# Image Cover Sheet

| CLASSIFICATION | SYSTEM NUMBER |
| --- | --- |
| UNCLASSIFIED | 513544 |

**TITLE**

Testbed for Intelligent Aiding Using the LOCATE Workspace Layout Tool - Final Report

**System Number:**

**Patron Number:**

**Requester:**

**Notes:**

**DSIS Use only:**

**Deliver to:**

*This page is left blank*

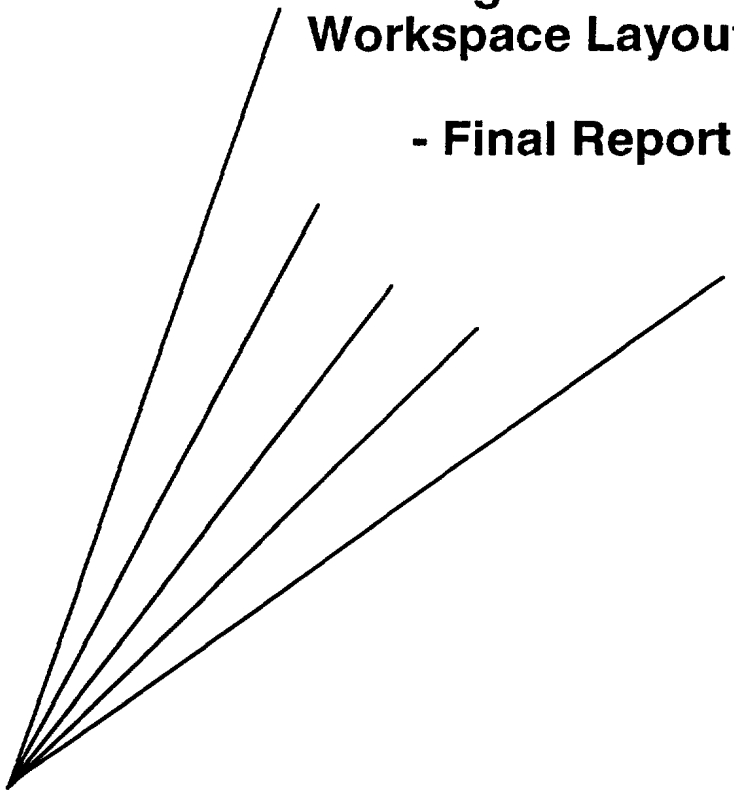
*This page is left blank*

Prepared for: Public Works and Government Services
Prepared by: Artificial Intelligence Management and Development Corporation
AIM (AC201, November, 1999)
Contract: W7711-9-7546/001/TOR
"Testbed for Intelligent Aiding Using the *LOCATE* Workspace Layout Tool"
Scientific Authority:
Mr. Keith C. Hendy
Simulation and Modelling for Acquisition, Rehearsal and Training (SMART)
Defence and Civil Institute of Environmental Medicine



# Testbed for Intelligent Aiding Using the *LOCATE* Workspace Layout Tool

## - Final Report -



Artificial Intelligence
Management and Development Corporation
206 Keewatin Ave., Toronto, Ontario M4P 1Z8

# Table of Contents

APPENDICES



FIGURES

## *Abstract*

The purpose of the work reported here was to develop generalizable principles and techniques for intelligent aiding useful to projects developed at Canada's Defence and Civil Institute of Environmental Medicine (DCIEM). In realizing that purpose, the principles and techniques of intelligent aiding implemented in DCIEM's LOCATE Workspace Layout Tool were combined with other, similar principles and techniques from related areas of research.

LOCATE is a CAD tool that allows users to create designs and analyze their communication efficiency. The LOCATE tool is serving as the focus for discussion, design and implementation in an emerging testbed for elaborating ideas on intelligent aiding. Earlier work on building an infrastructure to support such aiding in LOCATE was extended so that the tracking of all low-level user actions is now complete, as is the ability of LOCATE to infer the goals those actions might imply.

Further extensions included the identification and elaboration of categories of actions that imply a need for help. A category of "Non-Responsive Actions," or actions for which there is no system response, was selected for implementation. Help was designed and implemented, first of all, to inform users that such actions produce no response and second, to identify and help users achieve whatever goal(s) they may have had in mind when they performed those actions.

Task and help, actions and goals are represented in explicit models of task, user and system in LOCATE, and are displayed in associated windows in its interface. In the Task Model, an action and goal history is maintained as the user creates and analyzes designs. As that work proceeds, help of various kinds is initiated by LOCATE and, in a similar way, its System Model maintains an action and goal help history.

Several projects were identified as likely candidates for the application of the principles and techniques identified, but none was at a stage where such application was possible. Those and other projects will continue to be monitored for opportunities to incorporate intelligent aiding capabilities.

## Background

For several years, the Defence and Civil Institute of Environmental Medicine (DCIEM) has supported the development of *LOCATE*, a specialized Computer-Aided Design (CAD) tool for creating and analyzing workspace layout designs (Hendy, 1984, 1989).

Recent efforts (W7711-8-7476; W7711-8-7480) have focused on adding intelligent help and analysis to that system. Principles inherent in the work, however, have applicability to a much wider range of software, and many development projects within DCIEM potentially could benefit from a more detailed study and application of those principles.

Some of the fundamentals used in building intelligence into a product like LOCATE include:

- Tracking user interface actions and representing associated goals;
- Interpreting those actions and goals within a framework of separable models for task, user, system and dialogue (Edwards, in press; Edwards & Hendy, 1992; Edwards and Mason, 1988);
- Using those interpretations to aid users, especially new users, in learning the software and in performing the task for which the software was designed.

Almost all of the work to date has been in the context of providing intelligent aiding within the LOCATE workspace layout tool. Although considerable scope still exists for extending LOCATE and its intelligent aiding features, DCIEM would like to expand its focus by generalizing the principles that underlie that technology in ways that could prove useful to other development efforts at DCIEM.

The purpose of the work reported here was to examine the principles of intelligent aiding as implemented in the LOCATE workspace layout tool, to develop generalizations that could prove useful to one or more additional projects within DCIEM, to outline how those principles might be applied to at least one other existing DCIEM project and to begin the process of isolating intelligent aiding technology so that it might serve as a standalone pluggable module in other software.

As part of those efforts, LOCATE will function as a testbed for elaborating ideas on intelligent aiding. Some of the more practical ideas will be incorporated into LOCATE itself, but the primary focus of the testbed work is to clarify and extend intelligent aiding concepts that can serve a wider range of software development efforts. As the work progresses, other practical tools like LOCATE will be added to the testbed.

## Research Approach

The research approach for this project involved summarizing the techniques used to design and implement intelligent aiding in the LOCATE Workspace Layout Tool, developing generalizations of the principles they represent and implementing some additional, select intelligent aiding techniques into LOCATE.

Investigations were carried out in the related areas of Artificial Intelligence, Human Computer Interaction and Perceptual Control Theory to identify techniques and tools that illustrate useful principles for intelligent aiding. Work on standalone shells or pluggable modules as well as integrated intelligent aiding systems were of particular interest. Participation in the User Modelling (UM99) conference in which some of the above techniques and tools were discussed contributed to the present work.

These investigations along with the summary of LOCATE's intelligent aiding features provided input for identifying general principles that might be applied both to LOCATE and to other software projects at DCIEM.

To illustrate the general principles of intelligent aiding that emerged from this work, it was proposed that an overview be prepared outlining how those principles might be applied to a specific development project within DCIEM, other than the LOCATE project. Deciding on the target project would involve discussions with the Scientific Authority and other interested parties at DCIEM.

A key concern of this effort was determining how to create an intelligent aiding component that is at least partially decomposable from the software it supports. Thus, investigations into standalone shells and pluggable expert systems were considered particularly useful.

LOCATE is a mature tool that is now being used in practical applications. Its primary role in the present project was to contribute to an emerging testbed of knowledge about intelligent aiding techniques and how they may be applied more widely. That contribution was made through gaining a deeper understanding of LOCATE's current support for intelligent aiding and through new features, both experimental and practical, that were considered possible additions to the LOCATE application.

## STUDY OBJECTIVES

Study objectives for this project were:

- to review the current intelligent aiding principles and features that have been incorporated into the *LOCATE* Workspace Layout Tool;
- to investigate techniques and tools in the fields of Artificial Intelligence, Human-Computer Interaction and Perceptual Control that illustrate useful principles of intelligent aiding. A key area for investigation was that of User Modelling;
- to combine known principles from those fields with those used in the development of *LOCATE*;
- to explore how those principles might support the design and implementation of standalone, pluggable intelligent aiding components that could be generalized to a wide variety of software projects at DCIEM;
- to identify a preliminary set of requirements for building such a component or components;
- to incorporate some select intelligent aiding capabilities into *LOCATE*, based on those requirements;

# Intelligent Aiding in the LOCATE Workspace Layout Tool

## A Dual Approach to Intelligent Aiding

Intelligence in any entity, including a piece of software, is often a matter of perception. It is difficult, if not impossible, to know what is driving behaviour that is perceived to be intelligent, but the perception is often based on things like the novelty or complexity that is seen to characterize a set of actions. Much of the apparently intelligent behaviour of contemporary software, such as real-time correction of typing errors in a word processor, are little more than clever implementation techniques, informed by previously collected empirical data that indicate typical errors users make in typing.

As work to incorporate intelligence into the *LOCATE* Workspace Layout Tool proceeded, it became clear that two approaches to this general task are necessary. The first takes advantage of clever implementation techniques of the kind just described, where a designer is able to anticipate what will happen and provide for various system responses based on the details of what occurs. Such "intelligent" behaviour is largely a reflection of a designer's ability to deal in relatively fixed ways with contingencies involved in a user's interaction with software.

That fact should not detract, however, from the usefulness of those techniques. They can be effective in informing users about how a piece of software works, in building their confidence in the software to help them master its features and better perform the task at hand, and in predisposing users to want to work with the software.

Further, they serve as useful implementation features until a theory can be constructed to incorporate them and, they support the very construction of such a theory by providing examples that lead to useful generalizations.

The second approach to incorporating intelligence into LOCATE involves a more fundamental structuring of information *qua* knowledge for the system. If a piece of software is able to monitor actions taking place at the interface and represent those actions in ways that allow it to "understand" their meaning relative, say, to the overall task, then it has an infrastructure for supporting substantial and reasoned intelligent help to its users.

Once that representational groundwork has been laid, knowledge about what is happening at the interface can become part of a reasoning system for determining the kind of help to provide to a user, when such help should be provided and the form it will take.

Both the above approaches emerged from earlier contract work as strategies for building intelligent aiding into the LOCATE tool and both depend heavily on LOCATE's ability to monitor user actions at the interface.

## BUILDING AN INFRASTRUCTURE

Building an infrastructure to make intelligent aiding possible in LOCATE began with two of the three fundamental notions mentioned earlier:

- tracking user actions and representing associated goals;
- interpreting those actions and goals within a framework of separable models for task, user, system and dialogue.

Only brief descriptions of these two ideas are offered here; more thorough treatments can be found in several earlier reports.

### Tracking User Actions and Representing Associated Goals

Designing and implementing the tracking of user actions was part of several earlier studies:

- A usability study of LOCATE (W7711-6-7320) was the first example of tracking user actions and followed the activities of human factors experts as they worked with and assessed the LOCATE software. Special summarizing software tallied the results and prepared them for further analysis using spreadsheet software;
- A proof-of-concept study was conducted to demonstrate a few examples of intelligent help by linking existing LOCATE code with rules in the underlying expert system that forms part of the LOCATE development environment (W7711-6-7321);
- Two studies focused on building a more complete infrastructure for the support of intelligent aiding and analysis (W7711-8-7476; W7711-8-7480).

The last two studies added substantially to the infrastructure for intelligent aiding by implementing the tracking of nearly all user interface actions and the inference of goals implied by those actions. Goals inferred by LOCATE include both those of user and system. The latter are goals that must be achieved by the system in order for the user to achieve his or her task goals.

Work in this contract phase completed the tracking task so that LOCATE now infers goals for all user actions.[1] A summary outline of all code for the work done on LOCATE is presented in Appendix A.

Additions to the tracking task principally involved inferring goals from changes users make in text items within LOCATE windows. Those inferences are specific in that they preserve the content of the changes made as well as the prior values of variables. Appendix B illustrates the new LOCATE actions and their associated goals.

**Explicit and Separable Models of Task, User and System**

Most user actions that change data in some LOCATE window relate to the task of creating and analyzing a workspace design. Tracking those changes results in changes in LOCATE's Task Model, which are displayed in a Task Model Window as actions and goals of the user and system.

Figure 1 shows an attributes window for one of several elemental workstations, the basic units in a LOCATE design. Changes have been made to the workstation's position (X and Y) and to its angle of rotation. A further change has been made to the angle of the workstation's Source/Receiver (S/R) node.

As indicated, such changes by a user are accompanied by the recording of the interface actions and associated goals in LOCATE's Task Model and by their display in its Task Model Window. Figure 2 shows the state of the Task Model Window after the changes in Figure 1 have been made.

---

[1] This is true for all LOCATE task related actions. It is also true for all currently implemented system-initiated help related actions. It does not apply to what might be termed, "meta-actions," i. e., user actions that involve accessing the Task, User or System Models or to any actions performed while those Windows are active.

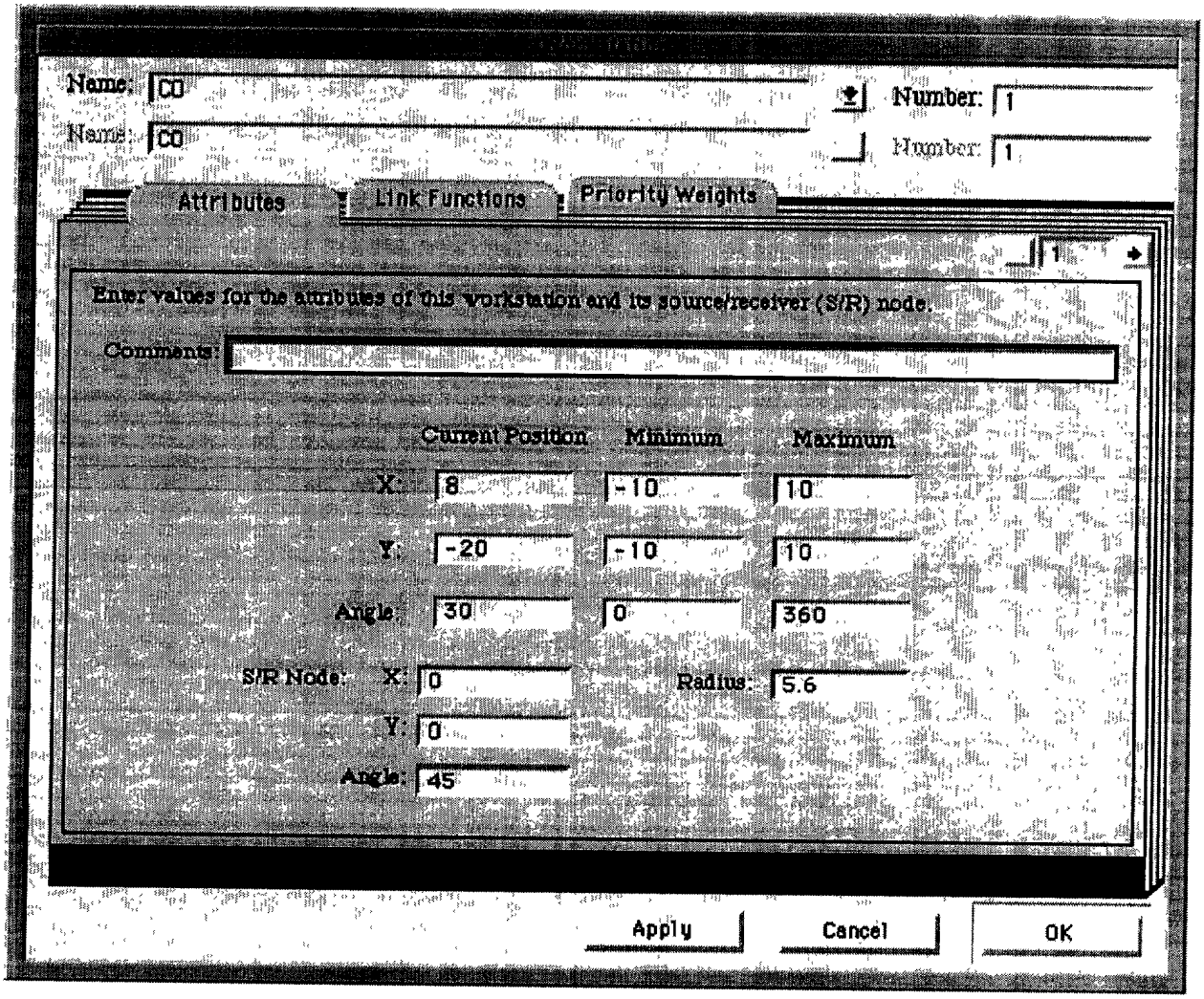



Figure 1. Text Changes in a Workstation Window

The Task Model Window is divided into three parts:

- Latest Action;
- Current Goals;
- Action and Goal History.

The *Latest Action* section of the window shows the last user interface action performed. The last action in this example is, "Workstation: Text changed in the S/R angle text box", which is LOCATE's way of indicating that the user changed the angle of the Source/Receiver (S/R) node of the selected workstation, in this case the Commanding Officer's (CO's) workstation.

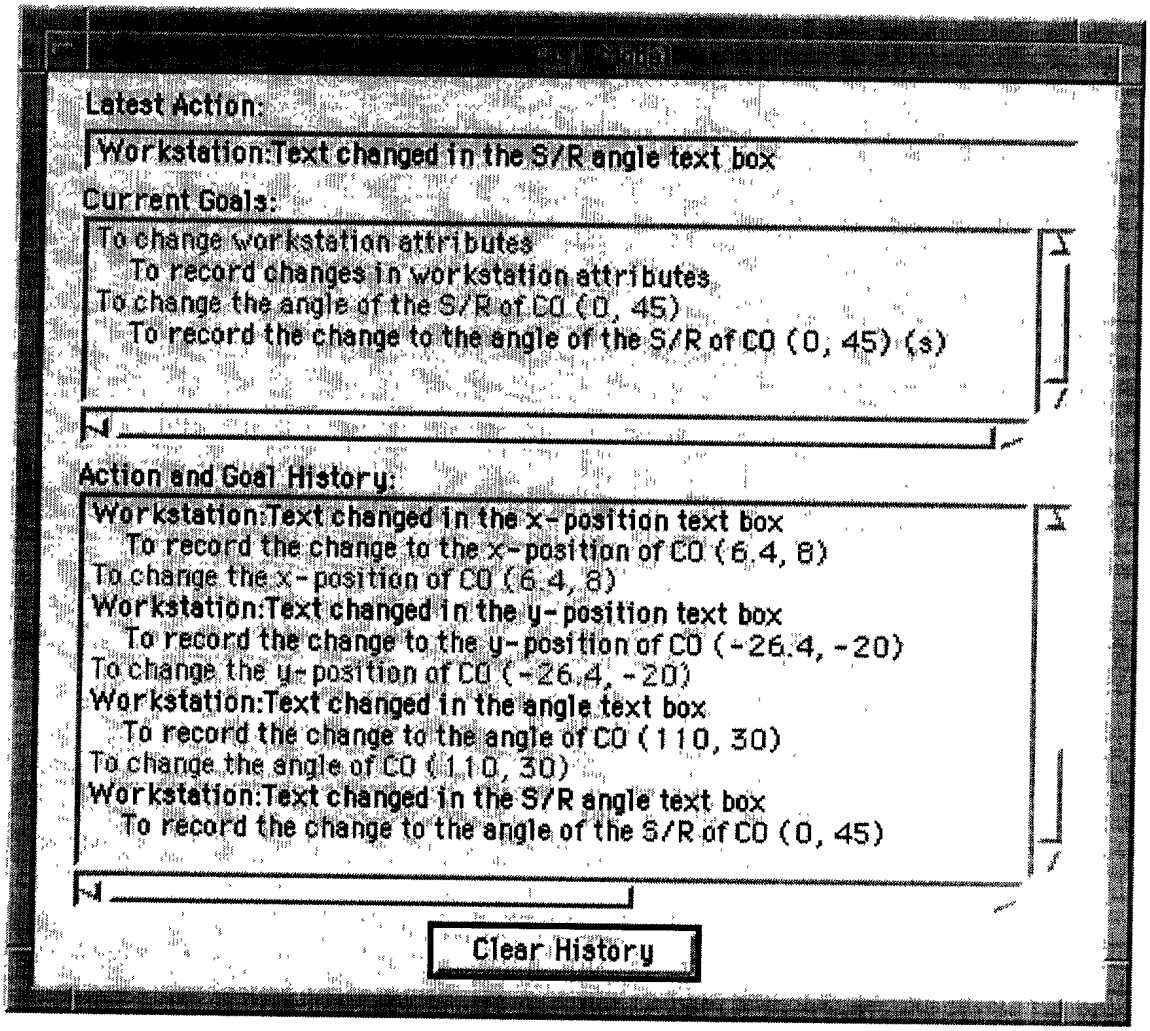



Figure 2. LOCATE's Task Model Window Showing Changes to Workstation Attributes.

The second section in the Task Model Window is the *Current Goals* section, which displays a list of the goals currently being pursued. These are goals that LOCATE has inferred from the user's actions. In this example, there is a high level goal, "to change workstation attributes" that appears at the top of this section of the window and which is followed by the most recent and more specific user goal of changing the angle of the S/R node in the CO's workstation.

Two numbers in brackets (0, 45) follow the low-level goal entry. They are the respective values for the rotation angle before and after the user made the change. System support goals to record the change made by the user appear indented under each of the two user goals. Those goals are necessary to the completion of the user goals. All goals are

colour-coded with user goals appearing in red and system goals in blue; user actions are recorded in black.

Both user and system goals in this example are followed by an "(s)," indicating that they have been satisfied. Even though the goals have been satisfied, they remain in the Current Goals portion of the window until the user performs some new action, at which time they are removed to an *Action and Goal History,* which is displayed at the bottom of the Task Model Window.

Since most goals transferred to the Action and Goal History have been satisfied, the trailing "(s)" is removed to reduce clutter in the window. The less frequent, unsatisfied goals are distinguished by adding a "(-)" after them when they are placed in the history portion of the window.

In tracking the changes made to text items in a window, LOCATE infers changes and associated goals even if the user has not clicked the OK or Apply buttons in that window. In such cases, LOCATE's inference appears to be a little misleading.

If the user makes several changes and then clicks the Cancel button, none of those changes are recorded. If that happens, the more general goal of changing attributes for the selected workstation will appear in the Action and Goal History as "cancelled," i. e., with a trailing minus (-) sign.

The Action and Goal History provides an "audit trail" of everything the user and LOCATE have done during a session. Any summarizing software that might be constructed to review what has been done during a session will be able to determine that "apparently satisfied" goals, indicating changes made to values in a LOCATE window, were actually cancelled and therefore not satisfied.

The advantage to handling goals in this way is that all of the user's actions are preserved in the history and, although it might prove somewhat confusing for someone who does not understand how goals are recorded, it provides a way of preserving a more complete record of what the user has done.

## PROVIDING HELP TO LOCATE USERS

Currently, four types of help are available to users of the LOCATE workspace layout tool:

- an on-line presentation outlining LOCATE's benefits and features;
- a quick start, hardcopy user manual;
- on-line and local hypertext help files;
- intelligent aiding based on tracking user actions and goals and on maintaining separable models of task, user and system.

The on-line presentation is the simplest introduction to LOCATE and can be found at the following demonstration Web site:

http://www.interlog.com/~jle/Presentation.html

The presentation was created by editing a previously prepared Powerpoint presentation file of LOCATE and using Macromedia's Director software to convert it to an online demonstration.

The second type of help is a quick start user manual which provides a summary introduction to LOCATE and is delivered to users along with the application. It describes key benefits and features that help users quickly apply LOCATE to their design problems.

The on-line hypertext help files were created from over two hundred identified information categories for LOCATE and contain more than 50 hypertext pages and over 100 graphics files. The help files may be accessed locally or on-line at the following demonstration web site:

http://www.interlog.com/~jle

Recent work converted the thesis on which LOCATE is based to electronic form and it is a goal of the LOCATE work to make some or all of that thesis available as part of LOCATE's hypertext help.

**Intelligent Aiding Through the Use of Models**

The second of the two key items necessary to building an infrastructure for the support of intelligent aiding is the creation of models for task, user and system as separable knowledge sources. How do these models support intelligent aiding in LOCATE?

*Task Model.* Information in LOCATE's Task Model provides a means for understanding what the user is doing at any given point in the creation and analysis of a design. It also is a basis for understanding the kinds of supporting activities LOCATE performs in helping a user achieve his or her goals.

*User Model.* The LOCATE User Model contains information about who the user is, his or her attributes and what the user knows about LOCATE. This last type of information is critical to intelligent aiding and is obtained in three ways:

- direct statements by a user about what he or she knows;
- observation of user actions and the consequent inferences;
- help information provided to the user by LOCATE.

Currently, some minimal information is obtained directly from the user. If the user wants LOCATE to track his activities, he needs to identify himself through his name, initials or some other type of identifier. At the time this information is provided, the user can ask LOCATE to track his activities and provide him with feedback. This information is obtained directly from the user at startup or when he selects "About You..." from the Help Menu.

With respect to observation and inference, LOCATE is able to track a few user activities and provide some clever help. One example from an earlier study is LOCATE's ability to know whether the user is opening windows by double-clicking on some object or by first selecting the object and then using a menu item selection to open its attributes window. If the latter, then, after several instances of menu selections, LOCATE puts up a help window telling the user about the more effective double-click method of opening windows.

Knowledge about the user is derived not only from observations such as those just described but also from the results of help information provided to the user by LOCATE. In the above example, after putting up the help window, LOCATE can conclude that the user likely knows how to open attribute windows using the double-click method.

Displaying the help information in this window and obtaining the user's answer as to whether he or she knows that double-clicking on a workstation opens its window means that LOCATE can now draw inferences about what the user likely knew and what he now knows and can store results of those inferences in the User Model.

An example appears in Figure 3. Colour-coding is used here in a similar way to its use in the Task Model Window. LOCATE's comments and inferences are in black, help provided appears in blue and user responses are colour-coded in red.

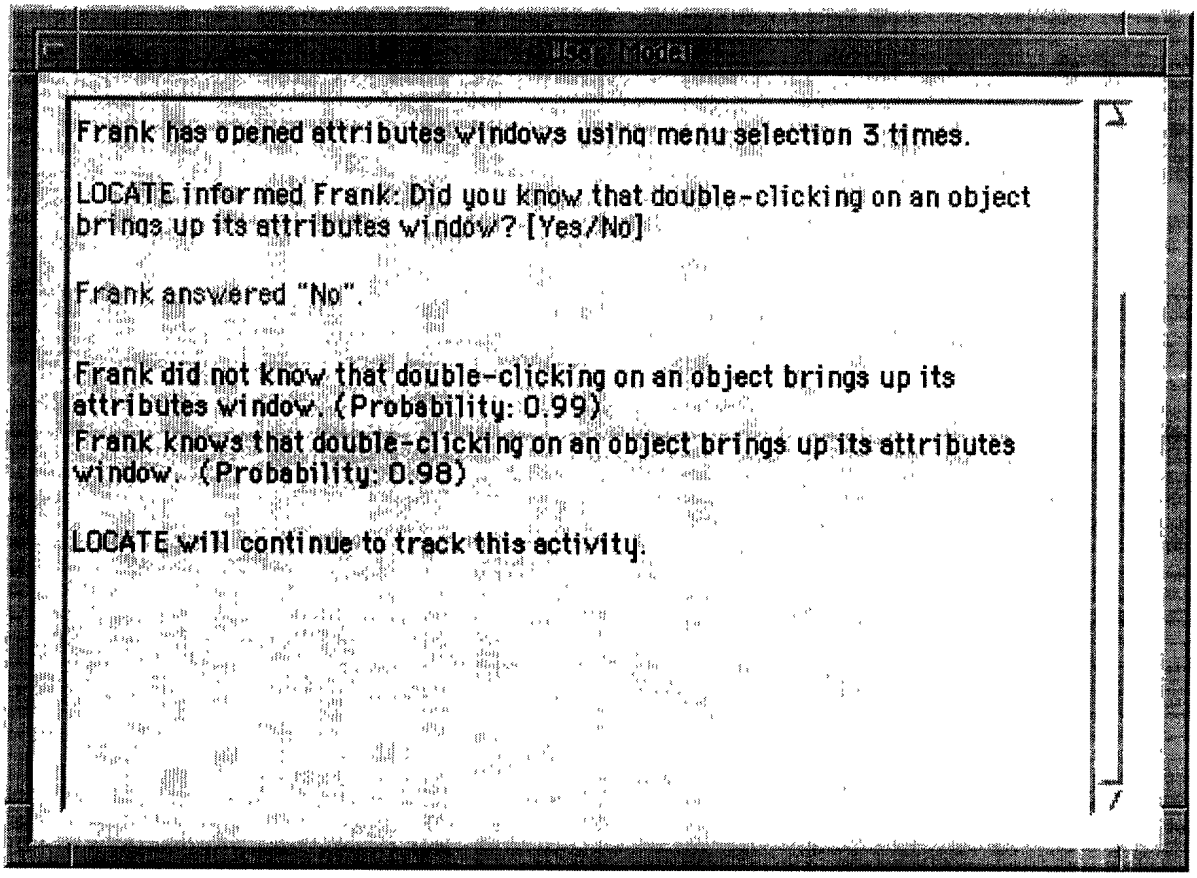



Figure 3. Example Information Appearing in LOCATE's User Model Window Following a User Response to a Help Question.

Detecting subsequent use of menu item selections by a user, in the example just described, will now have different implications for help than it did previously. If the user continues to use a menu selection method to open attribute windows, LOCATE will be led to the conclusion that this user prefers that method over the more efficient, double-clicking method.

*System (Self) Model.* In addition to using knowledge stored in its user model for intelligent aiding, LOCATE also makes use of information stored in its System (Self) Model.

The knowledge a system can have about itself includes knowledge (beliefs) about its own capabilities. The primary reason for creating such a model is to support the users of the software, and so, the knowledge contained in this model should be organized around helping users better understand the system's features. In LOCATE this is accompanied by linking the System Model to a Smart Help option in the Help Menu.

The help LOCATE currently offers to users is goal oriented. An example appears in Figure 4 on the next page. The window there is displayed to a user whenever she selects "Smart Help" from the Help Menu. Smart Help is organized around a concept of "How to...."

In the example, the user has asked LOCATE "How to print" and LOCATE has responded by displaying a list of the goals (more accurately, intentions) it understands are associated with the concept of "print." That list is displayed in a panel entitled, "Goals" on the left side of the window.

The list provides a user with choices that focus her inquiry. When she chooses one of the options in the list, a set of subgoals (a plan) appears on the right which outlines how she can achieve the goal she selected.

In the example, the user has selected "To print a design" and the system has displayed a list on the right that shows the subgoals *qua* actions necessary to print a design.

Notice that the query from the user is in terms of an action, the high-level response from LOCATE is characterized in terms of goals and the consequent plan is framed in a panel entitled, "Actions necessary to achieve the selected goal." Those actions are a mixture of subgoals, conditionals and "primitive" actions, among other things, but are constructed in a way to make sense to the user.

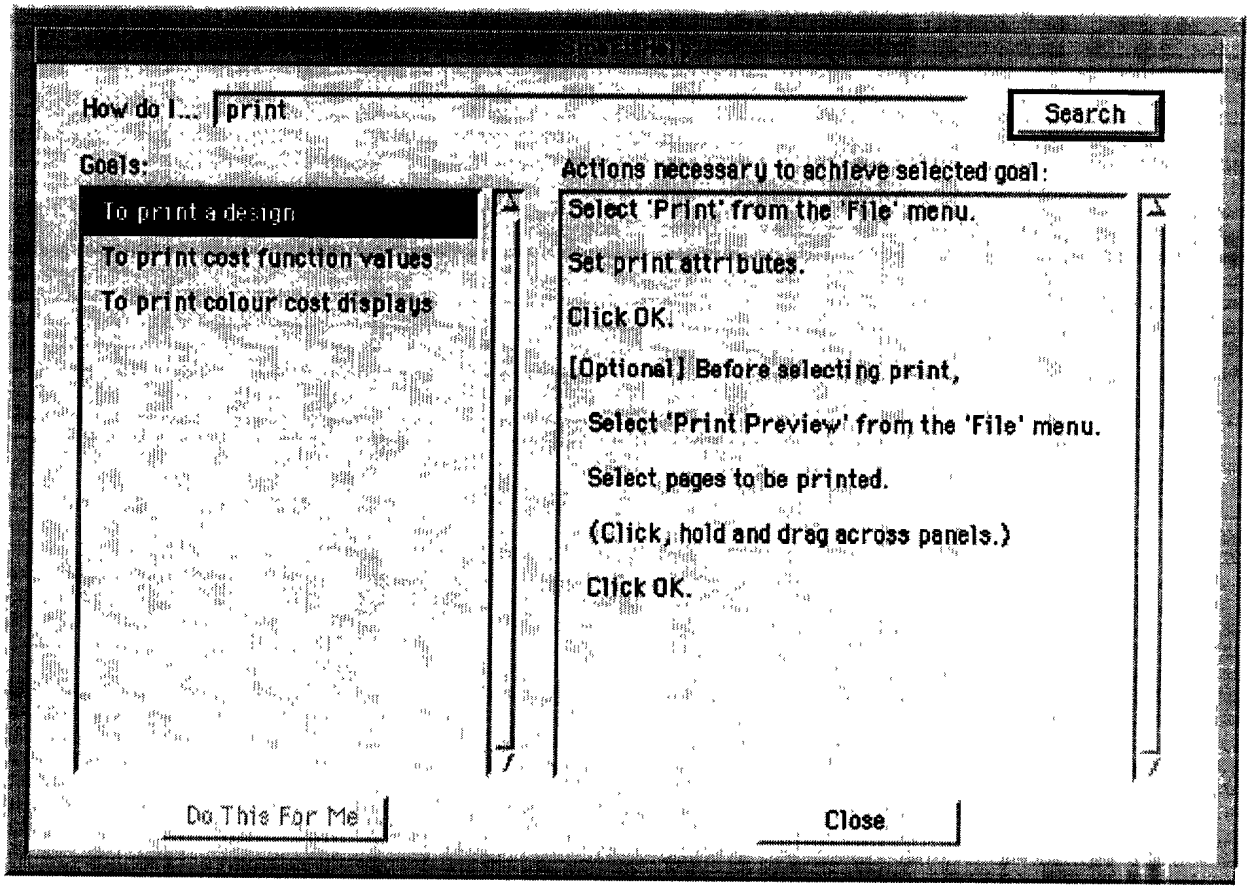



Figure 4. Example of LOCATE's Smart Help Window

Just as the task model reflects an understanding of what is happening in the interface through goal and plan recognition, LOCATE's help and analysis facilities support the user through a process of goal and plan generation. For recent work elaborating intelligent help as plan generation, see Küpper and Kobsa (1999).

As the user requests and receives help from LOCATE, the system's (self) model tracks what is taking place. Figure 5 shows the consequences for the system model of the interaction that has just occurred in the Smart Help Window.

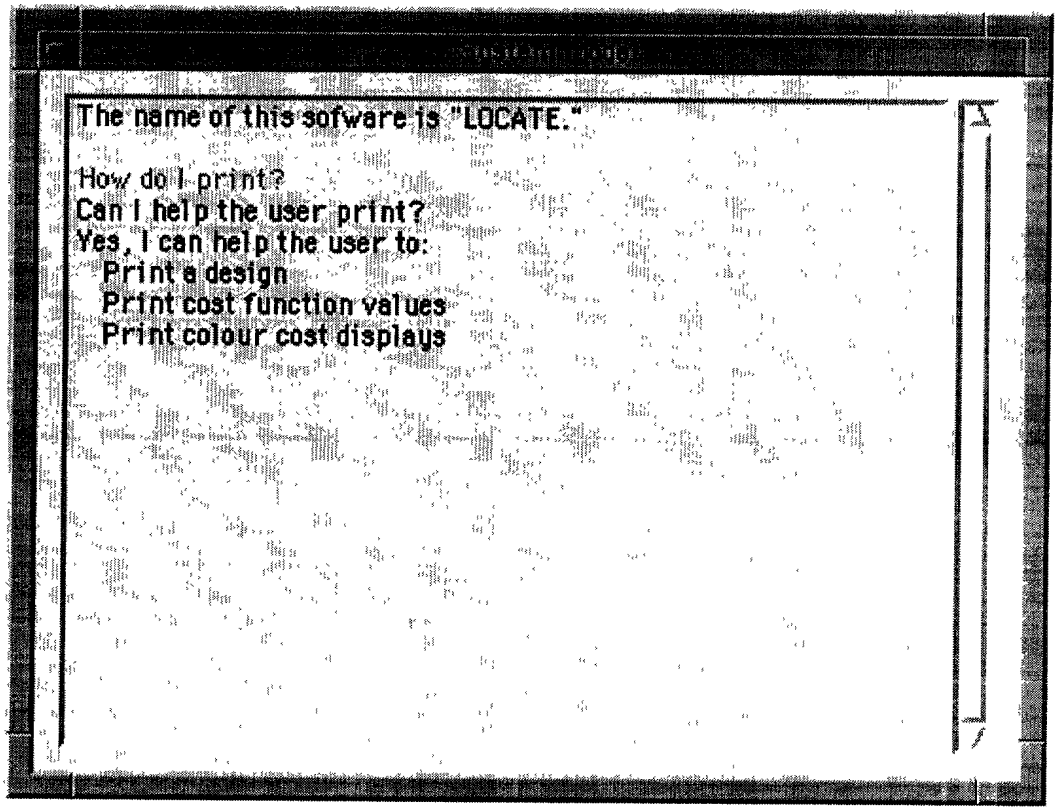



Figure 5. System (Self) Model as a Consequence of User Query in Figure 4.

The first entry in the window is a standard entry placed there at startup and simply identifies the system (in its own self model) to the user: "The name of this software is LOCATE."

The next set of entries reflects the user's request for help and LOCATE's response. These entries are constructed in a way that is meant to reflect a sense of self enquiry on the part of the system, relative to whether it is able to help the user with some identified problem.

This approach may have several advantages. First, it provides an intuitive way for a developer to track the system's exploration of its own knowledge and capabilities relative to user inquires. It also provides the possibility for a developer to request greater detail about that exploration at any point in the process.

For a user who might wish to view the system model, it communicates that the central purpose of self enquiry on the part of the system is to help the user better achieve his or her

goals. Also, it is likely that both the content and nature of such a self-enquiry approach will be non-threatening to users in the context of a system known to keep track of its own knowledge and capabilities.

Much of the information in this section was presented in earlier reports and was repeated here to provide a background for the new work on intelligent aiding now being conducted. That new work is in the context of building a testbed for the study of intelligent aiding and, as such, will extend beyond the specific work on the LOCATE workspace layout tool.

In the next section, a preliminary set of criteria for intelligent aiding of a more general nature is described. Areas of opportunity for providing intelligent help are identified, some of which have formed part of the implementation effort on LOCATE in this contract. Those areas of opportunity are meant to be a first step in generalizing the principles of intelligent aiding and analysis to other development efforts that might profit from their application.

# Identifying and Generalizing Principles of Intelligent Aiding

Beyond the additions of some intelligent aiding capabilities to LOCATE, a key purpose of the current work was to examine the principles implicit in such aiding and to combine them with others identified in the literature of several related areas, e. g., AI, User Modelling, Human-Computer Interaction and Perceptual Control Theory.

The material in the next few sections is an amalgamation of work done on LOCATE with that taken from some of those sources. The first section presents a preliminary list of criteria for intelligent aiding. Section two is a first step in developing principles for intelligent aiding that may be generalized beyond the LOCATE workspace layout tool. Section three identifies three projects at DCIEM to which principles might be applied. Finally, section four includes brief comments relating the work on the intelligence testbed to work in the area of Perceptual Control Theory, an area of particular interest to DCIEM.

## CRITERIA FOR INTELLIGENT AIDING

In the process of trying to understand principles of intelligent aiding, several criteria have emerged that will provide guidance to its application across a variety of software systems.

The criteria are derived in part from accepted principles that characterize any good scientific theory as well as those for good interface design. They represent a first step in developing a way of critiquing intelligent aiding systems and techniques. As such, they are not meant to be complete and are offered more as a basis for discussion and refinement than as a coherent and well-integrated set of guidelines. As DCIEM's testbed for intelligent aiding evolves, these principles should play an important role in providing direction and ensuring coherency. They are offered here in outline form:

A. Elegance and Simplicity of the Help Interface
- Minimal Obtrusiveness of Help Features into the Interface
  - *Exceptions*. Some software may be programmed for very active system participation in the task, which might involve taking over aspects of the task when certain circumstances arise or informing the user of critical information in a very direct way.

- Support for Multi-Featured Help in an Easy-to-Use, Intuitive Interface

B. Consistency of Content and Form of Help Presentation

C. Completeness of Help Offered

D. Relevance of Help Offered
   - High-Priority (High-Relevant) Help Communicated Simply, Effectively and Directly, i. e., at a time and in a form determined by the system;
   - Medium-Priority (Medium-Relevant) Help Communicated Simply, Effectively but Indirectly, i. e., at a time and perhaps even in a form determined by the user;
   - Low-Priority (Low-Relevant) Help Identified but not Communicated.
     - This help is held within the system until its priority has increased to the point where it qualifies as medium or high priority help, or until a specified amount of time passes, when it is purged from the help queue.

These categories cover different types of system-initiated help. By definition, user-initiated help is high priority.

E. Contextual Appropriateness
   - Relative to the Current Task
     *Task Goals & Plans currently pursued by User and System.*
   - Relative to the User
     *Adjusted for the User's Attributes.*
   - Relative to the System
     *Adjusted for on System Features.*
   - Relative to the User-Computer Dialogue
     *Adjusted for the Task and Help Related Dialogues.*

F. Demonstrable Increase in Effective Use of the Application
   - Demonstration of the Effectiveness through Usability Studies.

## PRINCIPLES AND SUPPORTING ELEMENTS FOR INTELLIGENT AIDING

In order for a system to provide truly intelligent help to its users, a number of key elements are required:

- an infrastructure that allows the system to track actions and infer goals;
- methods for gathering information about users, directly through information provided by the user and indirectly through tracking the interface actions;
- an ability to identify activities that imply a need for help;
- an infrastructure for providing help in the interface;
- an ability to recognize conditions under which adaptation is to occur and to "understand" the actions appropriate to those conditions;
- a facility for maintaining a history of user task actions and goals;
- a facility for maintaining a history of user help actions and goals;
- a facility for maintaining a history of user knowledge;
- a facility for capturing feedback from users about the help that has been provided;
- some ability to learn.

Although the material is more detailed than that in the previous section, it is still in the early stages of development and is intended more to provide direction to the evolving testbed than to be a complete set of guidelines for designing and implementing intelligent aiding.

Where possible and appropriate, material will be presented in general terms so that its applicability to a wider range of software is apparent. In many cases, however, it is restricted to the LOCATE tool, which is currently the core application out of which the testbed is emerging. In spite of the specific nature of the material, it still is easy to see its implications for a wider range of software.

Two general elements required for intelligent aiding are detection and the process of aiding itself. Detection is possible to the extent that there is an infrastructure in place within the software for detecting a variety of information that is both practical and useful to aiding. The infrastructure for detection should contain the following elements:

- User Attributes
  - Personal: Identification; preferences and interests.
  - Knowledge and proficiency relative to the task for which the software has been designed;
  - Knowledge and proficiency relative to the software [LOCATE].

- Tracking User Actions
  - All user interface actions should be tracked [Now true for LOCATE];
  - Types of actions that imply help should be identified [Some true for LOCATE].

- User Goal and Plan Recognition
  - User task goals should be inferred for all user actions [Now true for LOCATE]
    - System Support Goal and Plan Identification
      Support goals should be inferred for all user goals [Now true for LOCATE].

Describing the infrastructure in terms of content to be detected forms only part of a set of requirements necessary for intelligent aiding. It is also necessary to specify how that information is to be detected. In the case of LOCATE, personal user attributes are acquired directly through an "About You" window along with information about the user's knowledge and proficiency in the task of workspace design. The "About You" Window is accessed through an item in LOCATE's Help Menu.

A user's assessment of his or her level of knowledge and proficiency with LOCATE is obtained in the same manner, and also through the tracking of the interface actions as the user works with the software.

A third type of user information is obtained as a result of help provided by LOCATE. When help is given, LOCATE can conclude, with some level of certainty, that the user now knows the information provided. LOCATE is aided in its conclusions by asking users to indicate whether they have read the help information. This not only provides a slightly better foundation (than the standard OK and Cancel buttons) for conclusions about what a user knows but also justifies decisions about when the help information might be presented to the user again and in what form.

The outline below shows user attributes acquired by LOCATE, many of which will prove useful in other applications. The outline also gives indications ("-->") as to how LOCATE acquires the attribute information and some idea of what it does with it.

- *Detecting User Attributes*
  - Personal: Identification; preferences and interests.
    - Name (or Identifier);
      --> How acquired: "About You" Window
      --> How used: put in User Profile (User Model)
    - [No entries for personal preferences or interests in LOCATE at the moment.]
  - Knowledge and proficiency relative to the task of workspace design;
    - Proficiency with Workspace Design;
      --> How acquired: "About You" Window
      --> How used: put in User Profile (User Model)
    - Task-related interests could be obtained directly or indirectly by:
      - tracking information accessed from military standards sites such as the human factors guidelines;
        --> How acquired: [not yet acquired.]
        --> How used: put in User Profile (User Model)
      - monitoring information accessed through various design sites;
        --> How acquired: [not yet acquired.]
        --> How used: put in User Profile (User Model)
      - tracking the (types of) designs worked on by a user.
        --> How acquired: tracked through LOCATE's Cost Function History.
        --> How used: put in User Profile (User Model)
  - Knowledge and proficiency relative to the [LOCATE] software.
    - Proficiency with LOCATE software;
      --> How acquired: "About You" Window
      --> tracking use of LOCATE's features.
      --> How used: put in User Profile (User Model)

*LOCATE Preferences*

<u>Preferences currently incorporated into LOCATE</u>

- User enters name (or other identifier) or chooses from a list of users at startup. [If only a single user, an option for automatic log-in could be provided.]
  --> How acquired: Startup Window

--> How used: put in User Profile (User Model)
put in a LOCATE Users Profile

- User asks LOCATE to track activities;
  --> How acquired: "About You" Window
  --> How used: put in User Profile (User Model)
- User asks LOCATE to provide feedback on activities (intelligent aiding);
  [In future, level (or amount) of help may be specifiable by user.]
  --> How acquired: "About You" Window
  --> How used: put in User Profile (User Model)
- User asks LOCATE for reminder of activity tracking and intelligent aiding at startup;
  --> How acquired: Startup and "About You" Windows
  --> How used: put in User Profile (User Model)

<u>Suggested preferences NOT yet incorporated into LOCATE</u>

- A "Preferences" Menu Item is to be added to the bottom of LOCATE's Edit Menu. The associated Preferences window will allow a user to specify a set of simple preferences.
  An "Advanced Preferences" button will appear on the Preferences Window to allow more experienced users to set specialized preferences.
  --> How acquired: Preferences Window
  [The "About You" Window will contain a link to the Preferences Window.]
  --> How used: put in User Profile (User Model)
- Revert to default preferences.
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Specify default object attributes (e.g. size);
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Specify default link functions; priority weights.
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Specify default workspace attributes;
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)

- Specify default font for text tool
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Specify auto-save, auto-creation of a backup file
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Allow keyboard selection of palette tools
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Indicate preferred web browser;
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Specify e-mail address;
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Allow expert users to turn off certain alerts, e. g., confirmation when dragging an object outside a workspace;
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)
- Allow expert users to automatically commit priority weights;
  --> How acquired: Preferences Window
  --> How used: put in User Profile (User Model)

*Use-of-LOCATE Characteristics (not yet implemented)*

- Total time spent using the software;
  Demonstration scenario: if there is no activity for 2 minutes, LOCATE subtracts that time ("Idle Time") from the total time the application has been open, then resets the counter for another 2 minutes.
  Qualifying points: A user may be reading a manual, thinking, drawing on paper, examining files related to his design work, etc., however, this still does not constitute active use of the software.
  Times recorded, both within and across sessions:
  - "Total Time Open" (or, total time application has been open)
  - "Total Active Time Use" ("Total Time Open" minus "Idle Time")

  --> How acquired: background tracking by LOCATE
  --> How used: put in User Profile (User Model)

- Designs accessed, when accessed and whether modified (Yes/No);
  This will be threaded with designs recorded in the Cost Function History, which maintains a list of all designs opened during a session. One wrinkle occurs when a design is recalled, worked on for a period of time and then saved under another name.
  --> How acquired: tracking by Cost Function History
  --> How used: put in User Profile (User Model)
- Help accessed: Includes hypertext help or third-party help via Internet [it would only be possible to track this if a provision can be made for accessing help on the Internet, e. g., what specific standards information has been accessed at some web site];
  --> How acquired: not yet specified
  --> How used: put in User Help History
- Help accepted by User that has been offered by LOCATE (see below);
  --> How acquired: tracking system-initiated help
  --> How used: put in User Help History

- *Detecting User Actions and Goals*

In addition to detecting user attributes, it is clear by now that LOCATE also tracks user actions at the interface. One of the important consequences of tracking user actions has been the ability to infer and record the goal implied by an action. Other goals inferred by LOCATE now include system support goals necessary to the achievement of the user's goal.

A key consequence of tracking interface actions is the ability it provides for helping users. That begs the question, however, of what types of action imply a need for help. During the current work phase, a number of action categories were identified that qualify as a partial answer to that question. Those are described in outline form below along with indications of how the information is to be acquired and used to support intelligent aiding.

- *Detecting Actions that Imply a Need for Help*

<u>Contradictory Actions</u>

- These are apparently contradictory actions. For example, if LOCATE believes a user knows something yet observes him acting in ways that imply he does not, there is a need to clarify the apparent contradiction.

The situation arises after a user has been given help on some LOCATE feature and then acts in a way that implies he has forgotten, not understood or even read the help provided.

Additional help may be offered if the system believes the user should know the information as originally presented.

--> How acquired: tracking user actions and comparing them to knowledge in the User Model.

--> How used: put up appropriate Help Window

--> put in User Profile (User Model)

--> put in Help History

--> modify knowledge in the User Model

Non-Responsive Actions (Actions that have no effect[2])

- In LOCATE this type of action occurs when the user (control) (double) clicks on one of the following object types in a workspace:
  - a link;
  - a rotation arrow in a workstation or obstruction;
  - the Selection Box at the top of the palette;
  - a dimmed Cost Function Value Button, located at the bottom of the palette;
  - a dimmed "Save CF" button;
  - a ruler in the Design Window;
  - a Source/Receiver (S/R) node;
  - a text box;

  --> How acquired: tracking user actions

  --> How used: put up appropriate Alert.

  --> put in Help History (User Model)
- Other actions of this type occur when the user tries to resize any of the following non-resizable objects:
  - an S/R node;
  - a text box;
  - a complex grouped object (includes a workstation or an obstruction).

  --> How acquired: tracking user actions

  --> How used: put up appropriate Alert

  --> put in Help History (User Model)

[2] other than perhaps an audio signal indicating that the user action produces no response by the software.

During the current work phase, detection of all the above actions was implemented. The contents of the Alert windows that provide the user with information about why such actions do not produce a response from the system appear in Appendix C.

The Alerts function something like LOCATE's Smart Help Window. Part of their job is to identify possible goals a user may have had in mind when he performed the non-responsive action and to help him understand how he can use LOCATE to achieve that goal in other ways.

Non-Actions (UnUsed LOCATE Features)

- The System (Self) Model contains knowledge of its own capabilities. That list will be prioritized based on decisions about the importance of features. Those features will be compared periodically to the ones used so far by the user, in order to identify the features about which the user might be informed.

  --> How acquired: tracking the use of LOCATE features and comparing them to all available features

  --> monitoring the user's "Active-Use Time" with LOCATE to determine when to present what features;

  --> How used: at specified points in user's "Active-Use Time" with LOCATE, provides medium priority help informing user of unused features. [The user informed of 2 or 3 features at a time with the option of seeing more. If the user chooses not to see other items, they are shown at a later time.]

  --> put in User Profile (User Model)

  --> put in Help History

  --> Examples: provide user with a library of sample objects with pre-configured Link Functions, Obstruction Functions and Priority Weights, e.g. human, computer, radar screen, speaker, etc.

  Control-Shift-Click on a "prototype" EW (or other object). That brings up a window that allows the user to create an arbitrary number of instances of the prototype.

Prohibited (Invalid) Actions

- These are actions prohibited by the system. There may be different types within this category and some may overlap with the Non-Responsive Actions, described above. Currently, Prohibited Actions are being enumerated with a view to the emergence of a more complete description.

  Two examples are: 1) entering invalid data into an edit text box (data of the wrong type or out of some acceptable range); and, 2) attempting to run a cost function without having specified domain weights and having "committed" priority weight values.

  --> How acquired: tracking user actions

  --> How used: put up Alert with a description of the acceptable range of values.

  --> put in Help History (User Model)

Repetitive Activities

- Actions that a user performs over and over, either in rapid succession or within some determined interval. Examples include:
  - Repeatedly (double) clicking on some object that produces no response. [Most often, this will be a repeated (double) click after help has already been given for a Non-Responsive Action (see above). The distinction here is that Non-Responsive actions produce help after the first (double) click, so that it is impossible for a user to (double) click repeatedly before help is offered the first time. If LOCATE knows the user has read the associated help, yet, at a later time, he repeatedly (double) clicks on the same object, then that would indicate that the user may have forgotten or not understood the earlier help.];

  --> How acquired: comparing user help triggering actions to help already offered.

  --> How used: put up appropriate Alert.

  --> put in Help History (User Model)
- User repeatedly clicks on a palette tool, creates an instance of an object, clicks again on the same tool, creates another object, and continues in this way for some specified number of instances within some specific time frame.

  --> How acquired: tracking user interface actions

  --> How used: put up Alert suggesting the use of one of two multiple-object creation modes:

  1. double-clicking on palette icon to allow manual creation of multiple instances of the object type;

2. shift-clicking the palette tool to bring up a window that asks how many instances of the particular object type the user would like to create.

--> put in Help History (User Model)

- User repeatedly calculates a cost function using a menu selection or keyboard shortcut.
  --> How acquired: tracking user interface actions;
  --> How used: put up Alert that suggesting the use of the Cost Function Value Window, located at the bottom of the palette
  --> put in Help History (User Model)

- User repeatedly calculates a cost function using Cost Function Value Button.
  --> How acquired: tracking user interface actions;
  --> How used: put up Alert that suggesting setting the calculation of cost functions to "Automatic" mode in the Cost Function Window.
  --> put in Help History (User Model)

- User repeatedly enters the same Link Function or Priority Weight data for several workstations.
  --> How acquired: tracking user interface actions;
  --> How used: put up Alert that suggests one of two approaches:
    1. creating a prototype workstation and duplicating it;
    2. changing LOCATE's default link functions.
  --> put in Help History (User Model)

- User repeatedly uses menu item selection to open an object's attributes window.
  --> How acquired: tracking user interface actions;
  --> How used: put up Alert suggesting more efficient use by double-clicking object.
  --> put in Help History (User Model)

- User repeatedly clicks on objects obscured by other objects.
  --> How acquired: tracking user interface actions;
  --> How used: put up Alert suggesting use of "Send to Back" command;
  --> explain object hierarchy and when to use control-clicking

  Alternatives:
  --> provide the user with a list of objects currently obscured and give a choice of which object to bring to the front;
  --> provide a "Let me try it" button, allowing the user to try out bringing objects forward or sending them backward using the Control-Click command.
  --> put in Help History (User Model)

- Repeated use of some subset of palette tools.
  --> How acquired: tracking user interface actions;
  --> How used: suggest re-organization of palette based on tool frequency use

  Alternatives:
  --> automatically move most frequently used tools to top of palette; (to the left in a horizontal palette);
  --> at some suitable time, allow user to choose to reorganize the palette [reduces confusion that occurs if system dynamically changes tool positions in the palette];
  --> allow "Revert to a default" listing of palette items.
  --> put in Help History (User Model)

- *Detecting Actions that are Explicit Requests for Help*

  - User selects "Help" in Help Menu
    --> How acquired: detects user's explicit request for help.
    --> How used: puts accessed help information in Help History (User Model)
  - User double-clicks Keyboard icon
    --> How acquired: detects user's explicit request for help.
    --> How used: LOCATE displays a window showing all keyboard shortcuts;
    --> put in User Profile (User Model)
    --> put in Help History

- *Displaying User Actions and Goals*

In outlining how LOCATE acquires user attributes and tracks user actions above, the descriptions included how the information is acquired and how it is used. User attributes and tracked user actions are rarely represented directly in help windows presented to a user. They do appear, however, in LOCATE's Model windows, some examples of which were illustrated in figures presented earlier in this report.

Currently, users have access to those Model Windows and to all of the information stored in them. They may access any of the windows through LOCATE's Help Menu, as described below:

- *Task Model*

  The Task Model Window can be displayed by selecting "Task Model" from the Help Menu. The window contains the latest user action, current goals inferred from that action and an action and goal history for the user's session. [see the example in Figure 2, above]

- *User Model*

  The User Model Window can be displayed by selecting "User Model" from the Help Menu. It shows LOCATE's conclusions about who the user is, the user's preferences, his or her general skill level in workspace design and in the use of LOCATE, and knowledge about specific features of the LOCATE tool. [see the example in Figure 3, above]

- *System (Self) Model*

  A System Model Window can be displayed by selecting "System Model" from the Help Menu. This window is structured much like the Task Model Window but with a focus on the help offered to the user in contrast to the task being performed. The top portion of the window displays the latest help triggering action; the middle portion shows the help goals inferred from that action; and, the bottom portion is an Action and Goal Help History.

  Other contents of this window include LOCATE's knowledge of its own features in the context of how those features might help the user perform a given task

more effectively. [see the example in Figure 5 above, for an early implementation of this Window]

The outlined material above focused on the detection of information by LOCATE. It also gave some indications about how that information is used and the infrastructure necessary to support that use. The next several outlines describe that infrastructure in somewhat more detail.

- Goal and Plan Generation
  - The approach to intelligent aiding in LOCATE is plan generation. When LOCATE determines that help is needed, a help goal is instantiated which, in turn, produces a plan for providing that help. Both help and task goals in LOCATE are complex C++ objects. Appendix B to an earlier report (W7711-8-7480) provided specifications for the abstract goal class being used.

- Prioritizing Help Information
  - Three levels of help priority are identified in LOCATE:
    - High priority - Alerts are immediately displayed by LOCATE
    - Medium Priority - Light bulbs appear in a LightBulb Window and the user decides if and when he will view the help
    - Low Priority - the potential for providing the user with help on a certain topic is noted internally by LOCATE. No help is displayed to the user until the priority level for this help reaches the medium or high priority help categories.

- Interface Help Mechanisms
  - Alerts (High Priority)
    - Alerts are used for high priority help items, e.g., Non-Response Actions in LOCATE;
    - A user should have a way to retrieve the last Alert for a review of the help given [see Record-Keeping in Intelligent Aiding, below];

- Light bulbs (Medium Priority)
  - Light bulbs are standard indicators of help used in a wide variety of software applications;
  - LOCATE's light bulbs are displayed in a small window that appears on the right side of the Design Window's Menu Bar whenever a medium priority help has been generated for the user;
  - When the user clicks on a light bulb, a help window is put up displaying the help information.
  - No more than three light bulbs are displayed at any one time in the window.
- Other alternatives being considered:
  - A help information line at bottom of the Design Window;
  - "Tooltips" pop-up balloons, e. g., for additional information about palette items beyond just the names provided in the Selection Box;
  - An "Assistant" window similar to the Assistants in MS Word;
  - A rule regarding "Don't Show Again" buttons in help windows.
    If clicked, are there conditions where the system might invoke that help again?
  - Administered questionnaires to identify user attributes, e.g., interests and personal preferences of a user;
  - Entertaining mini-quizzes about what the user knows about LOCATE. This could take advantage of features incorporated in intelligent tutors.

- Record-Keeping: A User's Help History
  - A record is maintained of LOCATE-initiated help provided to the user, along with a date and time stamp; user initiated help has not been addressed in this context;
    - Help triggering actions by the user, e.g., performing a non-responsive action like double-clicking on a link, lead to the instantiation of help goals, which in turn create plans for what help is to be provided, when and how.
    - As help goals are carried out and satisfied, they are removed to an Action and Goal Help History. The Help History becomes part of the User Model.
    - The approach is similar to that employed in the Task Model where task goals are pursued, satisfied or cancelled, and then placed into an Action and Goal (Task) History.
    - Both Histories provide audit trails of Help and Task goals, respectively.
  - Help currently is organized chronologically in the Help History;
  - It is envisioned that a user will be provided with a way to access his or her help history. This will involve accessing information stored in the User Model;

- The Help History should retain contextual information but it is unclear how much context is necessary. The amount of context will be dependent on the particular purpose for recalling the history;
- The Help History also will identify help offered but not accepted. This would include high priority help that was "Not Read" and medium priority help, mediated through light bulbs, which was never accessed.

- Other Possible Infrastructure Mechanisms
  - Adapting the interface to particular users or to a given user under differing circumstances;
  - User-initiated allocation of tasks to LOCATE. This will involve an explicit re-assignment of some task to the software normally performed by the user, c.f., the as yet non-functional "Do it for me" button on LOCATE's Smart Help Window;
  - System-initiated tasks, with prior user consent specifying conditions under which the system is to take over the tasks;
  - Collaboration with LOCATE on tasks or on learning about the software.
  - System learned "macros" that permit LOCATE to perform the same or similar tasks in future, e. g.,
    - LOCATE tracks user actions:
      --> under certain conditions, it generates macros allowing it to repeat those actions in future;
      --> generalizes from actions with a view to performing similar actions. [Comment: Easier said than done.]

A key part of an infrastructure for providing intelligent aiding involves displaying the help information. Several interface mechanisms, already described, are elaborated in somewhat more detail below.

- *Displaying Help Information*

**Help Windows**

- Some consideration is being given to Alerts implemented as modeless windows that disappear after a few minutes [time to be dependent on the amount of information in the Alert.]

  Alerts and other help windows are given a default, "Not Read" button and clickable "Read" and "Don't Show Again" buttons. A user's choice provides slightly more confidence that the user has read or has not read a piece of help information and that helps the software to build reliable beliefs about what the user knows.

  [Note: The words, "Read" and "Not Read", are not entirely suitable, since "Read" can be interpreted as present tense.]

**History of Help Window (see Record-Keeping, above)**

- Some preliminary work has been done on an Action and Goal Help History Window, which is to contain:
  - help offered to the user by LOCATE.
  - help accessed by the user;
- The window is invoked by a "Help History" Menu Item in the Help Menu;
- Eventually, information in the Help History Window will be coded so that a user will know:
  - the specific help information offered;
  - its priority;
  - whether and when it was accessed;
  - when it was "Read" (or "Not Read");
  - whether the user chose the "Don't Show Again" option.

**Light Bulbs**

- Light bulbs signal the availability of secondary medium priority help. As help is offered, a light bulb will appear in the window. When the user clicks on the light bulb, the associated help will be displayed?
- As more help is offered to the user, up to three light bulbs will appear in the LightBulb window.

- When a light bulb is clicked and the help displayed, the light bulb will be removed from the Light Bulb window;
- The user will be allowed to clear all light bulbs from the LightBulb Window;
- Light bulbs remaining in the LightBulb Window will be cleared at the end of a session and the help associated with each placed in the User's Help History Window and identified as medium priority, unaccessed help.

**Keyboard**

- A keyboard icon will appear at the top of the Menu Bar to the right of the Light Bulb Window.
  - Double-clicking the icon will cause the display of a window showing all keyboard shortcuts.

- *Upgrading Intelligent Aiding in LOCATE*

Information useful in upgrading LOCATE's help system may be acquired in part by allowing users to provide feedback on how effective the current help is and what other kinds of help might be useful. Some points being considered are:

- Giving a user opportunities to provide feedback to the system (developers) about the help they receive from LOCATE.
  The feedback will be accessed by having users send it periodically by email or by having it sent automatically when a user logs onto the LOCATE web site, or onto coordinated sites, e.g., sites providing human factors guidelines and standards;

- Contextual information for any help given to a user may be necessary to understand the feedback a user provides. Depending on how help is structured, however, such contextual information may or may not be necessary;

- A more advanced concept is help that is modified "on the fly." [A simple example is the "Don't Show Again" button prevalent in the help windows of many contemporary applications]

- LOCATE resets the time when help, already shown to a user, will next be displayed. This provides another simple way of modifying help based on experience with particular users.

- *Some Miscellaneous Items*

A few miscellaneous items were identified for possible incorporation into the evolving testbed.

- Having the user choose the degree of interactivity with the various LOCATE models. For example, a window could be provided where interested users could review their User Model, and modify some of its contents;

- Assigning a higher weight to recently performed actions when making decisions based on observed user activity;

- Drawing inferences about user interests based on group membership (e.g. engineers working on a team might be interested in the same types of design). Group membership may be incorporated as part of a User Model. Inferences could be drawn as follows: if users A and B share interests m, n, and o, and user A has interest p, then user B may also have interest p;

- Identifying principles that serve the creation of standalone help modules. These principles will hopefully emerge from the current approach of keeping all intelligent aiding code localized in the LOCATE application.

As a final thought, in developing generalizations from LOCATE to software in other domains, generalization principles themselves may emerge from future work on intelligent aiding. Detecting such principles will be one of the items of business as the testbed evolves.

## APPLYING PRINCIPLES OF INTELLIGENT AIDING TO OTHER SOFTWARE PROJECTS

One of the goals of the current work was to identify one or more projects at DCIEM to apply the principles and techniques generalized from LOCATE and gleaned from other research on intelligent aiding.

Unfortunately, no project was currently available that would permit such an application. Three projects were identified, however, that will continue to be monitored with that goal in mind:

- the F-18 Incremental Modernisation Program;
- the Tactical Battlefield Command System;
- the Navy Command and Control System.

## COMMENTS ON THE TESTBED AND PERCEPTUAL CONTROL THEORY

Of some interest to DCIEM is how the evolving testbed for intelligent aiding might be coordinated with concepts in Perceptual Control Theory (PCT).

The concept of goal is central to Perceptual Control Theory as well as forming a key part of the work in the emerging testbed. As has been illustrated throughout this report, goals are inferred from user actions and play an important part in offering help to users.

Somewhat more questionable is the concept of **plans**. The next work phase on the intelligence testbed will seek to incorporate a more general notion of goal into both LOCATE and the testbed work and, in addition, examine plan generation and recognition with a view to incorporating those concepts in more extensive ways.

As the concept of plan does not appear to have a corresponding construct in Perceptual Control Theory, an investigation was conducted to determine how it is addressed, if at all, within PCT.

William Powers, the major proponent of Perceptual Control Theory, talks of goals at each level in his eleven-level hierarchy of perceptual control. His 6th level ("Relationship") and 8th level ("Sequence") clearly seem to lay a foundation for a concept of plan and his 9th level ("Programs") completes that work. The definitions below are taken from descriptions of those levels that appear in the work of Robertson and Powers (1991):

- **Relationship** ($6^{th}$ Level):

  the constraint of two otherwise independent perceptions (events), such that their behaviour is not totally independent, e.g., a dog chasing a cat; tick coming before tock; a cup containing coffee; etc.

- **Sequence** (8th Level):

  familiar short single or repetitive series of elements (elements drawn from any lower-order perceptions). This involves the **ordering** of discrete elements, e.g., A,B being different from B,A.

- **Programs** (9th Level):

  Sequences are specific orderings of symbols. At this level, "symbols and sequences are perceived in terms of a network of relationships similar to those found in computer programs....The essence of a program is a test followed by a choice...There can be multi-way branches."

It seems clear from the definitions of these three levels that the notion of plan is consistent with the writings on Perceptual Control and seen most clearly at the levels described above, especially the level of Programs.

How plans are translated into actions as they are realized across Powers' levels is hinted at in the two levels of *Sequence* and *Relationship.* This observation should not detract, however, form the fact that the word, "plan" does not actually appear in the theory.

More consideration needs to be given to whether the construct could have a more formal role in dealing with design and implementation issues based on Powers' hierarchy. The question that needs pursuing is, "How might plans be accommodated not only "horizontally" within Powers' layers, as seems clearly possible at his Programs level, but also "vertically" across his layers, which seems less certain from his writings.

# Summary

Groundwork for intelligent aiding in the LOCATE workspace layout tool was extended during this contract. The principal accomplishments were:

- changes to values in all of LOCATE's windows are now tracked and associated goals inferred.
- actions that change those values and their implied goals are displayed in a Task Model Window;
- work on intelligent aiding within LOCATE was reviewed and principles and supporting elements illustrated through its software features were combined with those taken from other content areas as a first step in developing a testbed for the general study of intelligent aiding;
- a preliminary set of criteria for designing and implementing intelligent aiding was developed;
- issues surrounding requirements for intelligent aiding were explored and a set of informal requirements constructed;
- action categories areas were identified that imply a possible need for help to which LOCATE and other software might respond with appropriate types of aiding;
- select aspects of intelligent aiding were implemented in LOCATE, the primary example being the presentation of Alerts following Non-Responsive Actions, i. e., actions that typically produce no functional response by the software.

# Future Work

The next step in the development of DCIEM's testbed for intelligent aiding is to expand the LOCATE goal-oriented approach to include more general forms of plan generation and recognition.

Currently, goals are inferred directly from the user's actions along with system support goals. Those goal inferences are fixed and follow directly from the user's actions. There is in fact little ambiguity in inferring goals from the specific user interface actions in LOCATE.

Future work will examine research in the field of plan recognition with a view to incorporating more general recognition techniques into LOCATE and, more generally, into the intelligence testbed.

In a similar way, intelligent aiding is viewed as a form of plan generation. System initiated aiding begins with a goal to provide help to a user about some capability. In the current work phase, some help goals were identified and some plans were implemented as a means of realizing those goals.

As with plan recognition, part of the next phase of work will be to develop a more general plan generation facility for LOCATE. As that facility is developed, it will be tailored in a way to make it applicable to other software applications.

Although several development projects at DCIEM were identified as likely candidates for the application of intelligent aiding principles from this testbed, none were at a stage that would permit application in the short term.

Those and other development projects will continue to be monitored as work on the testbed proceeds. In the meantime, new principles of intelligent aiding will be designed and implemented for the LOCATE workspace layout tool in ways that will permit them to be generalized to other software development efforts as the opportunity arises.

# References

Bass, E.J., Ernst-Fortin, S.T., and Small, R.J. (1997). Knowledge base development tool requirements for an intelligent monitoring aid. *Proceedings of the Tenth Annual Florida Artificial Research Symposium (FLAIRS-97)*, Daytona Beach, FL, May 12-14, pp. 412-416.

Broadbent, G. (1988). *Design in architecture.* London: David Fulton Publishers.

Edwards, J. L. & Mason, J. A. (1988). Toward intelligent dialogue with ISIS. *International Journal of Man-Machine Studies, 28*, 309-342.

Edwards, J. L., & Hendy, K. (1992). Development and validation of user models in an air traffic control simulation. Paper presented at the Second International Workshop on User Modeling. International Conference and Research Center for Computer Science (IBFI), Dagstuhl Castle, Germany, August 10-13, 1992.

Edwards, J. L., & Hendy, K. (1999). A Testbed for Intelligent Aiding in Adaptive Interfaces. Paper to be presented at the 1999 AAAI Spring Symposium on Adaptive User Interfaces.

Edwards, J. L., Scott, G.L, & Hendy, K. (1999). The LOCATE Workspace Layout Tool as a Basis for Building an Intelligence Testbed. Paper submitted to the Journal of User Modeling and User Adaptive Interaction.

Hendy, K. C. (1984). 'Locate': A program for computer-aided workspace layout. Master's Thesis, Department of Electrical Engineering, Monash University, Clayton, Victoria, Australia.

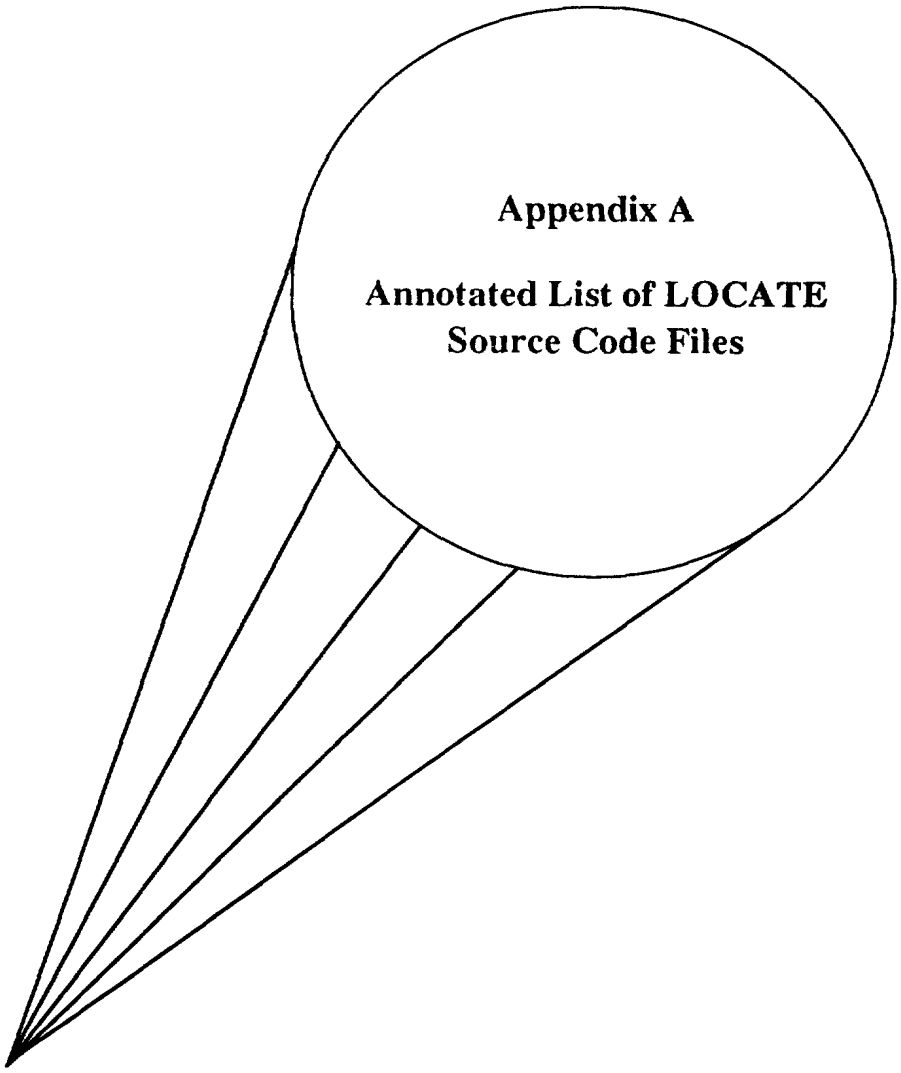
Hendy, K. C. (1989). A Model for Human-Machine-Human Interaction in Workspace Layout Problems. *Human Factors, 31(5),* 593-610.

Polson, M.C., & Richardson, J.J. (Eds.) (1988). *Foundations of intelligent tutoring systems.* Hillsdale, N.J.: Lawrence Erlbaum.

Powers, W. T. (1973) *Behavior: The control of perception.* New York: Aldine De Gruyter.

Richardson, R. J., & Powers, W. T. (Eds.) (1990) *Introduction to modern psychology: The control-theory view.* Gravel Switch, KY: The Control Systems Group.

Simon, H. (1981). *The sciences of the artificial.* 2nd edition. Cambridge, MA: MIT Press.



# Appendix A

## Annotated List of LOCATE Source Code Files

Artificial Intelligence
Management and Development Corporation

**Notes:**
i) .c files are C source code
.h files are C header files
.rc files are Open Interface resource description files
ii) Some extraneous files in the Locate folders are outdated and will be erased in future.

**AEVENT.C**
- This file contains the code to handle Apple Events for opening documents by double-clicking in the Finder.

**ALLOBJINFO.C, ALLOBJINFO.RC**
- The "AllObjInfo" module contains the code and Open Interface resources necessary for the "All Objects Info" window.

**AOBS.C, AOBS.RC**
- The "AObs" module contains the code and Open Interface resources necessary for the "Fixed Obstruction" window.

**ASSIGN.C**
- Original Locate C file

**CFALERT.C, CFALERT.RC**
- The "CFAlert" module contains the code and Open Interface resources necessary for the alert box that appears when there have been changes to the design but no cost function has been run since those changes have been made. The alert box appears before displaying the Cost Function History window .

**CFBROWSE.C, CFBROWSE.RC**
- The "CFBrowse" module contains the code and Open Interface resources necessary for the "Cost Function History" window.

**CFCHECK.C, CFCHECK.RC**
- The "CFCheck" module contains the code and Open Interface resources necessary for the "Cost Function Checks" window.

**COSTCOLR.C, COSTCOLR.H, COSTCOLR.RC**
- The "CostColr" module contains the code and Open Interface resources necessary for the "Cost Display Editor" window.

**COSTDISP.C, COSTDISP.H, COSTDISP.RC**
- The "CostDisp" module contains the code and Open Interface resources necessary for the "Cost Display" window.

**COSTFN.C, COSTFN.RC**
- The "CostFn" module contains the code and Open Interface resources necessary for the "Cost Function" window.

**DRAWROTD.C, DRAWROTD.H**
- C code for handling the drawing of rotated objects

**DXF.H**
- Header file with DXF format constants

**DXFOPT.C, DXFOPT.H**
- The "DXFOpt" module contains the code and Open Interface resources necessary for the "DXF Import Options" window.

**EDITOR.C, EDITOR2.C, EDITOR.RC**
- The "Editor" module contains the code and Open Interface resources necessary for the main Locate window (includes code for Diagrammer, palette, rulers).

**EVAL1.C**
- Original Locate C file

**EWATTR.C, EWATTR.H, EWATTR.RC**
- The "EWAttr" module contains the code and Open Interface resources necessary for the "Workstation" window, which now includes separate tabbed sections for Attributes, Link Functions and Priority Weights.

**EXTERN.H**
- Original Locate header file

**FORMAT.H**
- Original Locate header file

**FUNCT1.C**
- Original Locate C file

**GOALOBJ.CPP**
- Contains code for defining and handling the C++ goal object

**HEADER.DXF**
- Contains information that gets added to all exported DXF files

**HELPALRT.C, HELPALRT.RC**
- The "HelpAlrt" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows.

**HELPMORE.C, HELPMORE.RC**
- The "HelpMore" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows containing the "More" button.

**IMPRTDXF.C**
- C code for handling the importing of a workspace from DXF format

**INFOUPD.C, INFOUPD.H, INFOUPD.RC**
- The "InfoUpd" module contains the code and Open Interface resources necessary for the "Information Update" window.

**INFOWIN.C**
- C code for handling the "Object Info" window

**LFSUMM.C, LFSUMM.RC**
- The "LFSumm" module contains the code and Open Interface resources necessary for the "Link Function Summary" window.

**LINKDISP.C, LINKDISP.RC**
- The "LinkDisp" module contains the code and Open Interface resources necessary for the "Link Display" window.

**LOCATE.C**
- Based on the original LOCATE.C file, this contains the code necessary for loading in a workspace and for computing the cost function.

**LOCATE.DAT**
- Open Interface compiled resources that are used by the Locate application at run-time.

**LOCATE**
- The Locate application

**LOCATE.H**
- Original Locate header file

**LOCATE.μ**
- Locate project for CodeWarrior 11

**LOCATE.RC**
- Open Interface resources in text format

**LOCNEW.C**
- C code for handling the creation of a new workspace

**LOCNEWEW.C**
- C code for handling the creation and deletion of workstations and obstructions

**LOCSAVE.C**
- C code for handling the saving of a workspace

**LOCSAVEDXF.C**
- C code for handling the saving of a workspace in DXF format

**MAIN.C, MAIN.RC**
- The "Main" module contains the "main" function which starts up the application.

**MISC.C**
- Original Locate C file

**MISCRSRC.RC**
- The "MiscRsrc" module contains Open Interface resources needed by the application (primarily menu and icon resources).

**MULTIOBJ.C, MULTIOBJ.RC**
- The "MultiObj" module contains the code and Open Interface resources necessary for the "Multiple Object Creation" window.

**NEWUSER.C, NEWUSER.H, NEWUSER.RC**
- The "NewUser" module contains the code and Open Interface resources necessary for the "About You" window.

**OPT.C**
- Optimizer code for changing positions and angles

**OPTIM.C**
- Original Locate C file

**OPTOPT.C, OPTOPT.RC**
- The "OptOpt" module contains the code and Open Interface resources necessary for the "Optimizer Options" window.

**OPTSET.C, OPTSET.RC**
- The "OptSet" module contains the code and Open Interface resources necessary for the "Optimizer Settings" window.

**OPTSTAT.C, OPTSTAT.RC**
- The "OptStat" module contains the code and Open Interface resources necessary for the "Optimizer Status" window.

**OPTSWAP.C**
- Optimizer code for swapping workstations

**ORIGIN.C**
- Original Locate C file

**OTHEROBJ.C, OTHEROBJ.RC**
- The "OtherObj" module contains the code and Open Interface resources necessary for the "Other Object" window.

**OUTPUT.C**
- Original Locate C file

**PALEDIT.C, PALEDIT.RC**
- The "PalEdit" module contains the code and Open Interface resources necessary for the "Palette Editor" window.

**PRINTPREV.C, PRINTPREV.RC**
- The "PrintPrev" module contains the code and Open Interface resources necessary for the "Print Preview" window.

**RULEEW.C, RULEEW.H, RULEEW.RC**
- The "RuleEW" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Workstation attributes.

**RULEGEN.C, RULEGEN.H, RULEGEN.RC**
- The "RuleGen" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up object attributes.

**RULEOB.C, RULEOB.H, RULEOB.RC**
- The "RuleOB" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Obstruction attributes.

**RULEOO.C, RULEOO.H, RULEOO.RC**
- The "RuleOO" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Other Object attributes.

**RULER.C, RULER.RC**
- The "Ruler" module contains the code and Open Interface resources necessary for the "Ruler" window.

**SMRTHELP.C, SMRTHELP.RC**
- The "SmrtHelp" module contains the code and Open Interface resources necessary for the "Smart Help" window.

**SPLASH.C, SPLASH.RC**

- The "Splash" module contains the code and Open Interface resources necessary for the startup screen.

**SPLASH2.C, SPLASH2.H, SPLASH2.RC**

- The "Splash2" module contains the code and Open Interface resources necessary for the "More on Locate" window.

**START.C, START.RC**

- The "Start" module contains the code and Open Interface resources necessary for the usability "Start" window.

**STARTUP.C, STARTUP.RC**

- The "Startup" module contains the code and Open Interface resources necessary for the help reminder at startup.

**STARTLOG.C, STARTLOG.RC**

- The "StartLog" module contains the code and Open Interface resources necessary for the startup window that allows the user to enter a user name for the help system.

**SYSMODL.C, SYSMODL.RC**

- The "SysModl" module contains the code and Open Interface resources necessary for the "System Model" window.

**TASKMODL.C, TASKMODL.RC**

- The "TaskModl" module contains the code and Open Interface resources necessary for the "Task Model" window.

**USERMODL.C, USERMODL.RC**

- The "UserModl" module contains the code and Open Interface resources necessary for the "User Model" window.

**WEBBROWS.C, WEBBROWS.H, WEBBROWS.RC**

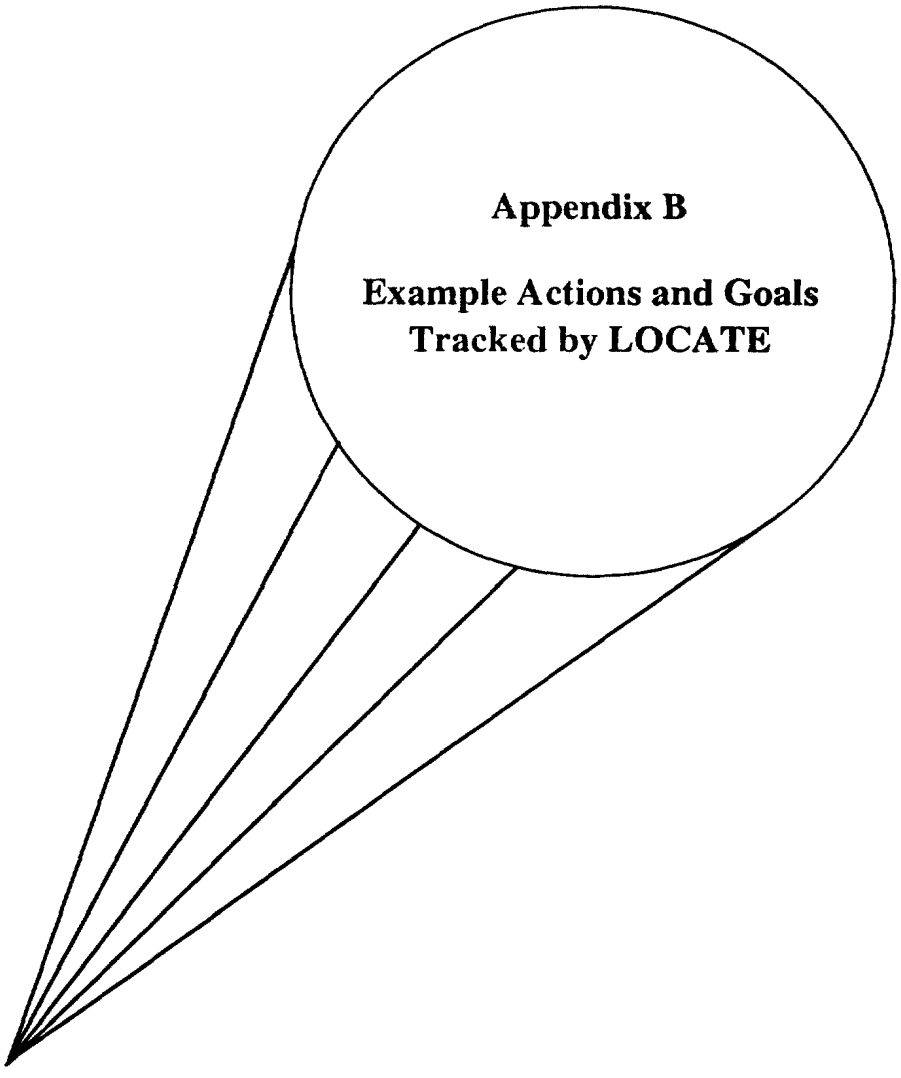
- The "WebBrows" module contains the code and Open Interface resources necessary for the "Web Browser" window.

**WOBS.C, WOBS.RC**

- The "WObs" module contains the code and Open Interface resources necessary for the "Elemental Obstruction" window.

**WSATTR.C, WSATTR.RC**

- The "WSAttr" module contains the code and Open Interface resources necessary for the "Workspace Attributes" window.



# Appendix B

## Example Actions and Goals Tracked by LOCATE

Artificial Intelligence
Management and Development Corporation

Below is an example list of actions and goals that LOCATE is capable of tracking and inferring. They include illustrations of changes to edit text boxes in LOCATE's Workspace Attributes Window. All LOCATE windows now support the inferences of goals when user actions change text in the window.

A second example set illustrates actions and goals inferred when a user repositions, resizes or rotates an object in the workspace.

With the completion of the current work, LOCATE now tracks all user actions in the interface and infers associated goals. The only exceptions are actions taken when invoking and examining the contents in the Task, User and System (Self) Model Windows. These are "meta-actions" relative to the task of workspace design. Such actions could be tracked and their goals inferred if it were determined that such tracking had practical purpose.

In an earlier report entitled, "Intelligent Help in the Workspace Layout Tool," a more extensive list of actions and inferred goals is presented. User goals are indicated below in red and system support goals in blue.

- Workspace Attributes window opened
  - U • To change workspace attributes
    - S • To record changes in workspace attributes
  - U • To examine the workspace attributes [si]
    - S • To display workspace attributes [si]
  - U • To open the Workspace Attributes window [si]
    - S • To display the Workspace Attributes window [si]

- Workspace:Text changed in the x-dimension text box
  - U • To change the x-dimension of the workspace (20, 40) [si]
    - S • To record the change to the x-dimension of the workspace (20, 40) [si]

- Workspace:Upper-left zero point radio button clicked
  - U • To change the zero point to the upper-left corner of the workspace [si]
    - S • To record the change to the zero point to the upper-left corner of the workspace [si]

- Workspace:Item selected in the ruler scale pop-up menu
  - U • To change the ruler scale of the workspace (inch, cm) [si]
    - S • To record the change to the ruler scale of the workspace (inch, cm) [si]

- Workspace:Domain Weights tab selected
  - U • To change the domain weights of the workspace
    - S • To record changes in the domain weights of the workspace
  - U • To examine the domain weights of the workspace [si]
    - S • To display the domain weights of the workspace [si]

- **Workspace:Auditory link weight checkbox clicked**

U • To turn the auditory link weight on [si]

S • To record the change to the auditory link weight [si]

- **Workspace:OK**

U • To accept any attribute changes to the workspace

S • To display any attribute changes to the workspace

S • To undisplay the workspace attributes window

S • To record any accepted attribute changes to the workspace

U • To change workspace attributes [s]

S • To record changes in workspace attributes [s]

U • To change the domain weights of the workspace [s]

S • To record changes in the domain weights of the workspace [s]

- **Workspace:Cancel**

U • To reject any attribute changes to the workspace

S • To undisplay the workspace attributes window

S • To cancel any attribute changes to the workspace

U • To change workspace attributes [-]

S • To record changes in workspace attributes [-]

U • To change the domain weights of the workspace [-]

S • To record changes in the domain weights of the workspace [-]

Example Set II: Actions and goals relative to objects in LOCATE's design window.

- **Object clicked-and-dragged**

U • To move a desk ((10.41,-12.53), (13.34,-13.90)) [si]

S • To display a desk in a new position ((10.41,-12.53), (13.34,-13.90)) [si]

U • To change the coordinates of a desk ((10.41,-12.53), (13.34,-13.90)) [si]

S • To update the X and/or Y coordinates of a desk ((10.41,-12.53), (13.34,-13.90)) [si]

- **Object handle clicked-and-dragged**

U • To resize a desk ((0.67,0.39), (1.02,1.02)) [si]

S • To display a desk with a new size ((0.67,0.39), (1.02,1.02)) [si]

U • To change the dimensions of a desk ((0.67,0.39), (1.02,1.02)) [si]

S • To update the dimensions of a desk ((0.67,0.39), (1.02,1.02)) [si]

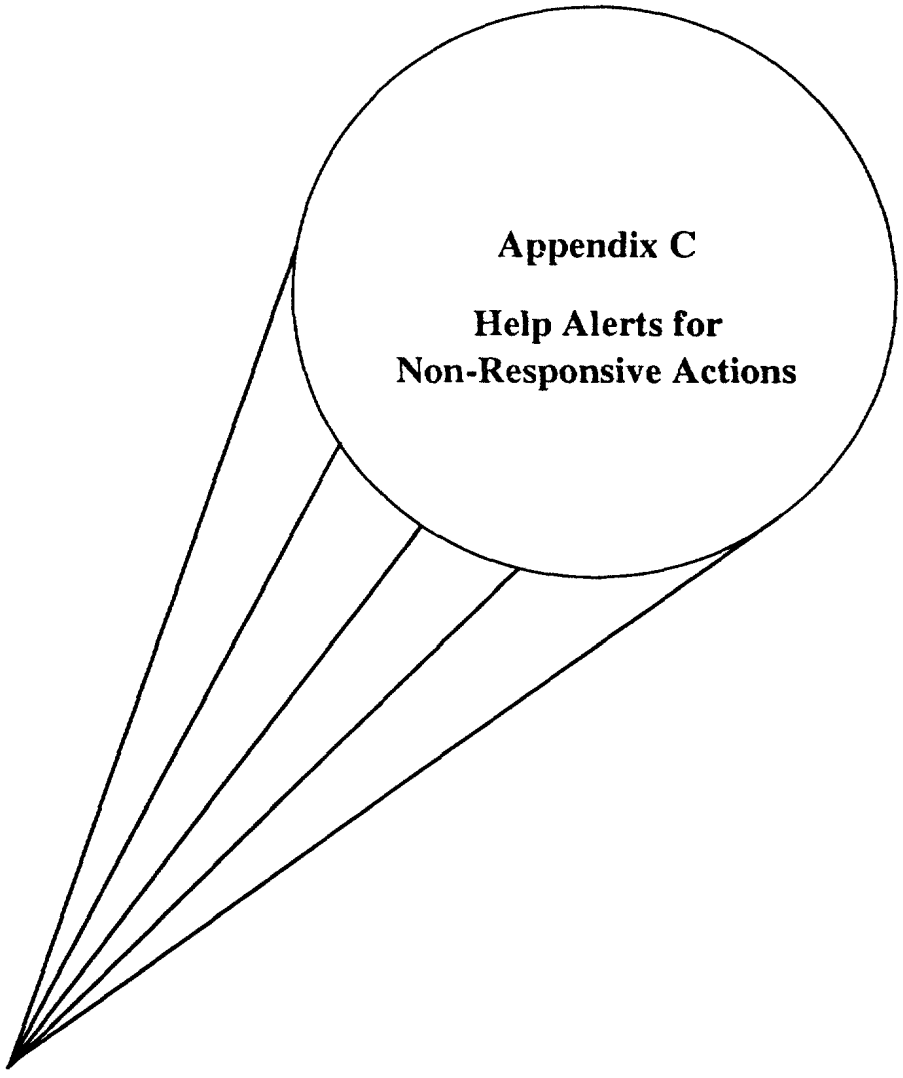
- **Object clicked-and-dragged when in rotate mode**

U • To rotate a desk (0.00, 75.00)

S • To display a desk in a newly rotated position (0.00, 75.00)

U • To change the angle of a desk (0.00, 75.00)

S • To update the angle of a desk (0.00, 75.00)



# Appendix C

## Help Alerts for Non-Responsive Actions

Artificial Intelligence
Management and Development Corporation

This Appendix contains the wording for all the Help Alert Windows that LOCATE displays for users who perform a Non-Responsive Action, that is, an action that produces no system response other than a simple audio signal.

Help Alerts are separated by a double row of dashed lines. Wording for the buttons appear at the bottom of each Alert. The Non-Responsive action appears in boldface in the lead-in comments to each Help Alert.

You have **double-clicked on a link**, an action not associated with any LOCATE function. your goal is:

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To access Link Function Data | 1. Select a workstation.<br>2. Select "Link Functions" from the Data Menu. |
| B. To access Priority Weight Data | 1. Select any pair of workstations.<br>2. Select "Priority Weights" from the Data Menu. |
| C. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

Don't Show Again Read Not Read

==========

==========

You have **double-clicked on a rotation arrow within a workstation**, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To adjust the workstation's angle of rotation | 1. Double-click on the workstation to bring up its attributes window.<br>2. Change its angle of rotation.<br>[Note: The rotation angle for the workstation's Source/Receiver (S/R) node may also be changed in this window.] |
| B. To open the workstation's attributes window. | 1. Double-click anywhere in the workstation except on the rotation arrow. |

| | |
|---|---|
| C. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

Don't Show Again Read Not Read

==========

==========

You have **double-clicked on a rotation arrow within an elemental obstruction**, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To adjust the obstruction's angle of rotation | 1. Double-click on the obstruction to bring up its attributes window.<br>2. Change its angle of rotation. |
| B. To open the obstruction's attributes window. | 1. Double-click anywhere in the obstruction except on the rotation arrow. |
| C. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

Don't Show Again Read Not Read

==========

==========

You have **double-clicked on a rotation arrow within a fixed obstruction**, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To adjust the fixed obstruction's angle of rotation | 1. Double-click on the fixed obstruction to bring up its attributes window.<br>2. Change its angle of rotation. |
| B. To open the fixed obstruction's attributes window. | 1. Double-click anywhere in the fixed obstruction except on the rotation arrow. |

C. Some Other Goal

How Do I:
[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]
[Edit Text Window].

Don't Show Again  Read  Not Read

==========

==========

You have **clicked on an inactive "Save CF" button**, an action that produces no LOCATE response as long as the button is dimmed.

| CONDITION OF BUTTON | COMMENTS |
|---|---|
| ACTIVE (highlighted) | Cost Functions are computed automatically. They are saved by clicking on the active button. |
| INACTIVE (dimmed) | Cost Functions are computed manually. Cost functions are automatically saved. |

- MORE <button>

Don't Show Again  Read  Not Read

==========

==========

-- MORE --

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To compute cost functions **manually**. | 1. This is the default condition for LOCATE<br>2. Cost functions may be manually computed in three ways:<br>a. Click on the Cost Function Value Window at the bottom of LOCATE's palette;<br>b. Select "Cost Function" from the Execute Menu;<br>c. Select "Cost Function..." from the Execute Menu. |

3. In Manual mode, cost function values and associated designs are automatically saved to a Cost Function History. These may be permanently saved when they are first created or when they are recalled from the History, during a LOCATE session.

Note: In this condition, the "Save CF" button just above the Cost Function Value Window is dimmed (inactive).

B. To compute cost functions **automatically**.

1. Select "Cost Function..." from the Execute Menu.
2. Click the radio button that reads, "Automatic."

Note: In this condition, any change to objects in the design space signals LOCATE to recompute the cost function. Although cost functions are computed automatically, they and their associated designs are no longer saved to the Cost Function History. This is done by clicking on the now highlighted, "Save CF" button.

C. Some Other Goal

How Do I:
[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]
[Edit Text Window].

Don't Show Again Read Not Read

==========

==========

You have **clicked on an inactive Cost Function Value Window**, an action that produces no response when cost functions are being computed automatically by LOCATE.

| CONDITION OF BUTTON | COMMENTS |
|---|---|
| ACTIVE (highlighted) | Cost Functions are computed manually by default. They are saved automatically. |
| INACTIVE (dimmed) | Cost Functions are computed automatically. They are saved by clicking on the "Save CF" button. |

- MORE <button>

Don't Show Again Read Not Read

You have **double-clicked on the text area above the palette**, an action not associated with any LOCATE function. This area shows the name of the currently-selected tool in the palette.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To display the name of palette tools | 1. Hold down the Control key and move the cursor over the tool whose name you would like to know. |
| B. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

Don't Show Again Read Not Read

==========

==========

You have **clicked on one of the rulers in the design window**, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To change ruler settings | 1. Select "Rulers..." from the Arrange Menu.<br>2. Ruler settings appear in the lower left corner of the Workspace Attributes Window.* |
| B. To turn off the display of rulers | 1. Select "Show/Hide Rulers" from the Arrange Menu. |
| C. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

* Other ways to open the Workspace Attributes Window include selecting "Workspace" from the View Menu or simply double-clicking in an open area of the workspace.

Don't Show Again Read Not Read

==========

==========

You have **double-clicked on a text object**, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To edit the text object | 1. Click on the text tool in the palette.<br>2. Click inside the text object.<br>3. Edit the text. |
| B. To resize the text object | Note: Text objects cannot be resized. |
| C. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

Don't Show Again Read Not Read

==========

==========

You have control-double-clicked on the Source/Receiver (S/R) node of a workstation, an action not associated with any LOCATE function.

| IF YOUR GOAL IS... | THEN... |
|---|---|
| A. To change the attributes of the S/R node | 1. Select the workstation associated with the S/R node.<br>2. Select "Workstation" from the View Menu.*<br>3. Settings for the S/R node appear in the bottom portion of the Workstation Attributes Window.* |
| B. Some Other Goal | How Do I:<br>[Tell LOCATE what you would like to do OR select the "Read" button below to dismiss this window]<br>[Edit Text Window]. |

* Other ways to open the Workstation Attributes Window include simply double-clicking on the Workstation.

Don't Show Again Read Not Read

==========

==========

You have **clicked-held-and-dragged on a handle of a Source/Receiver (S/R) node of a workstation**, an action not associated with any LOCATE function.

The S/R nodes of workstations are NOT resizable as they represent points with no dimensions.

Other objects NOT resizable include:
Text objects
Grouped complex objects, that is, objects including a workstation or an obstruction.

Don't Show Again Read Not Read

==========
==========

You have **clicked-held-and-dragged on a handle of text object**, an action not associated with any LOCATE function.

Text objects are NOT resizable in LOCATE.

Other objects NOT resizable include:
Source/Receiver (S/R) nodes.
Grouped complex objects, that is, objects including a workstation or an obstruction.

Don't Show Again Read Not Read

==========
==========

You have **clicked-held-and-dragged on a handle of a grouped, complex object,** an action not associated with any LOCATE function.

Grouped, complex objects, which include workstations or obstructions, are NOT resizable.

Other objects NOT resizable include:
Source/Receiver (S/R) nodes
Text objects

Don't Show Again Read Not Read

==========
==========

# DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1 ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g , Establishment sponsoring a contractor's report, or tasking agency, are entered in section 12 )
Artificial Intelligence Management and Development Corporation
206 Keewatin Avenue, Toronto, ON M4P 1Z8
CANADA

2. DOCUMENT SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable)
UNCLASSIFIED

3 DOCUMENT TITLE (the complete document title as indicated on the title page. Its classification should be indicated be the appropriate abbreviation (S,C,R or U) in parentheses after the title.)
Testbed for intelligent aiding using the LOCATE workspace layout tool — Final Report (U)

4. DESCRIPTIVE NOTES (the category of the document, e.g., technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)
Final Report

5. AUTHOR(S) (Last name, first name, middle initial. If military, show rank, e.g. Burns, Maj. Frank E.)
Edwards, Jack L.

| 6. DOCUMENT DATE (month and year of publication of document) | 7 a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.) | 7.b. NO. OF REFS. (total cited in document) |
|---|---|---|
| November 1999 | 59 | 12 |

| 8.a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) | 8.b. CONTRACT NO. (if appropriate, the applicable number under which the document was written) |
|---|---|
| Project 3ad13 | PW&GS W7711-8-7546/001/TOR |

| 9 a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.) | 9 b OTHER DOCUMENT NO.(S) (any other numbers which may be assigned this document either by the originator or by the sponsor.) |
|---|---|
| AC201 | DCIEM No. CR 2000-029 |

10. DOCUMENT AVAILABILITY (any limitation on further dissemination of the document, other than those imposed by security classification)

| | |
|---|---|
| X | Unlimited distribution |
| | Distribution limited to defence departments and defence contractors; further distribution only as approved |
| | Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved |
| | Distribution limited to government departments and agencies; further distribution only as approved |
| | Distribution limited to defence departments; further distribution only as approved |
| | Other |

11. ANNOUNCEMENT AVAILABILITY (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (10.) However, where further distribution (beyond the audience specified in 10) is possible, a wider announcement audience may be selected.)

12. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development Include the address.)
Defence and Civil Institute of Environmental Medicine
P.O Box 2000, 1133 Sheppard Ave West, Toronto, ON M3M 3B9 CANADA

13 ABSTRACT ( a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U) It is not necessary to include here abstracts in both official languages unless the text is bilingual)

The purpose of the work reported here was to develop generalizations from principles and techniques of intelligent aiding that will be useful to projects developed at Canada's Defence and Civil Institute of Environmental Medicine (DCIEM).

In realizing that purpose, the principles and techniques of intelligent aiding implemented in DCIEM's LOCATE Workspace Layout Tool were combined with other, similar ones from related areas of research. LOCATE is a CAD tool that allows users to create designs and analyze their communication efficiency.

The LOCATE tool is serving as the focus for discussion, design and implementation in an emerging testbed for elaborating ideas on intelligent aiding. Earlier work on building an infrastructure to support such aiding in LOCATE was extended so that the tracking of all low-level user actions is now complete, as is the ability of LOCATE to infer the goals those actions imply.

Further extensions during this work phase included the identification and elaboration of categories of actions that imply a need for help. A category of "Non-Responsive Actions," or actions for which there is no system response, was selected for implementation. Help was designed and implemented first of all to inform users that such actions produce no response and second, to help them identify and achieve whatever goal(s) they had in mind when they performed those actions.

Task and help, actions and goals are represented in explicit models of task, user and system in LOCATE, and are displayed in associated windows in its interface In the Task Model, an action and goal history is maintained as the user creates and analyzes designs. As that work proceeds, help of various kinds is initiated by LOCATE and, in a similar way, its System Model maintains an action and goal help history.

Several projects were identified as likely candidates for the application of the principles and techniques identified here, but none was at a stage where that could be done in the short term. Those and other projects will continue to be monitored for opportunities to incorporate intelligent aiding capabilities.

14 KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible, keywords should be selected from a published thesaurus, e g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

HUMAN ENGINEERING TOOLS

HUMAN MODELLING

WORKSPACE LAYOUT

WORKSPACE DESIGN

FACILITY LAYOUT

COMPUTER-AIDED DESIGN

INTELLIGENT HELP

ADAPTIVE INTERFACES

LOCATE

EXPLICIT USER MODELS

#513544