordances for widget-level interactions is offset by our ability to evaluate their benefit in a straightforward way. We outline a quantitative method for evaluating user interface affordances, in simulation, as a precursor to (but not a substitute for) usability testing. In Section 5 we explore how our modeling approach addresses affordances at higher levels of abstraction, those that deal with problem-solving in a specific domain. We describe a decision support system for exploratory data analysis and explain how our interpretation of affordance influences its interface.

Finally Section 6 reviews the scope of the work. Because our interpretation of affordance relies on an abstract model rather than the real world, it does not capture all of the subtlety of the concept. The level of modeling abstraction also precludes direct guidance in the construction of user interface mechanisms. Further, the affordances we can model quantitatively at this point are related to low-level widget interactions. Nevertheless we believe that the approach takes a promising direction, tying together various threads of practical and theoretical work on affordance into a single conceptual framework.

2 PERSPECTIVES ON THE NATURE OF 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 (Gibson, 1979, 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 is the act of sitting itself. An examination of these positions gives 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 to the relationship between an agent and its surrounding environment (Gibson, 1979, p. 143). Many ecological researchers rely on this definition, with one notable specialization: an emphasis on affordances for action. Kirlik et al. (1993, p. 934) define an affordance as "a relationship between properties of the environment and properties of an organism's action capabilities." 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 (Mark et al., 1990). 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. (1990, p. 325), in testing the ability of human subjects to perceive whether a seat can be sat upon, define affordances as "those actions which the particular arrangement of environmental substances and surfaces will support." In a similar vein, Gaver (1991, p. 79) describes affordances as "potentials for action." It is clear that these authors are using "action" and "potential for action" as a kind of shorthand, but it suggests that it may be plausible to interpret affordances as actions.

- *Affordances are perceived properties.* Baecker et al. (1995, p. 1) write, "Affordances are the perceived properties of an artifact that indicate how it can be used." This view, popularized by Norman (1988), has become so widespread in HCI circles that 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—this complete change of focus is resisted by some ecological researchers. As Flach (1995, p. 6) writes, "[This] confuses the affordances of an object with the information that specifies the affordances."

- *Affordances are mental constructs.* A final view is that affordances are subjective in nature. Vera and Simon (1993a, p. 41) see affordances as "internal representations of complex configurations of external objects." 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 (1972, p. 789) observe that "adaptive devices shape themselves to the environment in which they are embedded." Instead, they believe that for regularities in the world to be perceptible by an agent, some change in mental representation must occur, and that 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 (Vera & Simon, 1993b).

These differing viewpoints have been discussed in the cognitive science literature, with little agreement about their potential integration (Agre, 1993; Greeno & Moore, 1993; Suchman, 1993; Vera & Simon 1993a, 1993b). Fortunately, this has not slowed practical application of the ideas.

The approach of Kirlik et al. (1993) is most similar in spirit to our own. They examined the supervisory and manual control of a simulated scout craft in a dynamic uncertain environment containing friendly and hostile agents. Tasks included locomotion, sighting, and searching. A central part of their study was the definition of affordance distributions. An affordance distribution is a multi-dimensional array, overlaying a map of the environment, in which numerical entries reflect opportunities for action. A high value in one entry of a locomotion affordance distribution, for example, indicates that the craft can easily move from that position in the environment. Lower values are found along the edges of obstacles and in cul-de-sacs. All values in an affordance distribution are constructed through analysis or simulation, and tuned if necessary during a model validation phase. To evaluate their system, Kirlik et al. developed process models that used the information in the affordance distributions to guide their behavior. Based on standard measures of performance, the process models generated behavior very similar to that of a human crew. Their results support the view that affordances can play a significant role in constraining and guiding behavior in a complex environment.

The Ecological Interface Design (EID) framework of Rassmussen and Vicente (1988; 1990) addresses the relationships between affordances, rather than affordances themselves. Their approach fits affordances into a means-ends hierarchical structure. They begin with a list of affordances identified by Gibson: survival, nurturing, falling-off, communicating, locomotion, sitting-on, swimming-over, support, bumping-into, walking-on, manufacture, getting-underneath, lifting, cutting, obstacle, sinking-into, carrying, graspable, copulation, standing-on, shelter, nutrition, drinking, breathing, and throwing (Vicente & Rasmussen, 1990). Intuitively, we tend to group these affordances by their similarity to one another. Cutting and lifting, for example, are more similar than cutting and locomotion. Locomotion, in turn, is more similar to communicating than to survival. A coarse decomposition classifies affordances into three categories: *how*, *what*, and *why*. For example, a surface might provide locomotion (*what*), with walking the means (*how*) and survival the abstract goal (*why*). A system in which affordances are directly related to one another, and in which these relationships can be directly perceived, gives an agent two significant properties: a way to concentrate on important problems in the system, and the information necessary to solve the problems. The hierarchy guides the agent from abstract overviews to the details of the environment, using specific relationships between levels in the hierarchy. The EID framework has shown itself to be a useful, practical tool (Christoffersen *et al.*, 1997).

A third direction is suggested by Vera and Simon's view of affordances as mental constructs. By identifying and categorizing invariants, an automated system might determine their relevance to its own actions, and thus capture the affordances of an environment (Kalish, 1993). Alterman et al.'s FLOABN system (1998) is probably the most mature example of this approach. FLOABN addresses the broader problem of pragmatic action, interacting with the everyday environment. In the context of representing affordances, Alterman et al. nevertheless offer useful insights into the relationship between affordances and cognition. FLOABN relies on two key observations: the everyday environment is designed to provide an agent with explicit guidance for appropriate interaction, and it is semi-permanent. These observations lead to a model of cognition in which an agent relies on memory to solve recurring problems, and exploits information in the environment to guide its selection of appropriate actions. This model is explicitly ecological, in contrast with a more common AI model in which agents are isolated, ahistoric, general-purpose problem solvers. FLOABN accommodates a view of affordance as mental constructs in that an agent's use of objects is facilitated by a familiarity with their properties and past skill acquisition. Other approaches rooted in AI and cognitive science are described in the literature of situated action (Agre & Chapman, 1987; Agre & Chapman, 1990; Suchman, 1987), robotics (Arkin et al., 1997; Schmill et al., 1998), machine learning (Cohen et al., 1996; Cohen et al., 1997; Oates & Cohen, 1996), and ecological psychology (Kalish, 1993). Alterman et al. (1998) give a good introduction to this area.

Note that these theoretical and practical efforts make little distinction between physical and user interface affordances. Some concentrate on physical affordances exclusively. Significant differences exist, however, between acting in the real world and interacting with a user interface. These differences can be exploited to construct a more specific description of affordances in the user interface.

Our account relies heavily on parallels between artificial intelligence planning and human-computer interaction. These two areas share a history of research, the most prominent common ancestor probably being the work of Newell and Simon (1972) in *Human Problem Solving*. One of the most obvious similarities is a reliance in both HCI and planning on the concept of a problem space. A problem space is an internal representation of a task environment, consisting of a set of elements U, a set of operators Q, an initial state of knowledge u_0, a problem with goal G, and the knowledge available to solve the problem. In Newell and Simon's view, problem solving (independent of task and human subject) takes place via search in a problem space. A solution to a problem involves reasoning from the state u_0 to the state G by constructing a plan, a sequence of appropriate operators from Q. For planning researchers a problem space reduces real-world tasks to a manageable set of relevant features; for HCI researchers it provides the basis for cognitive models of action in the user interface. A problem space can be as abstract or as detailed as necessary; in this article we concentrate on relatively low-level representations.

Consider a simple example from the AI literature. An agent, human or artificial, is given the goal of picking up a block. In a simple problem space formulation, achieving this goal requires that the block is within reach of the agent and that the agent's hand is empty. We can express this action as a STRIPS planning operator (Fikes & Nilsson, 1971; Penberthy & Weld, 1992), as in Figure 1.

FIGURE 1 ABOUT HERE.

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 when its precondition holds in the current state, and its application results in the creation of a new state in which the effect of the operator holds. While this particular problem space formulation is more familiar in planning than in HCI, clear similarities to GOMS (John & Kieras, 1996a; John & Kieras, 1996b; Kieras, 1988) and related task analysis methods exist. We might represent an icon selection task in a graphical user interface, for example, in closely analogous terms. The divergence between the HCI and planning perspectives is in the choice of a problem-solving agent, in one case a human being and in the other a computer program, but not necessarily in the problem-solving methods that are relevant.

Working in a problem space of this kind provides a number of benefits for a problem-solving agent (and for those who model problem solving.) For example, an agent needs to consider only those properties of objects and the environment that are explicitly specified. The agent can ignore the color of the block, its texture, its orientation, and so forth—it need only know that the object *is* a block, and nearby. Actions, both for sensing and for effecting changes in the environment, are perfectly predictable. There exist no other agents, no passage of time, and no extraneous events that might disturb the execution of the plan. These assumptions, which mainly eliminate consideration of the cost and uncertainty of acting, significantly simplify models of the problem-solving process (Allen et al., 1990; Weld, 1994).

Unfortunately, a problem space is not a panacea; in some or even most cases it only delays an important difficulty. A problem space solution must eventually be mapped to a real task in a real environment, at which point the clean abstractions of the problem space can become a liability. From an HCI perspective, the difference between solving problems in a problem space and acting in the real world is captured by the notions of the gulf of execution and the gulf of evaluation (Lewis, 1998; Norman, 1991; Norman & Draper, 1986). 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. The gulf of evaluation describes the difficulty of determining whether goals have been met. If we accept Newell and Simon's account of human problem solving, then these two gulfs describe the mapping from a problem space to the "real world" of the user interface. Building an effective interface means ensuring that the interface matches the user's perceptual, attentional, and cognitive abilities, such that neither gulf is too large to be crossed. Planning researchers face the identical problem: an agent may be able to generate a plan in a problem space (using STRIPS operators or some equivalent formalism) but be unable to execute it, due to the limitations of its sensors and effectors. Managing the difference between plan generation and plan execution has been an active area of research in planning for almost thirty years.

What exactly are the problems that a problem-solving agent faces, whether described in terms of gulfs of execution and evaluation, or plan generation and execution? HCI and planning have independently arrived at similar categories of problems, and we find a striking relationship between their methods for dealing with them. Classical AI planners assume that their actions are deterministic; interface design guidelines dictate that system responses be predictable. Classical planners assume that the only changes to the environment are due to the effects of their actions; direct manipulation principles argue against changes occurring in the interface that are not initiated by the user. These and other correspondences, shown in Figure 2, point to a pattern: planning researchers and HCI researchers concerned with user interface design 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 by 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.

FIGURE 2 ABOUT HERE.

3 AFFORDANCES IN PLANNING TERMS

We begin with a concrete, familiar example, the affordance of a button for being pressed. (Affordances can be much more abstract, but we delay discussion of this point until Section 5.) In the physical world, a button is usually raised from a background surface, with a distinct, visible border; its surface offers enough friction to prevent a fingertip from slipping off; its mechanical construction may make travel smoother when pressure is applied to its center rather than at the very edge; auditory and tactile feedback often accompany the successful completion of the action. In the user interface, button icons have comparable visual and auditory properties, with visual feedback substituting for tactile feedback (e.g. icon highlighting and dynamic changes to the mouse cursor.) We even see a real, physical affordance for clicking buttons in the user interface in the tendency of the mouse button to spring back to its “up” position. Mappings between the properties of physical objects and those of interface objects allow knowledge of real affordances to transfer via metaphor to the interface (Anderson, 1993; Carroll et al., 1988).

Consider a problem space representation, shown in Figure 3, for pressing a button icon in an interface. Pressing a button with this set of operators entails moving the mouse pointer from its current position to a position over the button icon, pressing the mouse button, and releasing it. The sequence of operators is generated and executed, the state of the environment changing appropriately. In order to manage the complexity of a realistic task, this representation abstracts away most of the detailed features referenced by the affordances described above. Visual, tactile, and interactive properties reduce to predicates such as `pointer-over`, `visible`, and `button`.

FIGURE 3 ABOUT HERE.

Each operator in our representation has an associated execution cost. This is an extension of the classical planning problem space, in which a planning agent searches for a sequence—*any* sequence—of operators that satisfies the goal it is given. By taking cost into account, an agent can search for the least expensive course of action that satisfies a goal. Cost values may be determined analytically or empirically, and may draw on information not explicit in the symbolic operator representation. This might include the duration of the operator, the difficulty encountered, the probability of failure, the quality of the effects, and other factors. For simplicity in our discussion, we will assume that the execution cost of an operator is its expected duration. Thus the execution cost of the `(pointer-over ?object)` predicate is the time it takes to move the mouse from its current location to a position over the object. We do not use a deterministic measure of cost, because 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 environmental uncertainty.

In addition to execution cost, we associate an evaluation cost with the determination of the values of specific predicates. Some predicates, such as `(pointer-over ?object)` and `(mouse-state down)`, become true as a result of the application of an operator. The values of other predicates, however, such as `(button ?object)`, are not changed by an operator. Rather, the problem is that their value may be unknown; the implicit operators that determine these values have an evaluation cost. As with execution cost, we will assume that evaluation cost is expected duration. The evaluation cost of the `(button ?object)` predicate is then the time it takes to recognize an icon as a visual representation of a button that can be pressed. For some predicates we must consider both types of cost, execution in the context of its establishment as part of the effect of an operator, and evaluation where its determination depends on user judgment.

At this point we can return to our discussion of the nature of affordances to see how the various definitions of physical affordances fit into our modeling approach as user interface affordances. We identify five distinct types, summarized in Figure 4 and discussed below.

FIGURE 4 ABOUT HERE.

The 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 point in the environment. This corresponds directly to the view of affordances as actions or potentials for action. In our user interface domain, the `Mouse down on`, `Mouse up on`, and `Move` operators are all simply afforded. The effect of each operator, executed in sequence, establishes the necessary precondition for the next operator.

If we take execution and evaluation costs of operators into account, a version more interesting than simple affordance arises:

> 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.

A simple example of a precondition execution affordance is the application of Fitts' law for the sizing and placement of controls. If a specific control is made larger, or can be positioned so that the average distance to reach it is decreased, then its duration cost will be reduced. In our set of operators, an increase in the size of a button or other icon introduces a precondition execution affordance for the `Move` operator, by reducing the cost of establishing its precondition, `(pointer-over ?object)`. Note that in a real user interface the affordance for `(pointer-over ?object)` is partly due to visual feedback, which counts as an evaluation cost: deciding whether the mouse pointer is over the object or not. A mechanism that reduces this evaluation cost also contributes to an execution affordance for the `Move` operator; execution affordances are not limited to manipulation of execution cost.

Another simple example of a precondition execution affordance can be seen in interactive drawing programs, in which the user can select fine lines and small points. This selection works even for objects that would ordinarily pose difficulties for an operator that relied on exact positioning of the mouse pointer. Instead, drawing programs relax the requirement that the pointer be exactly over small objects, effectively reducing the cost of establishing the `(pointer-over ?object)` precondition within the context of a `Mouse down on` operator.

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.

Like precondition execution affordances, effect execution affordances reflect a view of affordances as relationships. Consider the effect `(pointer-over ?target)` of the `Move` operator. The same mechanisms that provide precondition affordances for `Mouse down on` act as effect affordances for `Move`. In our simple domain it may be difficult to see the need for effect execution affordances as an an independent concept, but recall our robot agent of Figure 1, whose single `pickup` operator results in a block being held in its hand. The `holding` predicate in the effect of `pickup` depends on the match between the size, shape, weight, and texture of the block and the properties of the agent's gripper. Properties of these relationships can act as effect execution affordances. Like precondition execution affordances, these reflect a view of affordances as relationships.

Another important aspect of affordance is its relationship to perception. Some predicates in our operator set are meant to be established through through direct action, while others have fixed values that must be detected. This distinction gives rise to a new 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 operator set the `Mouse down on` operator has the precondition predicate `(button ?object)`. The cost of deciding whether a given object is a button icon depends on features of the icon not represented explicitly in the operator. These factors must instead be encoded in the evaluation cost function. Does the object look button-like, with a raised surface and a distinct, generally convex border? If we were to create a series of unconventional button icons that shared fewer and fewer recognizable properties with standard button icons, we would gradually be increasing the cost of their recognition, and thus reducing the precondition evaluation affordance for operators such as `Mouse down on`.

Note that an evaluation affordance need not depend only on reducing the direct evaluation cost of implicit perceptual operators. An important element of the direct perception view of user interaction is that properties of an artifact be discoverable through exploration (Flach, 1995). In modern graphical user interfaces, many icons give some visual indication that they are button icons (the mouse pointer changing shape, the icon becoming highlighted, or a small documentation window popping up, for example) whenever the mouse pointer passes over them. The predicate `(button ?object)` can be established in this way through the execution of operators, rather than direct evaluation of the predicate. This requires some exploratory activity on the part of the user, with its associated execution cost—and some evaluation cost is in the end inevitable—but the combination may contribute to an overall decrease in the cost of determining its value, and thus provide an evaluation affordance.

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. We ordinarily distinguish feedback from affordance, which argues against conflating the two ideas. Our goal is not to remove a useful distinction but rather to show how both concepts fit into the same framework. 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. An icon generally gives some visual indication that it has been selected, which creates an effect evaluation affordance based on the `(object-clicked ?object)` or `(object-selected ?object)` predicates of the `Mouse up on` operator.

These four types of affordance, plus simple affordance, allow us to describe the relationship between an agent and its environment in ecological terms. Our definitions for interface affordances cover the first three definitions of generic or physical affordances given in Section 2; modeling affordances as mental constructs, the fourth definition, entails learning new operators, which is beyond the scope of this work. Our examples are 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 in a broader context we can ask whether they give insight into the user interaction with an interface and whether they give guidance for interface design, issues we discuss in the next two sections. We can also ask how well these definitions match our understanding of affordances in the physical world.

Interaction with physical objects does not usually take the form of discrete problem space operators, as in a user interface. Nevertheless we see a strong parallel in the importance of cost and its ability to express the ease or difficulty with which physical actions are evaluated and executed. Consider a sample of affordances identified in the literature, as shown in Figure 5. In each of these cases, the authors identify an object or environmental feature and discuss the activities that it affords and which of its features produce the affordance. The features provide both evaluation affordances and execution affordances. For example, in an experiment that requires subjects to ascend a staircase with risers at a particular height, we can measure the effort involved, for the execution cost of the activity. Similarly, we can ask subjects to assess whether the risers allow stepping (rather than crawling), which involves a judgment that can be quantified: how often are they correct, and is the judgment correlated with the execution cost? Execution and evaluation costs can plausibly be combined to provide a measure of the affordance of a physical activity.

FIGURE 5 ABOUT HERE.

Our definitions of different types of interface affordance 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.* We have cast our definitions, except for simple affordance, in terms of an increase or decrease in cost. This corresponds naturally to the intuition that afforded activities are in some sense easier to perform, or easier to perceive as being possible to perform. The properties of an artifact afford some specific activity in that (a) their absence would make the activity more difficult, or (b) their presence facilitates the activity in comparison to alternative activities.

- *Affordances involve tradeoffs.* If we introduce an affordance by making it easier to achieve the precondition of some action, then we may make it more difficult to carry out other actions that have conflicting preconditions. This matches our intuitions about physical artifacts: the same property (heaviness) that gives a sledge hammer its affordance for a powerful swing, for example, simultaneously makes it inappropriate for more delicate tasks.

- *Affordances are mutual relationships.* This is clearly the case for physical affordances: whether one can sit on a chair depends both on the seat height and the length of one's legs. Mutuality is not emphasized in our representation, but is accommodated in two ways. First, the cost of an operator can vary depending on the agent performing it. Second, different agents may have access to different sets of operators. Different problem-solving agents may thus associate different affordances with the same environment.

- *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. Given some action, it is a simple matter to identify environmental features that facilitate the action. 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.

These points argue that our account of affordances in planning terms has some plausibility. More specifically, our consideration of planning cost brings us close to the traditional interpretation of user interaction as problem-solving behavior (Woods, 1991). Card et al.'s (1983, p. 27) rationality principle holds that "a person acts so as to attain his goals through rational action, given the structure of the task and his inputs of information and bounded by limitations on his knowledge and processing ability." Intuitively, the rationality principle tells us that an agent will choose the easiest route that results in the completion of the task it has set out to accomplish. Rationality is invariably formalized in terms of cost, which provides a strong link to planning research (Boutilier et al., in press).

4 GENERIC USER INTERFACE AFFORDANCES

In this section we show how our plan-based model of affordance can contribute to an understanding of low-level actions in a graphical user interface. We identify a number of straightforward affordances and explain how the planning representation provides for an initial evaluation.

Figure 6 shows an extended 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 or confirmation, dragging, and so forth. Consider the task of moving the pointer until it is over an icon and then clicking on it. Although this is one of the commonest tasks we encounter in a graphical user interface, our performance is not entirely reliable over the long term. In a study of the application of Fitts' Law to performance using different input devices, for example, MacKenzie et al. (1991, p. 164) found that 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." A later study found a mean error rate of 1.8% for pointing with the mouse (MacKenzie & Buxton, 1994). Worden et al. (1997) found a mean error rate of 5% (for both young and old users) in a study that directly examined targeting difficulties with the mouse. We can account for these difficulties and explain why suggested improvement mechanisms work in terms of our affordance definitions.

FIGURE 6 ABOUT HERE.

Let's begin with precondition and effect execution affordances. In some cases an icon selection attempt fails because of the difficulty (cost) of establishing or maintaining the `pointer-over` predicate in the precondition of the `Mouse down on` operator. A number of different affordances can counteract this difficulty:

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 (Worden et al., 1997): 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 (Münch & Dillman, 1997): 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 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 as fine line segments in graphical drawing interfaces often rely on such a mechanism.

Area cursors (Worden et al., 1997): The area cursor has a "hot spot" larger than the conventional single pixel, which provides a greater area for intersection with selectable objects.

Many static properties already induce precondition execution affordances: the mobility of the mouse, the corresponding mobility of the pointer, the unchanged size of the icon, and so forth.

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 a restricted version is also possible.

Down sticky icons: When the pointer is over an icon that is sticky *and* the mouse button is down, the gain of the mouse is reduced. Cancelation is still possible, by moving the mouse off the icon, but at a greater cost.

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. 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. (1991) 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. Their later study found a mean error rate 4.2% for dragging with the mouse (MacKenzie & Buxton, 1994).

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), and the earlier state included a grabbed object, 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. The system must distinguish an unclick from a double click—both of which involve the same mouse events—but can do so based on whether the gesture begins when an a object is grabbed or not. Testing for the continuation of the mouse being held down would also help. Further, all this must be implemented as an "undo" mechanism rather than a simple delay to avoid a time penalty for ordinary drag-and-drop actions. Despite these potential difficulties and the complications for system event handling, this is a plausible 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 domain (say, to have a `Single click` operator instead of `Mouse down on` and a `Mouse up on`) the classification of a mechanism may change from a precondition affordance to an effect affordance. This potential ambiguity is an unavoidable aspect of our interpretation of affordance. Our operators are part of a model of a user interface, but no specific representation of the real world has a privileged status in general.

Turning to evaluation affordances, we find that 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 some interfaces, for example, moving the pointer over a selectable window decoration (such as a grow region) causes the pointer to change appearance to reflect the operator that becomes applicable.

Effect evaluation affordances, or feedback mechanisms, are also ubiquitous. Pressing a command accelerator keystroke may cause the associated 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.

4.1 Programmable User Models for Affordance Evaluation

A given mechanism in the user interface may appear to provide an affordance in principle, but this may not hold in practice. Most of the low-level affordances above have gone through usability studies, even rigorous experimental testing. Testing is a necessary but unfortunately often expensive task. A novel interface affordance may have several parameters that need to be set correctly for its effective use.

Consider the example of a capture affordance for imprecise selection of icons. How far should its range extend? One, two, five pixels? Further, some types of affordance are meant to handle relatively rare events, what are sometimes considered user errors. If in a given scenario we expect an error in 5% of the cases, and an affordance may have five plausible settings for a parameter, then we may need hundreds or even thousands of trials with real users to distinguish these settings in evaluating the affordance. Even though the general desirability of an affordance mechanism might be established through a small number of informal user tests, gaining an understanding of its specific behavior can be much more time-consuming. What we would like instead is a way to map out the performance space of an affordance automatically. We can do this with a programmable user model, or PUM (Kieras & Meyer, 1997; Young et al., 1989). A PUM simulates a user, to some appropriate degree of accuracy, in order to provide a designer with feedback before taking on more extensive (and expensive) usability testing.

We have developed a PUM that exploits one of the most obvious benefits of a plan-based implementation of a problem space: the ability to generate courses of action automatically. The planner embedded in the PUM is based on Graphplan, a modern planner in wide use among planning researchers (Blum & Furst, 1997). The planner executes in an open loop mode: it generates a plan to satisfy a given top-level goal and executes it without responding to the success or failure of its operators. Once the entire plan has been executed, the planner checks whether any operators have failed, to indicate the failure of the entire plan. If this has happened, a new plan is generated based on the current state of the interface. It begins execution, and eventually the task is completed. In general this is a poor approach to interacting with the real world, but for some domains it reflects natural behavior. In the context of the operators in Figure 6, for example, users do not spend the time or effort to see whether the pointer is entirely within a widget before clicking. They point and click, and discover errors retrospectively.

With each operator in a generated plan we associate three methods with its execution: an execution method, which determines exactly what happens in the environment when the operator is executed; a duration method, which determines how long it takes; and a success method, which observes changes to the environment and evaluates whether these match the effect of the operator. These methods provide the user modeling component of the PUM.

The domain of the PUM is the operators in Figure 6, with appropriately defined methods. Duration of pointer movement is based on a well-known finding from the psychology literature, Fitts' law (Fitts, 1954). Researchers commonly use the following version (MacKenzie, 1995):

$$T = a + b \log_2 (A/W + 1). \tag{1}$$

T is the total time to move a limb (e.g., a finger, an arm, a foot, or a mouse) to a target W units wide and A units distant. The values of the constants a and b are determined empirically for a specific task, set here to $a = -0.107$ and $b = 0.223$ for pointing with the mouse (MacKenzie, 1995). Under Fitts' law conditions, subjects must make targeting movements that maintain high accuracy (a 95% target hit rate requirement is common) while minimizing the duration of the movements. To implement this as an execution method we make a few additional assumptions. First, we use the distance between the starting position of the pointer and the center of a target widget for A. Second, we assume that movement is in a straight line through the center, with no positioning error perpendicular to the axis of movement. This has the effect of changing a two-dimensional targeting problem into a single-dimensional problem amenable to Fitts' law analysis. Third, we assume a symmetrical, normal distribution of positioning error along the axis of movement. This assumption of normality is consistent with other modeling efforts for similar tasks and is common in most kinds of regression modeling, though it is not entirely accurate (Meyer et al., 1988). The mean of the distribution is the center of a target widget, and its variance is such that 95% of the distribution is contained within the borders of the widget along the axis of movement. This distance between the borders is then W. With A, W, and the constants given above, we can compute the duration of the mouse movement operator. Movement distance toward the target widget is modeled by sampling from the normal distribution. In accordance with Fitts' law conditions, accuracy (or error rate) does not depend on the distance from the target, but is fixed in advance.

The other operators in the domain have much simpler behavior. Each `Mouse down on` and `Mouse up on` operator takes 0.2 seconds (Eberts, 1994). If the composite click takes place on a target widget, there is no additional cost. If a click occurs on the background, however, the model takes a fixed 0.5 seconds to react to lack of the expected system response, and a further 0.2 seconds for the decision to try again (Card et al., 1983). If the click is on an incorrect widget, the penalty is an arbitrary 5.0 seconds, which represents recovery time.

The environment in which the PUM operates is the simple dialog window shown in Figure 7, a GARNET application (Myers et al., 1990), which can be manipulated in the conventional ways. A scenario begins with the pointer positioned at a random point on the list box, just after the selection of an item. Two tasks are defined, the first moving to and clicking on the "OK" button, the other moving to and clicking on the "Cancel" button. Each task involves only three distinct actions: `Move`, `Mouse down on`, and `Mouse up on`. Though simple, each task is subject to failure in two different ways: the pointer might end up on the background of the window or its border, where a click has no effect, or the incorrect button may be clicked. Difficulties with `Mouse down on` and `Mouse up on` operators do not arise due to the limitations of the PUM.

FIGURE 7 ABOUT HERE.

This problem might be amenable to analytical solution, but it is at the far edge of feasibility. In addition to the non-uniform structure of the interface and its varied responses to errors in positioning the pointer, an analysis must take into account the recovery actions necessary to complete the task. A longer sequence of actions, or the consideration of more varied affordances, would be enough to render the analytical approach impossible; we choose simulation for its extensibility.

4.2 Performance and Analysis of Programmable User Models

Our analysis concerns a simple capture effect that activates a button if a click occurs within some short distance of its border. Suppose we add a capture affordance to both of the buttons on the task interface. We let the ratio of confirmation tasks to cancel tasks be 20:1. What should the capture range be for each button for best performance? We measure performance in three different ways: the mean duration of the task of clicking on either button, the mean number of errors (misses) that occur, and whether a button ever incorrectly captures a click. The measures are highly correlated; we assume that they are all equally important.

We divide the area between the two buttons in the dialog box in Figure 7 into four parts. With different calibrations of its capturing affordance, each button can claim clicks that fall into some number of the four areas. This establishes a set of conditions between which we can vary the capture effect of the affordance. We vary the level of each of the two buttons: $L_o(i)$ refers to the condition in which the OK button captures clicks within any of the i ($i \leq 4$) divisions closest to the button, $L_c(i)$ likewise for the Cancel button. $L_o(0)$ and $L_c(0)$ correspond to the cases in which the OK and Cancel buttons do not capture any clicks outside their regions. The capture effects are constrained such that there is no overlap between the buttons. We thus have a two way, incomplete factorial design.

To provide a reasonable sample of behaviors, including errors, we generated 50 cancelation and 1000 confirmation sequences at each level. We now have an exploratory task ahead of us. We are not concerned with the presence or absence of statistically significant effects of the two factors on our performance measures. Rather, we want to see the general structure in the experimental data, the location of any maxima or trends in the performance measures, to point in an appropriate direction for real user testing. Response surface techniques, modified for discrete data, are appropriate (Box et al., 1978). Figure 8 shows how mean duration varies with L_c and L_o. The mean duration of tasks is lowest with a larger capture range for the OK button and a small or zero capture range for the Cancel button. This is partly an effect of the relative distribution of clicking tasks; confirmation is 20 times as likely as cancelation. Changes in duration are caused mainly by errors in clicking. We can have the greatest effect in reducing the number of errors by altering the affordance for confirmation, because most errors occur during confirmation.

FIGURE 8 ABOUT HERE.

The data for error frequency, shown in Figure 9, confirms this intuition. There is little to be gained in error reduction by changing the capture range for the Cancel button. Error rates remain the same. Instead, we see differences between values only at the boundaries between OK button levels. Note that the differences in mean duration between conditions are miniscule, on the order of 10^{-2} seconds. A single error, every twenty trials, contributes only a small amount (a 0.7 or 5 second penalty) to the total. The differences in error rates, however, are potentially significant. The figures tell us that we can reduce a 4% error rate to 2% by adding the appropriate capture affordance to the OK button.

FIGURE 9 ABOUT HERE.

The last measure is of a potentially serious error: clicking the incorrect button. Figure 10 shows the frequency of the occurrence during testing. Here again of the distribution of confirmation and cancelation is a factor. In contrast to the above finding that we can reduce error rates by changing the capture range of the OK button, for serious errors we must change the Cancel button.

FIGURE 10 ABOUT HERE.

In summary, this experiment tells us that significant reduction of duration in this task is not a practical goal. Though we can detect statistically significant differences in mean duration, these would not be detectable in practice. If our goal is to reduce the relative percentage of errors, however, we can achieve a measurable reduction (from 4% to about 2.5%) simply by setting the capture affordance to its highest level for the OK button, and eliminating any capture effect for the Cancel button. By extending the capture effect beyond the region immediately between the two buttons, we can reduce the error rate even further, to less than 1%. These changes, though small and unobtrusive, may have a useful cumulative effect for mouse-intensive tasks and environments. Not surprisingly, we find such affordance mechanisms implemented in a wide variety of contexts in existing interfaces.

5 AFFORDANCES IN A DECISION SUPPORT SYSTEM

The model laid out in the previous section, while general, has serious weaknesses. In modern user interfaces, a user will only rarely encounter problems in using specific widgets, and remedies at the given level of abstraction can only provide a limited benefit. Problems arise more often at a conceptual level. What is it possible to do in the interface? Why can't a given operator be executed in the current state? How can one reach a desired goal? These questions can be cast in the same terms as questions about pointer movement and mouse gestures, relying on predicates and operators that represent concepts in the problem domain. In fact, a problem space traditionally acts as a means of abstracting away unnecessary detail, and so the low-level operators we have discussed up to this point are actually less common than representations that deal with knowledge and data in a specific problem-solving domain. In this section we examine the application of our affordance modeling approach to a problem in decision support for exploratory data analysis.

Over the past several years we have developed an assistant for data exploration called Aide (St. Amant & Cohen, 1998a, 1998b), shown in Figure 11. Aide's assistance takes the form of the partial automation of statistical strategies. Statistical strategies are formal descriptions of the actions and decisions involved in applying statistical tools to a problem (Hand, 1986). Strategies have been designed and implemented for simple and multiple linear regression (Faraway, 1992, 1994; Gale, 1986), MANOVA (Hand, 1986), collinearity diagnosis (Oldford & Peters, 1986), variable selection in specific contexts (Fowlkes et al., 1987), and other statistical procedures (Gale et al., 1993). The potential benefits of general-purpose, automated strategies are enormous, greatly reducing the work load of statisticians and providing non-statisticians with easily accessible expert advice.

FIGURE 11 ABOUT HERE.

Aide contains about a hundred hierarchical plans and operators, at different levels of detail, that capture elements of common statistical practice, such as the examination of residuals after fitting a function to a relationship, the search for refinements and predictive factors when observing clustering, the reduction of complex patterns to simpler ones, and so forth. The work described here concerns a limited version of the system that contains only the operators necessary to implement a single statistical strategy for simple linear regression. A representative subset of these twenty four operators is shown in Figure 12.

FIGURE 12 ABOUT HERE.

Linear regression, even for two variables, is not a trivial exercise. While the computation itself can be automated, the validity of the procedure relies on a number of assumptions that do not always hold (Cook & Weisberg, 1982). One of the most obvious assumptions is that each observation has two values, a predictor and a response. If for some observation either is missing, then the relationship must be adjusted (the simplest solution being to remove the observation.) Some potential difficulties can be tested before the regression is computed, but some appear only afterwards. Residual diagnostic procedures can detect problems such as unequal residual variance, the presence of influential observations, higher-order patterns that are not captured by the linear model, and so forth. These issues make regression modeling an iterative decision-making process. Data analysts make use of statistical theory, practical rules of thumb, and observations about patterns in the data in the incremental process of building a model.

To support this process, Aide provides a conventional statistical user interface that gives access to buttons and menu operations for loading a dataset, composing variables into relationships, computing summary statistics, generating linear models, partitioning data, running statistical tests, and so forth. As the user executes sequences of operators to explore a dataset, however, the system is also active, constructing comparable and complementary sequences to extend and critique the user's choices. The system offers advice about which operators are appropriate at a given point and provides navigational overviews of the paths taken through the space of exploration operators, as well as paths through the space of statistical results. The result is to some extent a mixed-initiative planning process, in which data analysis is sometimes driven by the user and sometimes by the system.

Our initial consideration of affordances in Aide remains with the generic user interface operators given in Figure 6, modified only slightly. As general as these operators are, they can lead to potentially useful affordances in Aide. One aspect of data analysis is accounting for the behavior of individual data points. Outlying values, for example, can unduly influence regression parameters, along with other nonrobust estimators of the properties of variables and relationships, such as central tendency (e.g., mean) and spread (e.g. variance). Depending on our data, we might be interested in a specific data point for arbitrary reasons, which means selecting a point from a scatter plot such as the one in Figure 13. The difficulty in selecting an individual point in this graph is the density of data. The planning problem is selecting one of n distinct operators, `Move over Point 1` ... `Move over Point n`, each with an effect containing `pointer-over Point i`. Because points overlap, however, the precondition `(visible Point i)` may not hold for a given point. We would like to facilitate the establishment of the `(visible Point i)` precondition for each data point, but not at the cost of making other selections more difficult—the system may have insufficient information to concentrate on one point.

FIGURE 13 ABOUT HERE.

One possible solution adapts a common interface mechanism, small documentation windows that pop up over some controls, such as ToolTips or a Magic Lens filter (Stone et al., 1994), to incorporate an interactive element in the information displayed. When the mouse pointer slows below a threshold speed over a dense area of the scatter plot, a small magnification window pops as shown in Figure 13. (It was a simple matter to modify the PUM of Section 4, adding relevant assumptions, to derive an initial value for the density necessary to activate the mechanism.) The magnification displays all points in a fixed radius around the pointer position. Points are spread out in the window by increasing their distance from the central point and by adding small amounts of random "jitter" to their position. The result is a display that preserves the general pattern of points, so that individual ones are recognizable by position and are also selectable. The same mechanism thus provides both an evaluation affordance for the `visible` predicate and an execution affordance via the `pointer-over` predicate.

The basic idea of this affordance generalizes to other problems that involve unreliable selection as well. For example, cursor placement can be difficult in a word processing system that displays text in different typefaces. Placing the cursor between two narrow-width italic characters (e.g., *"ti"* or *"fl"*) can result in positioning errors, especially if the displayed characters are small. The usual solution is to require the user to redisplay the document on a larger scale, or to mouse down and then fine tune the placement with the arrow keys. A variant of the affordance would work equally well here. It would present a small pop up window containing the characters under the pointer, well-separated and in Roman typeface, with an indication (a secondary cursor) of the current selection. This mechanism should be no more obtrusive than the small documentation windows that pop up over some widgets; it is simply a generalization of that functionality, adding a contextual element to the information displayed.

In our consideration of affordances for AIDE we also examined the more abstract set of operators in Figure 12, specific to the domain of statistical reasoning. Given these operators we can identify opportunities for affordances at a higher level of abstraction. Now if we were to proceed as we did in Section 4, we would define cost functions for evaluation and operator execution, along with appropriate methods to allow simulation. Unfortunately, because these operators represent composite activities of some complexity, it is impractical to develop quantitative estimates of execution and evaluation costs. Nevertheless we can still use the planning framework to give us qualitative guidance in the interface design.

One of the problems we face is guiding the user through a sequence of actions in the interface, indicating which operators are appropriate but not forcing them to be chosen. This problem is familiar to the developers of most agent-based and mixed-initiative systems. A conventional menu-based interface does little to accommodate such interactive guidance, for several reasons. Available commands are "hidden" within pulldown menus. There is no standard way to indicate that a specific command is appropriate or desirable, in contrast to being simply applicable. Conventional interfaces give little context for sequences of commands that represent common practice. Conventional systems are also entirely driven by the user, which is desirable for a number of reasons, but can prevent the system from taking useful independent action even when it is appropriate.

To address these points we developed some rules of thumb for introducing affordances into the user's interaction with AIDE, based on our normative representation of the statistical reasoning process as planning. This is a purely qualitative interpretation of our definitions of user interface affordances.

- An always-visible control for an operator constitutes a precondition execution affordance for that operator. That is, the user can select it with lower cost than if it were otherwise in a menu or available through a command line. This is common practice, as seen in customizable toolbars for frequent operators.
- If the relationship between an operator and its preconditions is made explicit, this introduces precondition evaluation affordances for that operator. Part of knowing whether an operator is appropriate is knowing what it depends on.
- If an operator's precondition can be established by some operator sequence, then the automatic execution of this sequence constitutes a precondition execution affordance.
- If automatic execution in service of a precondition is associated with a visual indication of the result, this indication constitutes a precondition evaluation affordance.
- If the automatic execution of an operator can be overridden by the user, then the mechanism for this constitutes a precondition execution affordance for *other* operators, or no operator at all (i.e., a cancelation of the given operator.)

Effect affordances turned out not to be a factor in this domain. Note that these are heuristics that require some judgment in their application. One unfortunately common approach to decision support is to automate whatever can be automated, and leave the rest to the user (Rouse et al., 1987; Silverman, 1992). In our rules, this is interpreted as providing an execution affordance for such activities. This must be balanced by the recognition that affording Activity X may preclude Activity Y. If an automated system has no means of judging whether X or Y is more appropriate, then its provision of an affordance for X (simply because it is possible to automate X) can be worse than useless. Recognition and evaluation of tradeoffs are critical elements in a design—all affordances are not equal.

Our rules led us to the navigational interface shown in Figure 14. The general appearance is similar to existing commercial graphical statistics interfaces, with data values for the various operator computations available in pop-up windows. A significant difference is the inclusion of explicit relationships between preconditions and operators, which represent the evaluation of features of the data. This creates a hierarchy of diagnostic evaluations that lead to the model shown.

FIGURE 14 ABOUT HERE.

Each precondition has an associated string that is displayed, as well as a check box to indicate whether it holds for the relationship under analysis. The check box values are initially set by the system, but can be modified by the user if appropriate. For example, although one of the system's diagnostic functions detects uneven variance in the residuals, the user can examine the relevant graph (by selecting the text item) and override the judgment by checking the box by hand. The evaluation propagates immediately up the hierarchy to the *Test model* operator, for which all preconditions now hold.

The design guidance provided by our affordance model, filtered through the qualitative rules, was limited. We depended heavily on our experience with an earlier version of the system and with other types of navigation systems (St. Amant et al., 1998). Consideration of affordances was beneficial nevertheless. It helped us focus attention on the areas we intended the system to contribute to the user's problem solving, and it provided a more structured justification for our design decisions, which in earlier versions of the system were based almost entirely on informal user evaluation and general design knowledge.

6 CONCLUSION

The work we have presented leaves open a number of problems for future work.

First, the operator representation has serious weaknesses as a reflection of reality. A great deal of human activity cannot be represented as sequences of precondition-effect pairs. Most importantly, and in addition to all of the issues raised in Figure 2, the representation does not capture the hierarchical nature of actions. For example, in our discussion of AIDE we described affordances in terms of independent sets of operators, one at the level of widget interaction, and one at the level of statistical modeling. It seems obvious, however, that reasoning operators at the higher level eventually decompose into base-level interface actions. 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. This connection is missing in our current work. We are nevertheless confident that our interpretation of affordance as planning cost will extend to hierarchical operators, which are common in practical planning systems (and are used in the full version of AIDE.) For example, precondition and effect affordances will remain the same, but we must now consider operators that expand into hierarchies of other operators at greater levels of detail; it is not entirely clear where affordances, which we currently associate with single operators, should stand in relation to structured sets of operators.

Second, the approach we have taken to modeling evaluation affordances in terms of cost is quite rudimentary. Our example of an implicit operator for recognizing a button icon—an operator with an expected duration—is far too simplistic to work in the long run. Our current work in the area of graphical perception operators holds some promise for better models in this area, in specific domains such as visualization, but this is only a first step. Eventually we will need to understand in detail how and why users interpret visual and interaction cues in the ways that they do; this depends on research in a wide variety of areas in HCI and cognitive science.

A third limitation of this work is the informal relationship between the model of affordance and interface design. The level of abstraction in the problem space representation does not allow for direct influence in design decisions, but instead can only point out potential opportunities. This characteristic is shared with other abstract task modeling techniques as well. We further observe that most of the mechanisms we describe have existed in commercial or research user interfaces for some time. This is not surprising, however; it is not unusual for the market to drive practice far ahead of theory. Our models take HCI theories concerning affordance one step closer.

We believe that this approach, despite these limitations, can provide clarity in the application of affordance concepts to the user interface.

NOTES

Acknowledgments. This article benefited greatly from the advice, criticism, and encouragement of Larry Birnbaum, Paul Cohen, John Flach, James Lester, Scott MacKenzie, Michael Young, and two anonymous reviewers. Thanks also to Scott MacKenzie for seeing the article through the review process.

Support. This research was partially 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 copyright notation hereon.

Authors' Addresses. Robert St. Amant, NCSU Computer Science, 446 EGRC, 1010 Main Campus Drive, Raleigh, NC 27695. Email: stamant@csc.ncsu.edu.

References

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

Agre, P. E., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. *Proceedings of AAAI-87: National Conference on Artificial Intelligence*, 268–272. Menlo Park, CA: AAAI Press.

Agre, P. E., & Chapman, D. (1990). What are plans for? *Robotics and Autonomous Systems* 6:17–34.

Allen, J., Hendler, J., & Tate, A. (Eds.). (1990). *Readings in planning.* San Mateo, CA: Morgan Kaufmann.

Alterman, R., Zito-Wolf, R., and Carpenter, T. (1998). Pragmatic action. *Cognitive Science* 22(1):55–105.

Anderson, B. (1993). Graphical interfaces considered as representations of the real world: Implications of an affordances-based model. In S. S. Valenti & J. B. Pittenger (Eds.), *Studies in perception and action II* (pp. 89–93). Hillsdale, NJ: Lawrence Erlbaum Associates.

Apple Computer, Inc. (1992). *Macintosh human interface guidelines.*

Arkin, R. C., Cervantes-Perez, F., & Weitzenfeld, A. (1997). Ecological robotics: A schema-theoretic approach. In R. C. Bolles, H. Bunke, & H. Noltemeier (Eds.), *Intelligent robots: sensing, modelling and planning* (pp. 377–393). Singapore; River Edge, NJ: World Scientific.

Baecker, R. M., Grudin, J., Buxton, W. A. S., & Greenberg, S. (Eds.). *Readings in human-computer interaction: Toward the year 2000.* San Francisco, CA: Morgan Kaufmann.

Blum, A., & Furst, M. (1997). Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* 90:281–300.

Boutilier, C., Dean, T., & Hanks, S. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research.* In press.

Box, G. E. P., Hunter, W. G., & Hunter, J. S. (1978). *Statistics for experimenters: An introduction to design, data analysis, and model building.* New York: Wiley.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Carroll, J. M., Mack, R. L., & Kellogg, W. A. 1988. Interface metaphors and user interface design. In Helander, Martin, editor 1988, *Handbook of human-computer interaction* (pp. 45–65). Amsterdam; New York: North-Holland.

Christoffersen, K., Hunter, C. N., & Vicente, K. J. (1997). A longitudinal study of the effects of ecological interface design on fault management performance. *Journal of Cognitive Ergonomics* 1:1–24.

Cohen, P. R., Oates, T., Atkin, M., & Beal, C. R. (1996). Building a baby. *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (pp. 518–522). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohen, P. R., Atkin, M., Oates, T., & Beal, C. R. (1997). Neo: Learning conceptual knowledge by sensorimotor interaction with an environment. *Proceedings of the First International Conference on Autonomous Agents* (pp. 170–177). New York: ACM.

Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression.* New York: Chapman & Hall.

Dix, A. J., Finlay, J. E., Abowd, G. D., & Beale, R. (1998). *Human-computer interaction* (2nd ed.). London: Prentice Hall.

Eberts, Ray E. (1994). *User interface design.* Englewood Cliffs, NJ: Prentice Hall.

Faraway, J. J. (1992). On the cost of data analysis. *Journal of Computational and Graphical Statistics* 1(3):213–229.

Faraway, J. J. (1994). Choice of order in regression strategy. In P. Cheeseman & R. W. Oldford (Eds.), *Building models from data: Artificial intelligence and statistics IV.* New York: Springer.

Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3/4):189–208.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47:381–391.

Flach, J. (1995). The ecology of human machine systems: A personal history. In J. Flach, P. Hancock, J. Caird, & K. Vicente (Eds.), *Global perspectives on the ecology of human-machine systems* (pp. 1–13). Hillsdale, NJ: Lawrence Erlbaum Associates.

Fowlkes, E. B., Gnanadesikan, R., & Kettenring, J. R. (1987). Variable selection in clustering and other contexts. In C. L. Mallows (Ed.), *Design, data, and analysis: by some friends of Cuthbert Daniel.* New York: Wiley.

Gale, W. A., Hand, D. J., & Kelly, A. E. (1993). Statistical applications of artificial intelligence. In C. R. Rao (Ed.), *Handbook of Statistics* 9:535–576.

Gale, W. A. (1986). REX review. In W. A. Gale (Ed.), *Artificial intelligence and statistics* (pp. 173–227). Reading, MA: Addison-Wesley.

Gaver, W. W. (1991). Technology affordances. *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems*, 79–84. New York: ACM.

Gibson, J. J. (1979). *The ecological approach to visual perception.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Greeno, J. G., & Moore, J. L. (1993). Situativity and symbols: Reply to Vera and Simon. *Cognitive Science* 17:49–59.

Hand, D. J. (1986). Patterns in statistical strategy. In W. A. Gale (Ed.), *Artificial intelligence and statistics* (pp. 355–387). Reading, MA: Addison-Wesley.

Hendler, J., Tate, A., & Drummond, M. (1990). AI planning: Systems and techniques. *AI Magazine* 11(2): 61–77.

John, B. E., & Kieras, D. E. (1996a). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction* 3(4):320–351.

John, B. E., & Kieras, D. E. (1996b). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction* 3(4):287–319.

Kalish, M. (1993). Affordance learning as a problem in information integration. In S. S. Valenti & J. B. Pittenger (Eds.), *Studies in perception and action II* (pp. 130–133). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In Helander, M. (Ed.), *Handbook of Human-Computer Interaction* (pp. 135–157). Amsterdam; New York: North-Holland.

Kieras, D., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4):391–438.

Kirlik, A., Miller, R. A., & Jagacinski, R. J. (1993). 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.

Lewis, M. (1998). Designing for human-agent interaction. *AI Magazine* 97(2):67–78.

MacKenzie, I. S., & Buxton, W. (1994). The prediction of pointing and dragging times in graphical user interfaces. *Interacting with Computers* 6:213–227.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems*, 161–166. New York: ACM.

MacKenzie, I. S. (1995). Movement time prediction in human-computer interfaces. In R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg (Eds.), *Readings in human-computer interaction: Toward the year 2000* (pp. 483–493). San Francisco, CA: Morgan Kaufmann.

Mark, L. S., Balliett, J. A., Craver, K. D., Douglas, S. D., & Fox, T. (1990). What an actor must do in order to perceive the affordance for sitting. *Ecological Psychology* 2(4):325–366.

Mark, L. S. (1987). Eyeheight-scaled information about affordances: A study of sitting and stair climbing. *Journal of Experimental Psychology: Human Perception and Performance* 10:683–703.

Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., and Smith, J. E. K. (1988). Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review* 95(3):340–370.

Münch, S., & Dillman, R. (1997). Haptic output in multimodal interfaces. *Proceedings of the 1997 Conference on Intelligent User Interfaces*, 105–112. New York: ACM.

Myers, B. A., Giuse, D., Dannenberg, R. B., Vander Zanden, B., Kosbie, D., Pervin, E., Mickish, A., , & Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer* 23(11):71–85.

Newell, A., & Simon, H. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice Hall.

Norman, D. A. (1991). Cognitive artifacts. In J. M. Carroll (Ed.), *Designing interaction: Psychology at the human-computer interface* (pp. 17–38). Cambridge; New York: Cambridge University Press.

Norman, D. A. (1988). *The psychology of everyday things.* New York: Basic Books.

Norman, D. A., & Draper, S. W. (1986). *User centered system design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Oates, T., & Cohen, P. R. (1996). Searching for structure in multiple streams of data. *Proceedings of the Thirteenth International Conference on Machine Learning*, 346–354. San Mateo, CA: Morgan Kaufmann.

Oldford, R. W., & Peters, S. C. (1986). Implementation and study of statistical strategy. In W. A. Gale (Ed.), *Artificial intelligence and statistics* (pp. 335–349). Reading, MA: Addison-Wesley.

Penberthy, J., & Weld, D. S. (1992). UCPOP: A sound, complete, partial order planner for ADL. *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, 103–114.. San Mateo, CA: Morgan Kaufmann.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-computer interaction*. Wokingham, England: Addison-Wesley.

Rassmussen, J. (1988). A cognitive engineering approach to the modeling of decision making and its organization in process control, emergency management, CAD/CAM, office systems, and library systems. *Advances in Man-machine Systems Research* 4:165–243.

Rouse, W. B., Geddes, N. D., & Curry, R. E. (1987). An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction* 3:87–122.

Schmill, M. D., Rosenstein, M. T., Cohen, P. R., & Utgoff, P. (1998). Learning what is relevant to the effects of actions for a mobile robot. *Proceedings of the Second International Conference on Autonomous Agents*, 247–253. New York: ACM.

Silverman, B. G. (1992). Human-computer collaboration. *Human-Computer Interaction* 7:165–196.

St. Amant, R., & Cohen, P. R. (1998a). Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics* 7(4):545–558.

St. Amant, R., & Cohen, P. R. (1998b). Interaction with a mixed-initiative system for exploratory data analysis. *Knowledge-Based Systems* 10(5):265–273.

St. Amant, R., Long, T., & Dulberg, M. S. (1998). Evaluation in a navigation testbed. *Knowledge-Based Systems* 11(1):61–70.

Stone, M. C., Fishkin, K., & Bier, E. A. (1994). The movable filter as a user interface tool. In *Proceedings of the CHI '94 Conference on Human Factors in Computing Systems*, 306–312. New York: ACM.

Suchman, L. (1987). *Plans and situated action*. Cambridge; New York: Cambridge University Press.

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

Vera, A. H., & Simon, H. A. (1993a). Situated action: A symbolic interpretation. *Cognitive Science* 17:7–48.

Vera, A. H., & Simon, H. A. (1993b). Situated action: Reply to reviewers. *Cognitive Science* 17:77–86.

Vicente, K. J., & Rasmussen, J. (1990). The ecology of human-machine systems II: Mediating "direct perception" in complex work domains. *Ecological Psychology* 2(3):207–249.

Warren, W. H., Jr. (1995). Constructing an econiche. In J. Flach, P. Hancock, J. Caird, & K. Vicente (Eds.), *Global perspectives on the ecology of human-machine systems* (pp. 210–237). Hillsdale, NJ: Lawrence Erlbaum Associates.

Weld, D. S. (1994). An introduction to least commitment planning. *AI Magazine* 15(4):27–61.

Woods, D. D., & Roth, E. M. (1988). Cognitive systems engineering. In M. Helander (Ed.), *Handbook of Human-Computer Interaction* (pp. 3–43). Amsterdam; New York: North-Holland.

Woods, D. D. (1991). The cognitive engineering of problem representations. In G. R. S. Weir & J. L. Alty (Eds.), *Human computer interaction and complex systems* (pp. 169–188). London; San Diego, CA: Academic Press.

Worden, A., Walker, N., Bharat, K., & Hudson, S. (1997). Making computers easier for older adults to use: Area cursors and sticky icons. *Proceedings of the CHI '97 Conference on Human Factors in Computing Systems*, 266–271. New York: ACM.

Young, R. M., Green, T. R. G., & Simon, T. (1989). Programmable user models for predictive evaluation of interface designs. *Proceedings of the CHI '89 Conference on Human Factors in Computing Systems*, 15–19. New York: ACM.

Zaff, B. S. (1995). Designing with affordances in mind. In J. Flach, P. Hancock, J. Caird, & K. Vicente (Eds.), *Global perspectives on the ecology of human-machine systems*, (pp. 238–272). Hillsdale, NJ: Lawrence Erlbaum Associates.

Figure captions

```
(: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:

Classical planning assumptions	User interface design guidelines
Deterministic effects. The effects of an action can be guaranteed to hold after its execution.	Consistency (Apple, 1992). 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 (Hendler et al., 1990).)	User control (Apple, 1992). 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 (Woods & Roth, 1988).
Passage of time. Plans remain appropriate in the interval between their generation and their execution.	Perceived stability (Apple, 1992). 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 delayed to avoid dead-ends.	Forgiveness (Apple, 1992). 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.

Figure 2:

```
(:operator |Mouse down on|
           :parameters (?object)
           :cost (mouse-down-cost ?object)
           :precondition (and (not (mouse down))
                              (visible ?object)
                              (pointer-over ?object)
                              (button ?object))
           :effect (and (mouse down)
                        (when (not (eq ?object background))
                          (object-activated ?object))))
(:operator |Mouse up on|
           :parameters (?object)
           :cost (mouse-up-cost ?object)
           :precondition (and (mouse down)
                              (visible ?object)
                              (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)
           :cost (movement-cost ?source ?target)
           :precondition (and (not (mouse down))
                              (pointer-over ?source)
                              (visible ?target))
           :effect (and (not (pointer-over ?source))
                        (pointer-over ?target)))
```

Figure 3:

Affordance type	Definition
Simple	Environment can reach a state in which precondition holds.
Precondition execution	Decreases cost of establishing precondition predicates.
Effect execution	Decreases cost of establishing effect predicates.
Precondition evaluation	Decreases cost of determining precondition predicates.
Effect evaluation	Decreases cost of determining effect predicates (feedback.)

Figure 4:

Object	Feature/predicate	Action/operator
Hammer (Gibson, 1979)	Heaviness	Swing arm
Ground (Gibson, 1979)	Solidity	Support (a body)
Corridor (Kirlik et al., 1993)	Unobstructedness	Allow line-of-sight
Shelf (Zaff, 1995)	Accessibility	Reach
Doorway (Warren, 1995)	Width	Allow passage
Stair riser (Mark, 1987)	Height	Climb
Chair (Mark, 1987)	Seat height	Sit
Incline (Mark et al., 1990)	Steepness	Walk/crawl

Figure 5:

```
(:operator |Mouse down on background|
           :precondition (and (not (mouse down)) (no-focus) (inactive))
           :effect (and (mouse down)))
(:operator |Mouse up on background|
           :parameters (?widget)
           :precondition (and (mouse down) (inactive))
           :effect (and (not (mouse down))
                        (when (in-focus ?widget)
                          (and (no-focus) (not (in-focus ?widget))))))
(:operator |Move into widget|
           :parameters (?widget)
           :precondition (and (not (mouse down)) (no-focus))
           :effect (and (in-focus ?widget) (not (no-focus))))
(:operator |Move out of widget|
           :parameters (?widget)
           :precondition (and (not (mouse down)) (in-focus ?widget))
           :effect (and (no-focus) (not (in-focus ?widget))))
(:operator |Mouse down on widget in focus|
           :parameters (?widget)
           :precondition (and (not (mouse down)) (in-focus ?widget) (inactive))
           :effect (and (mouse down)
                        (not (inactive)) (widget-activated ?widget)
                        (mouse-down-action ?widget)))
(:operator |Mouse up on activated widget|
           :parameters (?widget)
           :precondition (and (mouse down)
                              (widget-activated ?widget)
                              (in-focus ?widget))
           :effect (and (not (mouse down))
                        (inactive) (not (widget-activated ?widget))
                        (widget-done ?widget)))
(:operator |Move into widget in focus|
           :parameters (?widget)
           :precondition (and (mouse down) (in-focus ?widget))
           :effect (and (not (inactive)) (widget-activated ?widget)))
(:operator |Move with activated widget|
           :parameters (?widget)
           :precondition (widget-activated ?widget)
           :effect (and (when (movable ?widget)
                          (widget-moved ?widget))
                        (when (not (movable ?widget))
                          (and (inactive) (not (widget-activated ?widget))))))
```

Figure 6:

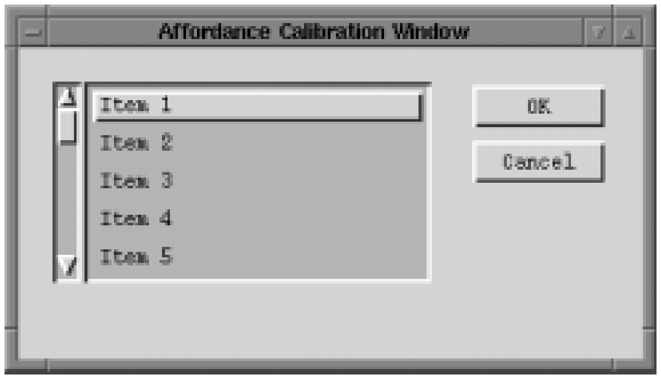

Affordance Calibration Window
Item 1
Item 2
Item 3
Item 4
Item 5
OK
Cancel

Figure 7:

	$L_o(0)$	$L_o(1)$	$L_o(2)$	$L_o(3)$	$L_o(4)$
$L_c(0)$	0.71	0.71	0.71	0.70	0.70
$L_c(1)$	0.71	0.71	0.71	0.70	-
$L_c(2)$	0.72	0.71	0.71	-	-
$L_c(3)$	0.74	0.73	-	-	-
$L_c(4)$	0.76	-	-	-	-

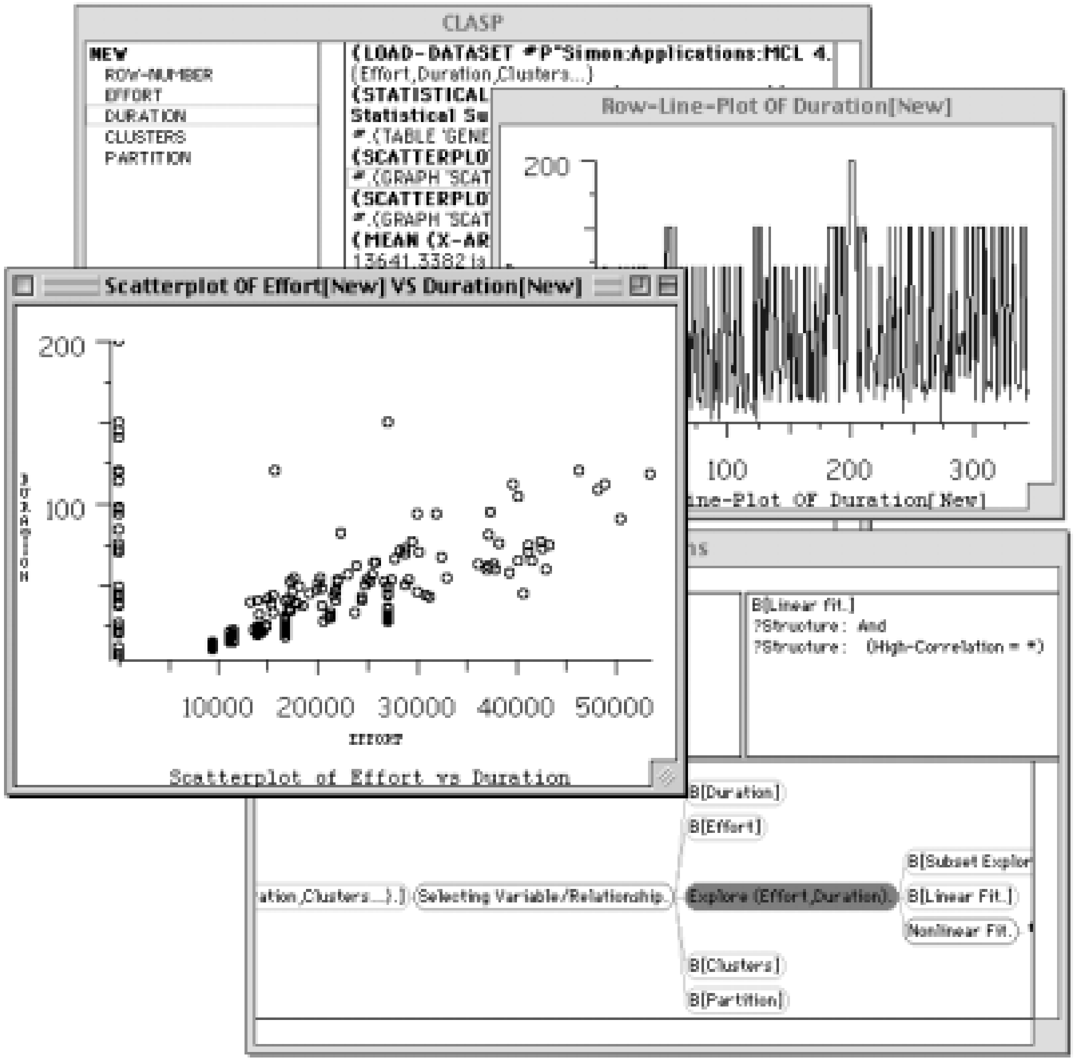
Figure 8:

	$L_o(0)$	$L_o(1)$	$L_o(2)$	$L_o(3)$	$L_o(4)$
$L_c(0)$	0.041	0.034	0.029	0.024	0.024
$L_c(1)$	0.041	0.034	0.029	0.024	-
$L_c(2)$	0.041	0.034	0.029	-	-
$L_c(3)$	0.040	0.033	-	-	-
$L_c(4)$	0.040	-	-	-	-

Figure 9:

	$L_o(0)$	$L_o(1)$	$L_o(2)$	$L_o(3)$	$L_o(4)$
$L_c(0)$	0	0	0	0	0
$L_c(1)$	0	0	0	0	-
$L_c(2)$	1	1	1	-	-
$L_c(3)$	5	5	-	-	-
$L_c(4)$	10	-	-	-	-

Figure 10:



```
(LOAD-DATASET #P"Simon:Applications:MCL 4.
(Effort,Duration,Clusters...)
(STATISTICAL
Statistical Su
#.(TABLE 'GENE
(SCATTERPLO
#.(GRAPH 'SCAT
(SCATTERPLO
#.(GRAPH 'SCAT
(MEAN (X-AR
13641.3382 (a
```

```
B[Linear fit.]
?Structure : And
?Structure :  (High-Correlation = *)
```

Figure 11:

```
(:operator simple-linear-regression
           :parameters (?y ?x)
           :precondition (and (y ?y)  (x ?x)
                              (observations-checked ?y ?x)
                              (y-checked ?y)
                              (linearity-checked ?y ?x)
                              (x-checked ?x))
           :effect (compute-regression ?y ?x))
(:operator check-observations
           :parameters (?y ?x)
           :precondition (and (y ?y)  (x ?x)
                              (observed negative-weights ?y ?x false)
                              (observed missing-values ?y ?x false))
           :effect (observations-checked ?y ?x))
(:operator check-weights
           :parameters (?y ?x ?observations)
           :precondition (and (negative-weights ?x ?observations)
                              (not (= ?observations :none)))
           :effect (and (action remove-observations negative-weights ?x ?observations)
                        (observed negative-weights ?y ?x false)))
(:operator check-weights-default
           :parameters (?y ?x)
           :precondition (and (object ?y)
                              (object ?x)
                              (and (not (negative-weights ?x :none))
                                   (not (negative-weights ?y :none))))
           :effect (observed negative-weights ?y ?x false))
(:operator check-missing
           :parameters (?y ?x ?observations)
           :precondition (and (or (missing-values ?x ?observations)
                                  (missing-values ?y ?observations))
                              (not (= ?observations :none)))
           :effect (and (action remove-observations missing-values ?x ?observations)
                        (action remove-observations missing-values ?y ?observations)
                        (observed missing-values ?y ?x false)))
(:operator check-missing-default
           :parameters (?y ?x)
           :precondition (and (object ?y)
                              (object ?x)
                              (and (not (missing-values ?y :none))
                                   (not (missing-values ?x :none))))
           :effect (observed missing-values ?y ?x false))
. . .
```

Figure 12:

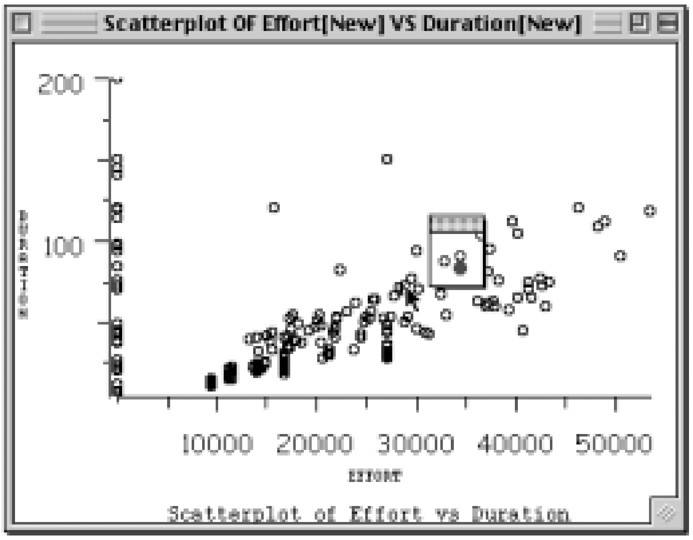
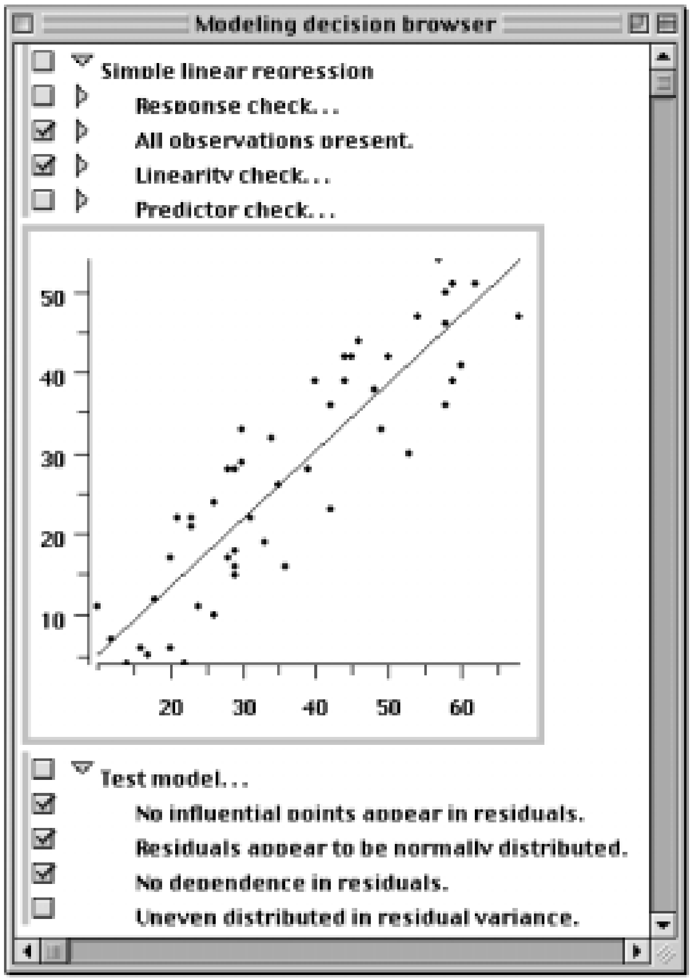

Scatterplot OF Effort[New] VS Duration[New]
200
100
DURATION
10000
20000
30000
40000
50000
EFFORT
Scatterplot of Effort vs Duration

Figure 13:



☐ Simple linear regression
☐ Response check...
☑ All observations present.
☑ Linearity check...
☐ Predictor check...

☐ Test model...
☑ No influential points appear in residuals.
☑ Residuals appear to be normally distributed.
☑ No dependence in residuals.
☐ Uneven distributed in residual variance.

Figure 14: