Navigation and Planning in a Mixed-Initiative User Interface

Robert St. Amant
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
stamant@csc.ncsu.edu

Abstract

Mixed-initiative planning is one approach to building an intelligent decision-making environment. A mixed-initiative system shares decision-making responsibility with the user such that it acts sometimes as a tool, to be directly applied to a specific task, and other times as an autonomous problem-solver. In the best case, the user can delegate the details of a task to the automated system without giving up the ability to guide and review the decision-making process. We have developed a simple mixed-initiative planner that incorporates a view of problem-solving as navigation. We have explored this notion in two different applications: exploratory statistical analysis and layout design for user interface dialogs. This paper discusses navigation issues in the context of these two systems, the potential benefits of the approach, and some implications for user interface design.

Introduction

A large class of software packages can be described as exploratory decision-support environments. Common domains are statistical analysis, mathematics, discrete simulation, computer-aided design, and graphic design. Exploratory environments in these domains give the user the computational facilities to make and evaluate complex decisions (Gallopoulos, Houstis, & Rice 1994). Unfortunately, most environments concentrate on support at the level of single operations, ignoring the context of the coherent decision-making process in which these operations are selected. For large problems, the cognitive burden on the decision-maker can easily be too great to manage.

Consider conventional statistical software, for example. Though modern systems contain a rich set of statistical operations, suitable for almost any application, they are relatively limited in their support of the decision-making process. While the user might say, "Generate a linear fit for this bivariate relationship," one could not expect the system to generate a least-squares or perhaps a resistant fit, check the residuals for indications (e.g., curvature, outliers, unequal variance), perform appropriate transformations, iteratively refit the data if necessary, and report all interesting results. Procedures like this, though often routine, require human judgment and interpretation at different points to be carried out properly. Complete automation of these procedures is impossible, from the programmer's point of view, as well as undesirable, from the user's point of view (Lubinsky & Pregibon 1988). Partial automation, however, in the form of intelligent automated assistance, *is* an alternative, a necessary one given the size and complexity of problems faced by data analysts. The goal is to balance system autonomy with accommodation of the user's knowledge about the domain and goals of the decision-making process.

We have taken a mixed-initiative planning approach to automated assistance. In a mixed-initiative system, the user and the machine both contribute to a problem solution—formulation, development, analysis, repair—without the the need for a constant exchange of explicit instructions (Burstein & McDermott 1996). User interaction is a central design issue. Approaches include casting the interaction as a peer-to-peer dialog (Ferguson, Allen, & Miller 1996), as a type of visual programming (Bonar & Liffick 1991), as an advice-exchanging relationship (Myers 1996), among many other possibilities. In our approach, user interaction is based on a model of problem-solving as *navigation.*

We have built two systems to explore this notion. The first, called AIDE, is an interactive system for exploratory data analysis (St. Amant & Cohen 1997a). AIDE incrementally explores a dataset, guided by user directives and its own evaluation of the data. In contrast to a conventional statistical package, AIDE attempts to interpret user actions in the context of more comprehensive analysis procedures, which are represented as script-like plans in its knowledge base. These plans evaluate indications in data, build appropriate descriptions of data, and attempt to combine results in a more coherent whole. A single user action can initiate planning activity to automatically extend or

```
(define-plan fit-relationship ()
  :satisfies (p-description :fit ?model ?structure ?fit-operation ?fit-relationship)
  :features  ((?structure ((:dataset-type relationship))))
  :body (:SEQUENCE
         (:SUBGOAL generate-fit (generate-fit ?structure ?fit-operation ?fit-relationship))
         (:SUBGOAL evaluate-fit (explore-by ?strategy ?model ?fit-relationship ?result))))
```

Figure 1: One component plan of a regression strategy

refine the direct result of the action, relieving the user of some of the burden of exploration.

The second system, called NIFT, is a tool for designing the layout of interaction objects in user interface dialogs.[1] As the user adds and modifies the text fields, menu boxes, buttons, and icons of a dialog box, NIFT explores alternative arrangements by repositioning, resizing, and regrouping the objects. The intention is to give the user a broader and more explicit view of alternatives in the sequence of design decisions that lead to a given layout.

AIDE and NIFT rely on the same mixed-initiative planner, AP, originally developed for AIDE alone. In both systems, navigation plays a central role in apprising the user of the system's decisions, its justifications, and its results. This paper describes navigation in AP's mixed-initiative framework, how it shapes user interaction in these two applications, and some potential implications for other types of interfaces.

Mixed-initiative planning

Mixed-initiative planners attempt to integrate user judgment into the otherwise automated solution process for complex problems. James Allen (1994) distinguishes mixed-initiative planning from conventional planning by three characteristics: mixed-initiative planners allow problem-solving initiative to change hands flexibly and opportunistically between the user and the system; they are able to shift focus of attention to meet changes in user needs; they contain mechanisms for maintaining shared, implicit knowledge. Burstein and McDermott (1996) expand on these and related issues in a summary of the state of the art.

The AP planner is a straightforward reactive, script-based planner. We have described its design elsewhere (St. Amant & Cohen 1996); a brief summary will be enough to support our discussion. A planning session begins with the establishment of a top level goal. AP searches through its library for an appropriate plan and expands it into its component control constructs: sequences, conditionals, iteration, mapping, and so forth. These expand in turn until a set of subgoals is reached. Subgoals are satisfied by the instantiation of further plans, or by primitive actions, which execute code directly rather than establishing subgoals.

A simple plan from the AIDE library, representing a fragment of a regression strategy based on Gale and Pregibon's REX (Gale 1986), is given in Figure 1. When executed, the plan first establishes a goal for the computation of the regression, and then a goal for evaluation or diagnosis of the result. This involves selecting and executing plans that test whether the resulting model is appropriate for the data. These further plans search for irregularities such as high leverage points, outliers in the residuals of the fit, and so forth.

The planning process is more complex than it might initially appear: several plans in the library can usually satisfy any single goal, and there may be an unlimited number of ways to bind a plan's internal variables to different values. For each decision, or *focus point* (Carver & Lesser 1993), a set of control rules decides which of the matching plans or valid variable bindings to select. The `fit-relationship` plan above is activated by control rules (not displayed) that decide whether a regression is appropriate for the relationship. In this case, the control rules check for continuous variables, lack of strong clustering, and no strong evidence of nonlinearity.

As planning continues, a network of focus points that represent plan selection and variable binding decisions is generated. When each focus point is reached, control rules choose from these options: to select from the current choices, to abandon the current decision to refocus on a different focus point, with different choices, or to abdicate and present the decision to the user. Similarly, when presented with a focus point decision, the user may select from the choices given, move to a different focus point, or possibly execute an operation not considered by the planner. Thus either participant may decide how to proceed at a given decision point, or whether to refocus on a different part of the process. As the decision-making process unfolds, the focus point network is extended by both the user and the planner, with backtracking when necessary, until the top-level goal has been satisfied.

The AP planner sacrifices much of the opportunism and flexibility of conventional generative planning techniques for the ability to represent procedural expert knowledge. Its design nevertheless handles much of the complexity of the domains of AIDE and NIFT. Further, it provides an appropriate environment for us to study user interaction through navigation.

[1] NIFT is based on the Interface Tools Designer, part of Digitool's MCL Common Lisp development environment.

Navigation

Navigation relies on a metaphor between wayfinding in a physical space and the properties of the target domain. The most familiar example is hypertext navigation on the World Wide Web. Unfortunately, as users browse from one point to another, sometimes following long chains of discovery, users can find it difficult to return to a previous point, to recall how the current point was reached, or to decide where to go next, difficulties summarized by Nielsen (1990) as the "lost in hyperspace" feeling. Common mechanisms to reduce these difficulties include differentiated regions (e.g., node coloring), maps, guided tours, landmark nodes, histories, and other summaries (Kim & Hirtle 1995).

Techniques for navigating through an information space are equally applicable to navigation through a space of decisions, such as the focus point network generated by the AP planner. The decision space differs in some ways, however, in particular in that it may be dynamically modified by a semi-autonomous system. Some of the assumptions of information navigation must be altered:

- Nodes correspond to elements of information, edges to relationships between the elements. In a decision space, the natural primitive element is a decision, but there are other plausible candidates: justifications, goals, primitive actions, control constructs, computed results. Similarly, the types of relationships between these decision elements may be much more varied than the usual association and hierarchy links in an information space.
- Navigational overviews show the user several elements at once in reduced form. While textual summaries and thumbnail graphics are often adequate for hypertext overviews, the display of information relevant to a decision is more difficult, potentially depending on the domain, the problem-solving context, and other factors. Mechanisms for reducing the complexity of overviews, such as fisheye views and zooming, may also need to be adapted.
- Navigation mechanisms generally treat an information space as being static, if potentially very large. The essence of a decision space, in contrast, is its malleability. The relationships between elements may change dynamically, and the exploration fringe expands based on both explicit actions of the user and potentially unexpected actions of the system. User disorientation becomes a serious issue.

The navigational facilities in AP do not address these issues in their full generality. Nevertheless, we find that conventional techniques can be usefully extended to support navigation through a decision space.

AIDE

AIDE is built on CLASP, an interactive environment for data analysis and exploration (Anderson 1995).

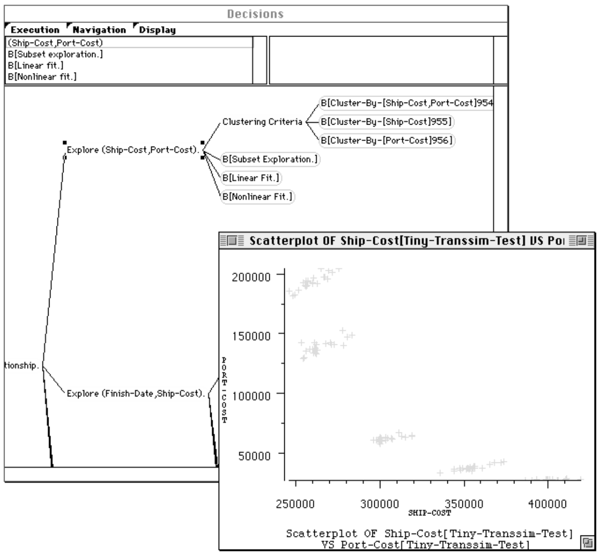

Figure 2: The navigational interface in AIDE

Through CLASP the user has access to a variety of statistical tools: graphical displays, data manipulation operations, model-fitting algorithms, and statistical tests. AIDE helps the user identify suggestive indications in the data, carry out routine statistical procedures, and evaluate and combine results. AIDE's plans are drawn from the literature of statistical strategies, which formally describe the actions and decisions involved in applying statistical tools to a problem (Hand 1986; Gale, Hand, & Kelly 1993).

The user can interact with AIDE just as one interacts with a conventional statistical system. A navigational interface allows a more collaborative style of interaction as well. Exploration involves making decisions about which relationships to explore, which operations to apply, how to interpret the results of operations, and so forth. These decisions can depend strongly on one another, and must sometimes be revised in light of new information. The navigational interface gives the user the tools to traverse and extend this space of decisions as represented by a focus point network.

Interaction between AIDE and the user commonly involves the system presenting a set of alternatives for a given decision, along with documentation of each alternative and a set of recommendations. The user then gives AIDE a command:

Resume (Step): Carry out the selected alternative.

Resume (Jump): Similar to *Step*, but indicating also that AIDE can exercise more autonomy in subsequent, related decisions.

Back: Return to the decision immediately preceding the current one, for review or revision.

Forward: Converse of *Back*.

History: Revisit a decision on a path between the current decision and the starting point.

Navigate/refocus: Revisit a previous decision, chosen from a graphical display of all earlier decisions.

Other: Execute a specified statistical operation, possibly unrelated to the current state of exploration. This is a catch-all for situations in which the user selects and executes an action unanticipated by AIDE; the limited coverage of AIDE's plans thus does not constrain the user's activities.

AIDE is most effective with the user alternating between *Step* and *Jump*, instructing the system to proceed until it determines that an "interesting" result has been found, and continuing from that point. The system responds with the requested action, plus subsequent results that are potentially relevant. This provides lookahead that can sometimes help the user make better decisions. The other commands are used when AIDE's exploration diverges from what the user judges the best course of action. In these cases, the user can modify AIDE's behavior locally, with *Back*, *Forward*, and *Step* operations, or globally with the other options, consulting a navigational overview if desired. Figure 2 shows an overview, with the parent of the currently active focus point highlighted. The graph displayed is a tree, but need not be; different sequences of decisions may lead to the same point. Focus point alternatives, their justifications, data objects, and results can be selected to display appropriate documentation.

The navigation metaphor for EDA provides the benefit of making statistical decisions explicit. As Peter Huber puts it, "Data analysis is different [from word processing and batch programming]: the correctness of the end product cannot be checked without inspecting the path leading to it." (Huber 1994, p. 69) In AIDE one can easily trace a sequence of statistical decisions through the focus point network; these sequences can be generated for printed output as well. In contrast, the direct manipulation style of interaction of many statistical interfaces leaves most of these decisions implicit, while the programming environments of other types of interfaces force one to reconstruct a decision sequence from a linear trace of user actions, including errors, false starts, and backtracking.

The navigation metaphor further provides a natural organization for the decision process. Conventional graphical statistics packages can deluge the user with graphs, tables, and statistical summaries, an organizational structure provided only at the level of the windowing software. In our experience, users often need to make annotations recording the relationships between results and the procedures that generated them. This problem is somewhat alleviated in AIDE, in that results are linked explicitly and automatically to decisions, which are in turn related through the focus point network.

While it might appear that a navigation facility is an unalloyed benefit for a statistical system, it has at least one disadvantage. A conventional statistical interface gives the user a set of problem-solving tools.

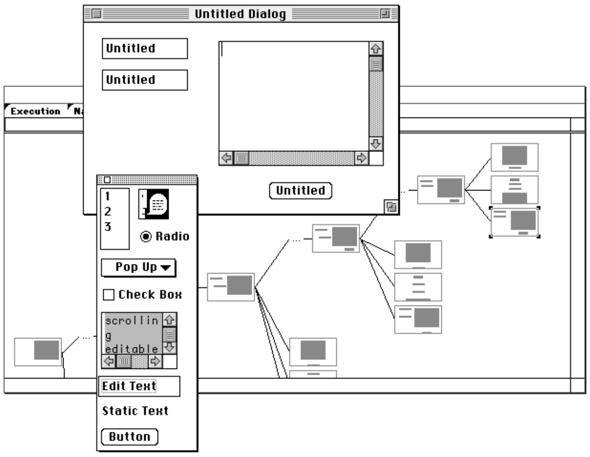

Figure 3: The navigational interface in NIFT

AIDE provides this same set of tools, *plus* a set of meta-level operations that potentially impose a larger cognitive burden on the user. This problem is well understood by hypertext researchers: the price of increased power and flexibility in browsing through an information structure is the greater potential for disorientation. Fortunately, in an empirical evaluation of AIDE (St. Amant & Cohen 1997b), we found that the use of local navigation operations did not reduce effectiveness; in fact, navigation was a significant positive factor in user performance for a data exploration task.

NIFT

With NIFT we intended to test the generality of both the AP planner and its navigational interface by attacking a very different domain. NIFT constructs layouts in single or multiple columns, with appropriate left or right justification, paying attention to screen space usage, balance, and related evaluation techniques. Strategies in NIFT are drawn from the literature of layout design; a good summary is given by Vanderdonckt et al (1994). The current implementation is functional but fragile; a full evaluation is planned for the summer.

For NIFT the existing navigational mechanisms were extended in two ways. First, we added simple display methods to help reduce the size and complexity of the navigation overview. There are methods for eliding long, non-branching chains of focus points, for condensing the display of focus point alternatives not on the path between the root focus point and the currently active decision, and for displaying focus points textually or graphically. Users can select the types of object (focus point, goal or plan instance, etc.) to be displayed, highlight sets of displayed objects by specifying type or value constraints, and mark specific objects as landmarks. Figure 3 shows a navigational overview, with focus points displayed as thumbnail layouts.

The other extension was to the set of navigational operations, in particular the provision of copy and paste methods. Copying and pasting are based on the notion that a sequence of decisions or operations may apply in more than one context. The operations act as a cross between visual programming and programming by demonstration, or PBD (Cypher 1993). It is visual programming in the sense that the user can directly select and manipulate a sequence of decisions; it has the flavor of PBD in that these decisions may not be an exact fit for a new situation, requiring inference on the part of the system. For example, one might select a vertical group of objects, left justify them, resize them to have the same width, and move the group to the left margin. These activities create a sequence of focus point decisions for the choice of each operation or parameterization. One can select this sequence and apply it without difficulty to a different group of objects, essentially defining an "after the fact" macro to be applied wherever appropriate.

The example above is straightforward. Suppose, though, that the user were to select a horizontally oriented group of objects, in order to align them by their top edges, resize them by height, and move the group flush against the top margin. The original sequence of operations is almost but not quite applicable. It would work only if the plan for alignment can distinguish vertically from horizontally oriented groups, and if it can reason that the default justification for these groups should be left and top, respectively. In addition, though the system can narrow the possibilities about object resizing and placement, there are still many plausible options. Macro-like sequences of focus point decisions cannot be automatically generalized to always do the right thing in a given situation. Nevertheless, there are benefits to a partial solution: the set of operations considered by the system is constrained; the start and end points of the sequence of operations are explicitly defined; changes can be made to a partially correct sequence of operations without requiring that the entire sequence be repeated; plausible alternatives are presented explicitly for review. All of these can be difficult issues for a PBD system but are ameliorated in a navigational framework.

Implications

Navigation operations are part of most user interfaces, though not always explicitly. Undo and redo operations, for example, fit naturally into the navigation framework. In Emacs one can review earlier actions with undo and redo operations, to restore an earlier state or simply to remind oneself of the thread of earlier work. Reverting a buffer corresponds to revisiting a root point in a hierarchy of editing decisions.

AP's navigational approach differs in two general ways from conventional user interfaces. First, decisions, and thus states, in the interaction process can be directly reviewed and retrieved, rather than in a sometimes tedious sequence of undo and redo operations. Past states can even be compared with one another. Second, the network of focus points provides a unified representation for both past decisions, made by the user or the system, and potential future decisions, i.e., those decisions made autonomously by the system to provide lookahead to the user.

Navigational mechanisms could benefit existing user interface technology in useful ways. To see how, it will be helpful to concentrate on a particular style of interaction; user interface wizards provide a good example.

The term "wizard" describes a style of interaction that has become popular in commercial user interfaces in the last few years. A wizard walks a novice user, step by step, through a common task. The interaction commonly takes place through a sequence of dialog boxes, each prompting the user for additional information, rather than in the standard interface. The user can move forward and backward through this sequence, and may quit at any time; at that point the wizard inserts an appropriate result into the application. An example is a letter-formatting wizard, found in some word processing packages. The wizard requests, in a fixed sequence, the type of letter desired (business, informal), the typeface, address information, and so forth. Once the user has entered the requested information, it is organized in the specified format.

Wizards have a good deal of untapped potential:

- Using a wizard doesn't help the novice figure out how to carry out the process without its help. One of the side effects of using a wizard is that the user may come to depend on it, and never be able to carry out the process on his or her own.
- Wizards often cannot be reinvoked to modify data structures they themselves have created, even to make trivial changes.
- A wizard generally does not make use of what the user has done up to the current point. The user must redo earlier work, responding to the wizard's requests for information.
- Wizards generally follow a fixed sequence, from start to finish, which can be inappropriate if a user wants access to just a single step. This is especially painful for users just leaving the novice stage, needing hints rather than instructions.
- Wizards implement a single approach to solving a problem. Consider a domain expert, unfamiliar with a software package, who turns to a wizard to produce a solution. The wizard follows its procedure, which differs from the expert's experience of how the problem should be solved. The result is confusion and reduced confidence in the system.

Each of these problems is addressed by AP. Assistance is tightly integrated into the interface; the system can follow up on the user's actions or its own; it can be "invoked" at intermediate points in a strategy; it can pursue several lines of reasoning simultaneously. One can give the system free rein, or examine its internal decisions in detail, an important feature in systems to be used by novices and experts (Bonar & Liffick 1991).

Conclusion

We have discussed a mixed-initiative planning system incorporated into two applications, one for exploratory data analysis, the other for layout design. A novel aspect of the interaction with these systems is the navigational interface, which as shown itself to be effective in formal and informal testing. Our approach is nevertheless far from a complete solution to the problems of mixed-initiative planning. The AP planner is relatively simple, applicable in domains with very specific characteristics: problems that involve incrementally constructed solutions, idempotent actions, automated evaluation, and at least partially codified expert knowledge. Navigation is furthermore part of a very traditional, command-driven style of interaction. In terms of flexibility and opportunism, AP suffers somewhat in comparison to dialog-based systems like TRAINS (Ferguson, Allen, & Miller 1996). We believe, however, that navigation mechanisms, as a supplement to other techniques, have a place in the guidance and understanding of the behavior of mixed-initiative systems.

Acknowledgments

This work owes a large intellectual debt to Paul R. Cohen. The comments of Martin Dulberg and three anonymous reviewers were also helpful, especially in focusing the discussion in this paper. Lisp code supplied by the developers of MCL and by Ralf Moeller and Voelker Haarslev made it possible to construct NIFT and the graphical portions of the navigational interface with relatively little effort.

References

Allen, J. F. 1994. Mixed initiative planning: Position paper. [WWW document]. Presented at the ARPA/Rome Labs Planning Initiative Workshop. http://www.cs.rochester.edu/research/trains/mip/.

Anderson, S. D. 1995. A simulation substrate for real-time planning. Computer Science Department Technical Report 95-80, University of Massachusetts at Amherst.

Bonar, J., and Liffick, B. W. 1991. Communicating with high-level plans. In Sullivan, J. W., and Tyler, S. W., eds., *Intelligent User Interfaces.* ACM Press.

Burstein, M. H., and McDermott, D. V. 1996. Issues in the development of human-computer mixed initiative planning. In Gorayska, B., and Mey, J. L., eds., *Cognitive Technology: In Search of a Humane Interface.* Elsevier Science. 285–303.

Carver, N., and Lesser, V. 1993. A planner for the control of problem solving systems. *IEEE Transactions on Systems, Man, and Cybernetics, special issue on Planning, Scheduling, and Control* 23(6):1519–1536.

Cypher, A. 1993. *Watch what I do : programming by demonstration.* MIT Press.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, 70–77.

Gale, W. A.; Hand, D. J.; and Kelly, A. E. 1993. Statistical applications of artificial intelligence. In Rao, C. R., ed., *Handbook of Statistics*, volume 9. Elsevier Science. chapter 16, 535–576.

Gale, W. A. 1986. REX review. In Gale, W. A., ed., *Artificial Intelligence and Statistics I.* Addison-Wesley Publishing Company.

Gallopoulos, S.; Houstis, E.; and Rice, J. R. 1994. Computer as thinker/doer: Problem-solving environments for computational science. *IEEE Computational Science and Engineering* 1:11–23.

Hand, D. 1986. Patterns in statistical strategy. In Gale, W., ed., *Artificial Intelligence and Statistics I.* Addison-Wesley Publishing Company. 355–387.

Huber, P. J. 1994. Languages for statistics and data analysis. In Dirschedl, P., and Ostermann, R., eds., *Computational Statistics.* Springer-Verlag.

Kim, H., and Hirtle, S. C. 1995. Spatial metaphors and disorientation in hypertext browsing. *Behaviour and Information Technology* 14(4):239–250.

Lubinsky, D., and Pregibon, D. 1988. Data analysis as search. *Journal of Econometrics* 38:247–268.

Myers, K. L. 1996. Advisable planning systems. In Tate, A., ed., *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative.* AAAI Press. 206–209.

Nielsen, J. 1990. *Hypertext and hypermedia.* Academic Press, Inc.

St. Amant, R., and Cohen, P. R. 1996. A planner for exploratory data analysis. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, 205–212. AAAI Press.

St. Amant, R., and Cohen, P. R. 1997a. Building an EDA assistant: A progress report. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics.*

St. Amant, R., and Cohen, P. R. 1997b. Interaction with a mixed-initiative system for exploratory data analysis. In *Proceedings of the Third International Conference on Intelligent User Interfaces.*

Vanderdonckt, J.; Ouedraogo, M.; and Ygueitengar, B. 1994. A comparison of placement strategies for effective visual design. In *HCI-94*, 125–143.