Providing Intelligent Help Across Applications in Dynamic User and Environment Contexts

Ashwin Ramachandran
Liquid Narrative Group
Department of Computer Science
Box 7535, NC State University
Raleigh, NC 27695
011.919.513.3038
Ashwin.Ramachandran@Blackbaud.com

R. Michael Young
Liquid Narrative Group
Department of Computer Science
Box 7535, NC State University
Raleigh, NC 27695
011.919.513.3038
young@csc.ncsu.edu

ABSTRACT
The problem of providing help for complex application interfaces has been a source of interest for a number of researcher efforts. As the computational power of computers increases, typical applications not only increase in functionality but also in the degree of interaction with the computational environment in which they reside. This paper describes an ongoing project to design an Intelligent Help System (IHS) that provides context-sensitivity not only through its modeling of application states but also its modeling of the interaction between applications and between an application and the environment in which it resides.

Categories and Subject Descriptors
H.5.2 **[User Interfaces]**: Graphical User Interfaces (GUIs)

I.2.m **[Computing Methodologies]**: Artificial Intelligence -- Miscellaneous

General Terms
Algorithms, Human Factors

Keywords
Intelligent Help Systems, Planning, Intelligent Interactive Environments

1. INTRODUCTION
There are powerful software tools available today used for both specialized and non-specialized tasks (e.g., 3D modeling, word processing, music library management). These applications are often used by novice users who attempt tasks without significant training or knowledge of the application's interface; this is known as the production paradox [1]. These kinds of applications are diverse and complicated in the variety of functionality they provide, often interacting with other applications on the user's system. With current platforms (Windows, Macintosh, Linux, etc) providing extensive multi-tasking facilities, interaction with these applications is sometimes affected by the context of the environment itself (e.g., application windows being minimized, maximized or obscured by those of other applications). The interdependencies between applications and their environments increase the difficulty of providing effective context-sensitive help when building an application's help documentation.

We have developed a tool called SmartAidè, an adaptive passive help system that gives step-by-step textual instructions coupled with system-driven execution of the actions that make up the task being described. The tool works on the premise that the user has a goal in mind when requesting help. The action sequences provided to a user's request for help are automatically generated by an AI planning system; the plans it produces are designed to execute within the user's workspace, changing the user's current system state (task and application context) to the task state that the user desires.

A planning-based approach to the generation of context-sensitive task help has several advantages over other techniques for automatic help generation. First, planning approaches create action sequences that can successfully execute from a wide range of initial states without the need for system designers to anticipate each of these states and enumerate the appropriate action for each. Consequently, plan-based systems do not require the designers to construct complex task models to determine user goals by tracking their interactions. Second, the use of planning enables the designer to bring together two action representations - the procedural representations used for action execution within the application environment and the declarative representation used by the planners. This kind of representation allows for the system both to support existing applications and to integrate with applications during interface design where the action definitions form the procedural action representation. Third, because planners search through a space of possible plans while constructing their plans, heuristics can be used to guide this search so that the nature of the final plan can be tailored to individual users' preferences. Finally, planners use techniques to

IUI'05, January 10–13, 2005, San Diego, California, USA.

ACM 1-58113-894-6/05/0001.

create their plans that add explicit models of the causal and temporal relationships between their plans' actions. Analysis of these structures facilitates re-planning in the face of action failure (e.g., when users interfere with the system's demonstration of the help being described) as well as the generation of effective explanations for the complex tasks being described [4].

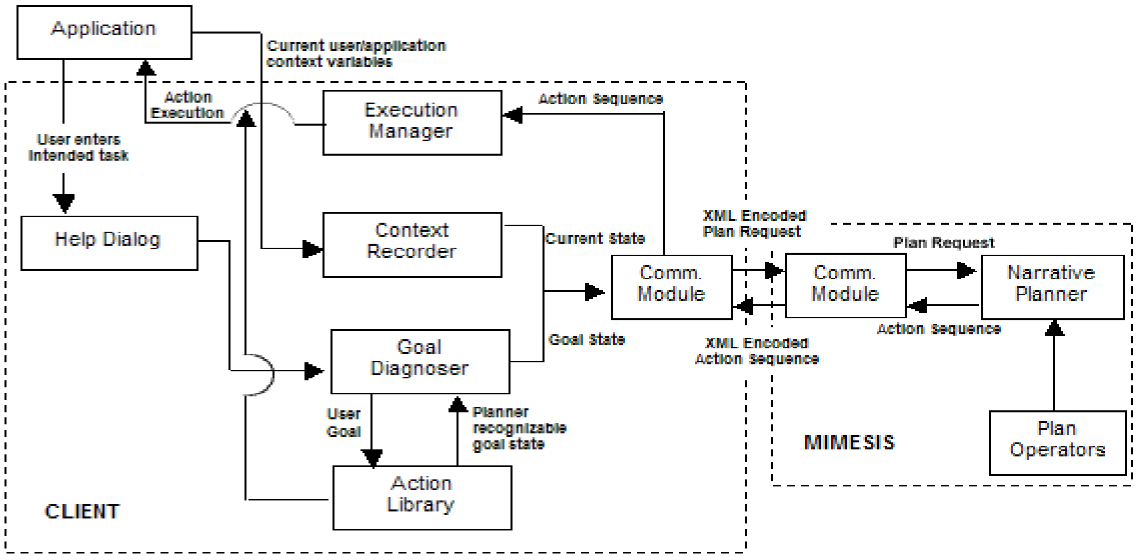

Figure 1. SmartAidè architecture

2. SMARTAIDÈ OVERVIEW

The SmartAidè system is built on top of Mimesis; an intelligent control architecture designed to structure and manage user interaction within virtual environments [6]. Mimesis uses a client/server architecture where high-level reasoning about plan structure and user interaction is performed by the intelligent control elements on the server while the client performs the low-level control of a virtual environment (in our case, the user's desktop environment).

Activity within SmartAidè begins when an application is first run. As each application starts, a SmartAidè Help Dialog for that application is created and minimized. The SmartAidè Context Recorder then records the state of all the objects within the application and the state of the application with respect to its environment (minimized, maximized, obscured, resized etc). As the user performs actions within his/her environment, SmartAidè matches these actions to declarative action definitions stored in an Action Library. Each entry in this library holds a STRIPS-like definition of pre- and post-conditions for an action [2] along with a pointer to text templates that can be used to provide help specific to the associated action.

To request help, the user brings up the SmartAidè Help Dialog and indicates the need for help with a specific task (in the current prototype implementation, a pre-defined set of general tasks for an application are listed in the Help Dialog as menu items). The help request is translated by the Goal Diagnoser into a planning problem specification, an expression describing the user's current workspace context, the desired state of his/her application and the library of actions available to the system to construct an action sequence achieving his/her goals. This planning problem is translated into XML and sent via a socket connection to the planning component within Mimesis.

The Mimesis planner uses DPOCL [5] as the planning algorithm for generating the plan structure. DPOCL uses *refinement search* [3] as a model for its plan reasoning process. The initial planning problem for DPOCL is created using the specifications of the current and goal states taken from the plan request from the client. Initially, the root node of the plan space graph is the empty plan containing just the initial state description and the list of goals that together specify the planning problem. Nodes in the interior of the graph correspond to partial plans and leaf nodes in the graph are identified with complete plans (solutions to the planning problem) or plans that cannot be further refined due for instance, to inconsistencies within the plans that the algorithm cannot resolve.

The complete plans contain a set of *steps* representing the plan's actions. These steps have *ordering constraints* that define the order in which the steps need to be executed, and *causal links* that connect two steps just when the effect of one step establishes the precondition for the other. Hence a complete plan is a minimal set of actions that ensures the transformation of the application environment from its current state to the user's desired goal state. Because the initial plan request contains an encoding of all relevant aspects of the current environment state and the plan operators contain a description of the relevant actions within the environment, the plan will be tailored to both the environment context and the application context.

Once a complete plan has been found, the steps in the plan are encoded as an XML message and sent to the Execution Manager. The Execution Manager builds a directed acyclic graph (DAG) in which the nodes represent primitive actions and arcs between two nodes indicate temporal dependencies between the two actions' execution. To execute the plan, the Execution Manager removes the minimal elements of the DAG (those actions that have no temporal dependencies upon other actions) and sends an XML description of each action and its parameters to the SmartAidè client on the user's machine. Using a pre-defined translation scheme, the client translates the action description into a function call with pointers to the appropriate system objects that make up the action's arguments.

This function is responsible for the correct implementation of the action characterized by the declarative representation used by the Mimesis planner. As each of the client-side actions execute, they first check the relevant system state to verify that their preconditions hold. Next, they perform their main task, changing the world according to their intended semantics. Finally, they re-check their post-conditions, validating that their execution has correctly changed the system state. If an error is encountered in this process (e.g., the user has altered the system state so that one of the action's pre-conditions no longer holds at the time the action is executed), an error message is sent by the function back to the Execution Manager. Should the function execute correctly, a termination message is sent back to the Execution Manager, indicating that the temporal dependencies in the execution DAG can be updated and new actions initiated for execution.

The functions that execute on the user's machine are responsible for the state changes that demonstrate the task needed to achieve the user's goals. These functions provide textual instructions (currently generated by instantiating templates for each action) displayed in a help window. As this text is displayed, the function also drives the actual execution of the action within the environment (e.g., as instructions describing the process of minimizing a window are displayed in the help texts window, the system moves the cursor to the window's minimize box and left-clicks). Hence the help is presented in such a way that it shows the user how to get his/her task done without trying to teach the user the underlying causal explanations for the actions.

3. CONCLUSIONS AND FUTURE WORK

We have presented an approach of providing context-sensitive help in situations where both the user and environment context affect the help provided to the user. SmartAidè has been designed currently to provide help for the iTunes and Finder applications on the Macintosh environment, though the plan-based knowledge representation can support a wide range of application and action types. To measure the effectiveness of the help provided by SmartAidè, we are devising an empirical evaluation where the performance of a set of users interacting with various help systems will be observed. The users, derived from a set of students comparable in their knowledge of the applications and the environment, will be divided into two groups, one having access to SmartAidè and the other having access only to the help provided by the applications they are using. Both sets of users will then be given a set of tasks to be performed using iTunes and Finder. During their interactions several parameters will be automatically recorded e.g. time taken to perform each task, the number of times help is evoked for each task and number of tasks performed. The evaluations will give us an indication of the effectiveness and usability of the system.

SmartAidè currently focuses on helping novice users. More conclusive studies with experienced users will help us determine the kind of help experienced users seek. In future work, we will encode knowledge of the information needed by users with intermediate skill into a user model. We will also incorporate a robust method of allowing users to interact more naturally when seeking help, rather than selecting from a menu of options, as is currently the case.

4. ACKNOWLEDGEMENTS

We would like to thank Craig Allen, who has worked extensively on implementing the SmartAidè system. This work was supported by NSF CAREER Award #0092586.

5. REFERENCES

[1] J. M. Carroll and M. B. Rosson, "Paradox of the active user," in J .M. Carroll, editor, *Interfacing thought: cognitive aspects of human-computer interaction*, pp. 80–111. MIT, Cambridge Mass., 1987.

[2] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," in James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, Morgan Kaufmann, 1990.

[3] S. Kambhampati, C. Knoblock and Q. Yang, "Planning as Refinement Search: A Unified framework for evaluating design tradeoffs in partial order planning," in *Artificial Intelligence special issue on Planning and Scheduling*, 1995.

[4] R. M. Young, "Using Grice's Maxim Of Quantity To Select The Content Of Plan Descriptions," in *Artificial Intelligence*, no. 115, 215-256, 1999.

[5] R. M. Young and J. D. Moore, "DPOCL: A Principled Approach To Discourse Planning," in *Proceedings of the Seventh International Workshop on Text Generation*, Kennebunkport, ME, 1994.

[6] R. M. Young, M. O. Riedl, M. Branly, A. Jhala, R. J. Martin and C. J. Saretto, "An architecture for integrating plan-based behavior generation with interactive game environments," in *Journal of Game Development*, vol. 1, issue 1, pages 51-70, 2004.