Automatic Generation of Help from Interface Design Models

Roberto Moriyon
Instituto de Ingenieria del Conocimiento
Universidad Autonoma de Madrid, mod. C-XVI
28049, Madrid, Spain
34-1-397-3973
roberto@gw.iic.uam.es

Pedro Szekely
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292
(1-310) 822-1511
szekely@isi.edu

Robert Neches
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292
(1-310) 822-1511
neches@isi.edu

ABSTRACT
Model-based interface design can save substantial effort in building help systems for interactive applications by generating help automatically from the model used to implement the interface, and by providing a framework for developers to easily refine the automatically-generated help texts. This paper describes a system that generates hypertext-based help about data presented in application displays, commands to manipulate data, and interaction techniques to invoke commands. The refinement component provides several levels of customization, including programming-by-example techniques to let developers edit directly help windows that the system produces, and the possibility to refine help generation rules.

KEYWORDS: Automatic Help Generation, Model-Based Interface Design, Hypertext-Based Help, Help Customization, Help Generation Rules.

INTRODUCTION
Help systems today are usually developed as separate artifacts from the systems they support. As a result, building and maintaining help systems requires substantial effort:

- The design of the help system replicates reasoning that went into the design of the application.
- Complex programming is needed because the help system must access internal application data structures.
- There is no support for changing the help system when the application is modified.
- Making help systems have a consistent interface across different applications is difficult.

This paper describes a system for constructing help systems called H3 (HUMANOID Hyper Help). H3 makes two main contributions. First, it delivers to end users more information than most help systems (e.g., Macintosh balloon Help and the MS Windows help system). Second, H3 lowers the cost of constructing help systems because it generates a default help system automatically, and it provides an easy way to customize the automatically-generated help messages to improve their quality.

H3 can produce basically four kinds of help messages. The first kind is a summary message describing the item that the user selects for help. The other three kinds provide answers to the questions "What commands are available?", "What is displayed here?", and "Where can I click, and what will happen?". H3 does not currently provide task-oriented help to answer questions like "How do I delete a file?".

The help information that H3 gives users goes beyond canned texts attached to the static portions of the display, like typical Macintosh balloon help [4]. H3 help messages consist of concatenations of pieces of canned texts with embedded links that show where information mentioned in the help text is displayed, and can also be used to access related information (like in hypertext systems). In addition, the messages are context sensitive, so that asking for help on an item produces different messages depending on context and application state (e.g. asking for help on a dimmed item produces a message explaining why the item is dimmed, where as asking for help when the item is enabled, will tell the user what object will be affected by the corresponding command).

H3 constructs a default help system for an application automatically by using the specifications used to construct the application's user interface. Developers can refine the automatically generated help system at different levels. The simplest level is to replace the default pieces of canned text by more appropriate ones. Developers do this by editing in-place the help windows that H3 produces. Developers can also add links to messages, and add messages to display elements that do not produce messages by default. Advanced developers can change the behavior of the help

system itself by defining new rules in the H3 rule system that computes the help messages.

H3 tries to leverage the skills of humans and computers. Humans are much more skilled than computers at writing prose, and since the quality of the text is critical to the success of the help system, H3 allows humans to write the text. However, other aspects of help, such as showing the mouse sensitive regions of a display, or listing the commands applicable to a selected object must be computed based on the state of the application. H3 performs these tasks automatically freeing the help developer from having to program these aspects of the help system. However, developers still have the option to customize the help messages associated with the computed aspects of the help system.

H3 assists help system developers in other ways too. Developers need not start with a blank slate: they do not need to determine the elements of the display that should have help messages, and they do not need to write the help messages from scratch either. It is often easier to refine existing material than to create it form scratch.

The rest of the paper is organized as follows: next we discuss related work to highlight the differences between H3 and other tools for constructing help systems. We then give some rationale for the design of the help systems that H3 generates, and show some examples of help windows for a file manager application. In the following sections we describe the implementation of H3 and then the facilities for customizing the help systems. We close with conclusions and directions for future work.

RELATED WORK

H3 is similar in spirit to balloon help on Macintoshes [4]. Both systems are geared towards providing help about features of an application that are visible on the application displays, and neither system supports task-oriented help to assist users with high level tasks (e.g., how do I insert a figure in my document). The are also many differences. H3 help is available for parts of the screen that can be nested at different levels and for groups of parts, while balloon help only displays information about atomic parts of the screen. Also, although in theory balloon help is available for both static and dynamic portions of the display, in practice this feature is difficult to use and almost no application uses this feature. Hence, balloon help almost never shows the mouse sensitive regions of a display, explain why a button is dimmed, and other such topics that depend on the dynamic behavior of an application.

The interface to the help system is also different in balloon help and in H3. The balloon help is more convenient to use because it simply pops up the help balloons while the user is interacting with the interface. H3 uses a separate help window to show the messages, and hence requires users to explicitly interact with the help system. This added complexity in H3 is necessary because it provides much more information to users, more than fits in a small pop up window.

H3 is also similar to the help in Microsoft Windows [5], or OS/2 [7] in that it uses a separate help window to show help, and in the use of hypertext links to navigate to related topics. The main difference is that these systems are oriented towards the generation of help for tasks, commands and key bindings, practically without any interaction with the interface, and only hypertext references to other messages are allowed in these systems. However, the links in our help messages not only point to other messages, but can refer to different parts of the window, commands, data displayed, etc., and in that case they are highlighted in the same color as the corresponding portion of the display. The other difference is that the MS Windows and OS/2 help systems provide little help about the dynamic aspects of an application. H3 currently does not provide good support for navigating the help system. H3 does not have a history mechanism or features that let users navigate the help information in an organized way.

Cartoonist [8] is another example of a system that generates help automatically from the models used to construct an interface. Cartoonist automatically constructs animations that show how to invoke the commands of an application. H3 provides hypertext messages rather than animations, and covers a complementary domain. It provides help on displayed data and required inputs as well as manipulations. Also, unlike other help generation systems, both Cartoonist and H3 generate help that depends on the context of an application and its interface.

Techniques based on AI technology have been proposed to produce help and documentation systems, [1, 6]. The AI systems have more sophisticated natural language generation facilities, and also include planners to construct help texts with better discourse structures than those in H3. However, the AI systems are typically difficult to integrate into applications because they require much more detailed models of an application, and the effort to construct these models is usually very large.

OVERVIEW OF H3

This section presents an overview of the H3 help system. The section first discusses general issues that influenced the design of H3 and then shows examples of the help screens that H3 can generate.

We have identified three issues that influence the design of help windows:

- *Topic* refers to the object explained in a help message. H3 currently supports the following topics: *data*, explains the meaning of data represented by the graphic and text elements that appear in a region of the display; *commands*, explains what the various commands of the application can do; *interaction*, explains the effect of input events (e.g., invoke the print command, select an

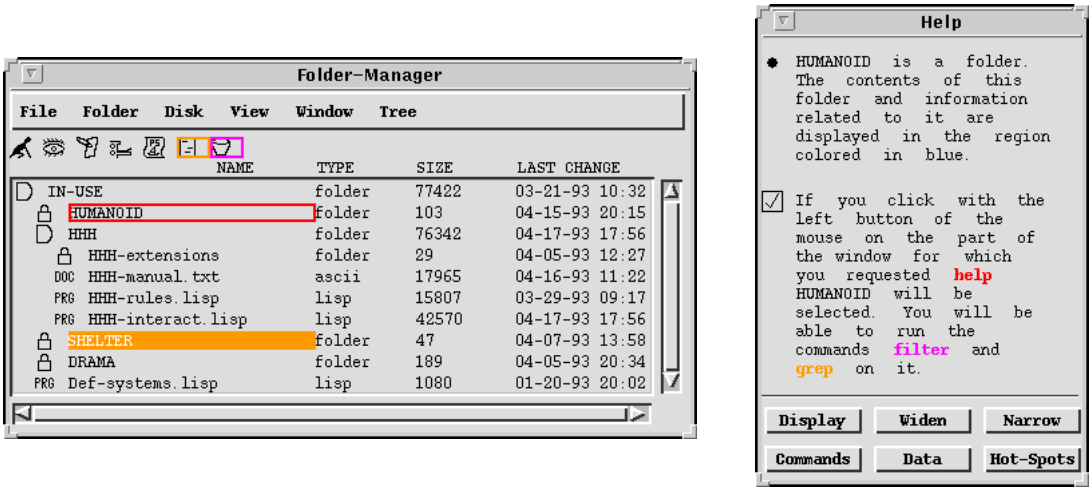

Figure 1. A folder manager application together with the help window showing messages for the folder named HUMANOID. HUMANOID is highlighted in red to indicate that it is the region selected for help. The last two icons below the menu bar are also highlighted because they are the references of the links called filter and grep that appear in the help text.

object). Other information classes are possible, but not currently supported by the H3 help generation algorithms. One example is *task*, which would explain how to perform user tasks using the commands of an application.

- *Scope* or *extent* refers to the size of the region of the screen to be explained. The displays of an application are organized hierarchically, and the messages to explain the elements of the display are different depending on the level of the hierarchy to be explained. For example, consider the window for the folder manager application shown in Figure 1. The region of interest could be the whole window, or the area showing the information about all the files, or for one entry that shows the name of the file, its type, size and modification date, or the label showing the name of a file.
- *Links to application displays*. The information in the help screens typically refers to information displayed in the application windows. For example, the second message in Figure 1. refers to the filter and grep commands. These commands are displayed as icons in the folder manager window (the last two icons below the menu bar, the ones with boxes around them).

We designed H3 to take into account these issues. When a user asks for help by pointing with the mouse to an object on an application display (e.g., the word HUMANOID in Figure 1.) and pressing the HELP key on the keyboard, H3 pops up a window with the data messages about the smallest graphical or textual element under the mouse cursor (the word HUMANOID). The help window has buttons that allow users to display the messages about other topics, and buttons to widen or narrow the scope or extent of the explained region (e.g., widen to the whole line showing the name, type and change data of the HUMANOID file). The links to application displays are shown using a color coding scheme: a different color is assigned to each link, and both the text in the help window and the corresponding region in the application display are highlighted using that color (e.g., the word grep is shown in light gray in the help window, and the corresponding icon is highlighted with a light gray rectangle in the application display[1]). In addition, selecting the link in either the text or the application display moves to the corresponding help screen. To avoid cluttering the application display with link highlightings, H3 only highlights the links corresponding to a single message in the help window. Users can select the message for which they want the links highlighted.

Example

This section briefly illustrates the capabilities of help systems generated using H3 by showing examples of several help windows for a file manager. Interestingly, our file manager application is similar to the Macintosh Finder and the MS Windows or OS/2 directory managers, but the help available through H3 is much more comprehensive. In addition, it is mostly automatically generated from the model used to construct the interface itself.

Figure 1. shows the help window produced by hitting the HELP key while the mouse is over the word HUMANOID. The word HUMANOID is highlighted with a black rectangle to indicate that it is the portion of the display being explained in the help window. The help window has two messages (in this case, the first message was customized by

[1]H3 highlights regions in the application using different colors rather than different shades of gray.

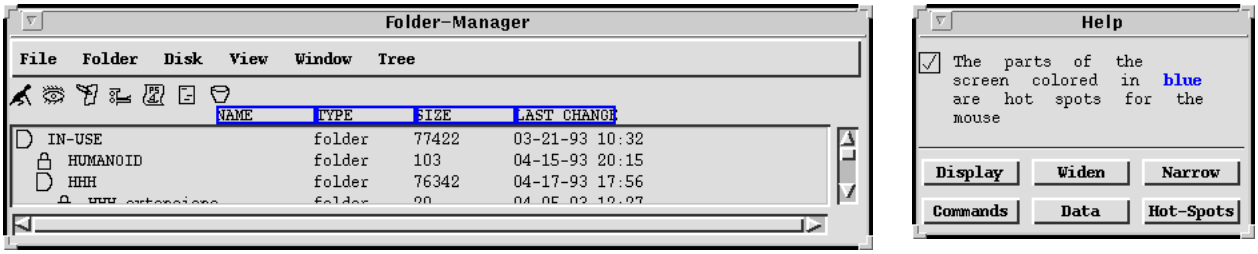

Figure 2. Help window showing the mouse sensitive regions (hot spots) in the headers pane of the folder manager. Each label in the header is highlighted in blue, as mentioned in the help message.

the developer, and the second one was generated by default). The first message is a data message that explains the meaning of the selected portion of the display. The second one is a command message that tells the user what operations are available on the selected element.

Each message in the help window has a bullet on its left. When the user selects the bullet, H3 highlights the portions of the display referred to by the links in the selected message. For example, in Figure 1. the second message is selected. The words filter and grep are displayed in different colors (shades of gray in the figure), indicating that they are links, and in the folder manager window the last two icons below the menu-bar are highlighted with the same colors to indicate that they display the objects referred to by the links. Selecting one of the links (e.g., grep) changes the help window to show information about link referent (the grep command). Should the user select the first message, all the elements to the right of the word `HUMANOID` would be highlighted in blue.

The commands at the bottom of the help window allow the user to access other help information. The `Display` command allows the user to display the initial help screen after navigating to see the information about other topics. The other two buttons in the first row let the user widen and narrow the area of the screen for which help is requested (e.g., widen to the whole line showing the complete information for a file). The last row of buttons let the user ask for help about different topics. `Commands` produces a summary of all the application commands that operate on the data displayed in the portion of the display being explained, `Data` describes the information displayed in the region being explained, and `Hot-Spots` highlights the mouse-sensitive elements of the application display and describes the input techniques that can be used to operate on them.

Figure 2. shows a help window about a different region of the screen and about a different topic. In this case the user asked to see the hot spots associated with the headings of the folder manager window (in this case, the message is was automatically constructed). Each word in the heading is highlighted with a rectangle because it is a hot spot. If the user selects the Commands button, the help window will display help about the commands that can be invoked by selecting the headings. The help window will have a message for each heading. For example, the customized message for the `SIZE` heading says "If you click the left button on SIZE, the contents of the window will be sorted according to their size"

IMPLEMENTATION OF HELP GENERATION

In H3 the help messages are specified as rules of the following form

```
When Conditions then Message-Descriptions
```

The conditions identify the context when a particular message is appropriate, and the message descriptions are templates that specify the text and links of message to be generated.

```
When ?object is selected for help
  and ?object has an interaction technique
      ?inter that invokes command ?cmd
  and command ?cmd has an input ?inp
  and interaction technique ?inter sets
      input ?inp to value ?val
then display the interaction message
  "If you "
  generate-activation-message (?inter)
  " the command "
  ?cmd
  "will be run using"
  ?val
  "as its input"
```

Figure 3. An example of a help rule (in pseudo-English)

Figure 3. shows the rule that generates messages of the following form (the rule is shown in pseudo-English to make it more understandable)

> If you click with the left button on the part of the window for which you requested help, the command Print will be run using H3-Paper.ps as its input

The conditions test whether the display element to be explained (`?object`) appears in a context where the message specified by the right hand side of the rule is appropriate. The words with a question mark preceding them (`?object`, `?inter`, `?cmd` and `?val`) are variables. When H3 evaluates the conditions of the rule, the variables

are bound to the objects in the model that satisfy the conditions. For instance, in the case of the help message shown above, *?cmd* would be bound to the model element corresponding to the print command, and *?inter* would be bound to the model element that interprets input events for *?object*. H3 tests these conditions by querying the specification of the interface using the standard set of terms used to specify the interface of any application. In Figure 3. the standard terms are shown in pseudo-English (e.g., `has an interaction technique`). Without such a declarative model it would be very hard to define generic rules that will work for any application.

The right hand side of the rules (message description) consists of pieces of text, variables bound in the left-hand side of the rule (conditions), and function calls that recursively invoke help rules to generate embedded pieces of text. For example, `generate-activation-message (?inter)` recursively invokes the help system to generate the message that explains how to invoke an interaction technique. The variables have two effects: first, a description of their value is substituted in the text of the message, (e.g., the print command is substituted with the string "Print"), and second, when the message is shown to the user, H3 highlights the region of the display where the value of the variable is presented. In addition, H3 assigns the same color to the substituted text and the highlighting of the region.

H3 currently contains 32 rules similar to the one shown above. These rules cover the default messages generated for all the generic building blocks of our interface development system (buttons, menus, icons, etc.), the default messages needed to explain the notion of "currently selected object", and default messages that deal with notions of lists of elements that might be displayed in a variety of formats such as rows, columns and graphs.

H3 currently does not have rules that require complicated inferencing. For example, H3 can explain that a command button is dimmed because one of its preconditions is not valid, and can produce a help message explaining why the precondition is not valid, but cannot explain what the user needs to do to make the precondition valid. For similar reasons, H3 does not support task-oriented help either. Producing such help is beyond the scope of rule systems like ours. It requires a planner that can compute the sequence of commands needed to perform a given task [8].

Architecture

Figure 4. shows the architecture of H3. The input to the help system is a help request, which is a region in the application display. The outputs are the Help Window, and the highlightings of regions in the Application Window.

The top-left portion of the figure shows the main elements of the HUMANOID interface generation system [3, 9, 10]. The model describes the commands and objects of an application (Application), the methods for presenting these commands and objects (Presentation), and the behavior of these objects in response to input events (Behavior). The HUMANOID Runtime System uses the information in the model to construct the application displays, and to interpret inputs from the user. The help system uses the runtime system to figure out where the various elements of a display appear on the screen, and also as a bridge to query the model for the properties of the application, presentation and behavior that define the application displays. H3 uses the information in the model to construct the help messages.

When H3 receives a request for help, the Rule Matcher runs through all the rules in the Help Rules database, and for each rule whose conditions are satisfied, H3 makes an entry in the Message Database. Each entry consists of the message descriptions of the rule and the values of the variables that satisfied the conditions of the rule.

The Message Database is passed to the Message Manager whose job is to eliminate redundant messages, to group together sets of related messages and to sort the messages. The Message Manager makes use of qualifiers that can appear in the help rules. The qualifier :exclusive eliminates messages of the same information topic produced by other rules, except for messages with the qualifier :additional. Messages with the qualifier :collect are grouped together as long as they have the same text portions but different bindings for their variables; this allows H3 to produce sentences of the form "Clicking left selects X, Y, and Z", rather than "Clicking left selects X. Clicking left selects Y. Clicking left selects Z."

The Display Manager displays the help windows. It computes the presentation of the values of the variables by substituting them in the text of the message for their textual

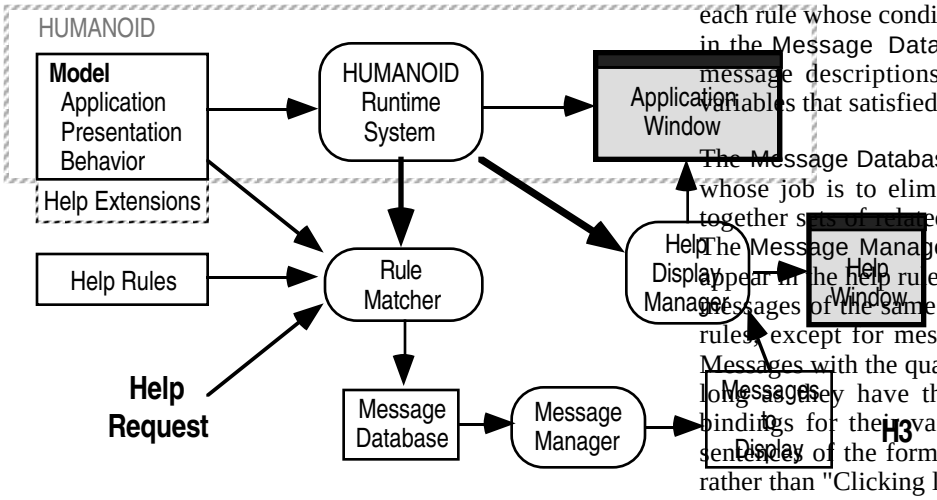

Figure 4. Architecture of H3. HUMANOID provides the enabling technology that allows H3 to access the information it need to construct the help messages and highlight the appropriate portions of the display.

representation, and by highlighting the regions of the Application Window where they are displayed. In addition, the Display Manager assigns colors so that the same color is used to display the value of the variable in the help message and to highlight it in the Application Window.

HELP SYSTEM CUSTOMIZATION

H3 supports several levels of customization. The simplest customization level allows developers to edit the text of the default messages to make it more appropriate for a particular application. The next level of customization allows developers to not only change the text, but also to add new links to a message. Defining new messages with simple conditions is similar to changing text and inserting references, and is also easy to do. The most complex level of customization is to add rules with complex conditions, which requires developers to learn the language for defining the conditions of rules.

The customization facilities use a programming by example paradigm, except for the facility for defining new rules with complex conditions, which requires entering a specification in the rule language. The help customization facility gives developers a window that looks exactly like the help windows for end-users (Figures 1 and 2), except that messages are editable, and there is an extra pane of buttons and menus to control the customization facilities.

Customization of Text

To change the text for a message, the help developer simply edits the text in the help window, and then selects the `Install` button. Before incorporating the new message into the help system H3 queries the developer to find conditions when the new message is appropriate. In general, the conditions need to be more restrictive than the conditions of the original message because they are adaptations of generic messages to specific situations. The customization facility supports the specification of additional conditions to restrict applicability of the message, and also allows dropping conditions to allow the message to appear in a wider set of contexts.

By default, H3 offers to constrain the rule so that the message appears only for regions using the same presentation method used to construct the object being explained, with the added conditions that the object appears embedded in exactly the same way inside other presentation methods. This prevents developers from inadvertently attaching help messages to reusable presentation methods that are used in other displays where the help message would not be appropriate. The developer has the option of dropping some or all of the context conditions to allow the message to appear in a wider set of contexts.

For example, suppose the developer wants to customize the message associated with the labels that show the name of a directory in the file manager application. The developer first asks for help as a user would do, by moving the mouse over one of the labels, and hitting the HELP key. The following message appears:

> The region for which you selected help shows a directory.

This is an instance of a default data message that displays the type of the object being shown. Suppose the developer edits the message to say :

> The region for which you selected help shows the name of a folder. Its attributes are shown next to it, and its contents are shown below.

H3 offers to constrain this message so that it appears only when the name of a directory is selected for help, with the added conditions that it appears inside a presentation method that presents the attributes and contents of a directory, and the directory appears as a child of another directory, and it appears inside a scrolling area, etc. In this case, the added conditions are too restrictive. The developer, would probably drop all the extra conditions except for the first one that constrains the message to only appear when the directory label is part of the presentation that displays its attributes and contents.

Adding References

When messages need to refer to objects other than the ones that appeared in the default message, developers must insert the references in the text. Two kinds of references can be added: references to objects displayed in the application window, or references to commands and inputs of the application, which might not be displayed in the application window.

References to objects displayed in the application window are added by clicking on them. The help developer first clicks on an object on the application display, and then uses the `Widen` and `Narrow` buttons to adjust the size of the region until the appropriate region is highlighted. H3 enters in the help text the expression that represents it (e.g., highlighted in light gray). H3 modifies the internal representation of the message to contain a function call that computes the new region based on the region originally selected for help.

For example, suppose the developer wants to modify the following message so that the second sentence contains a link that shows the user the region of the screen where the contents of the folder should be shown:

> The region for which you selected help shows the name of a folder. Its attributes are shown next to it, and its contents are shown below.

The developer selects the last two words in the previous message (the words "shown below"), and replaces them by a reference as explained above. The new message will read

> The region for which you selected help shows the name of a folder. Its attributes are shown next to it, and its contents are highlighted in light gray.

and the appropriate region of the application display will be highlighted in light gray. The internal representation of the message will contain the expression

```
get-part (parent (parent (?selected-object))
         :contents)
```

which means, go to the parent of the selected object, then go to its parent, and then find the part called :contents.

References to elements in the application model are inserted by bringing up a view of the application model in HUMANOID's model visualizer tool, and selecting the desired command, input or other component in that view. The result is also a parenthesized expression that computes the appropriate reference when the message is generated. Rather than using colors, the references are substituted by the name of the selected model element. For example, if the developer points to the model of the Print command, the reference will be substituted by the label "Print".

Defining New Messages

To generate new messages about an object that had no help attached by default, help developers enter the help refining mode, and then specify the object for which they want to specify help by simply selecting it in the application display. A menu allows them to specify the kind of help message they want to create, and the help window shows a default message for that kind of help. For instance, the default for data messages is

> The region highlighted in blue displays some data

The developer then proceeds to change the text and add references as explained above.

CONCLUSIONS AND FUTURE WORK

Our approach to the automatic generation of help text, and to the convenient customization of the automatically generated help messages, has several benefits:

- Low cost for adding help systems for applications. Default help comes for free, and there is an easy to tailor help to specific applications.
- Tight integration between help messages and application displays: users can point to the displays to ask for help, and the displays are highlighted to show the objects referred to in the text of the messages.
- Help messages are dependent on context (e.g., the message for the name of a file can be different depending on where in a window the file appears), and are also dependent on the state of the application (e.g., the message for dimmed buttons is different from the message of enabled buttons).
- Support for keeping the help system up to date when the application or the interface are modified. The automatically generated help messages will always be up to date because they are generated from the model that defines the application and its interface. Customized messages can be flagged as potentially needing revision because they include references to the model (this last feature is not currently implemented).
- Consistent interface to help systems across applications.

Future work on H3 will proceed along two directions. First, we need to get feedback from help system developers and end-users regarding the usability of the help generation tool and the quality of the generated help. We have built help systems for two simple applications (the folder manager and an object browser), and are currently working on constructing help for a large knowledge base development environment. We expect to have to refine the interface to the help system itself (i.e., the Display Manger). The display manager has not been the emphasis of our work, but we isolated it from the rest of the help system, so that we can refine it without impacting the rest of the system.

The second direction is to enhance the sophistication of the help system. We are interested in incorporating ideas from Cartoonist to be able to show animations, and to produce task-oriented help. Given that the models used in HUMANOID and Cartoonist are similar, we expect to be able to use their approach [2, 8].

ACKNOWLEDGEMENTS

Roberto Moriyon is supported by a grant from the Spanish Ministry of Education and Science. Pedro Szekely and Robert Neches are supported by ARPA through Contract Numbers NCC 2-719 and N00174-91-0014.

REFERENCES

1. Feiner, S., and McKeown, K.: Coordinating Text and Graphics in Explanation Generation. Proceedings AAAI-90. Boston, MA, 1990.

2. Foley, J., Kim, W.C., Kovacevic, S., and Murray, K.: UIDE - An Intelligent User Interface Design Environment. Intelligent User Interfaces. J.W. Sullivan & S. W. Tyler, Ed., Addison Wesley, 1991.

3. Luo, P., Szekely, P., and Neches, R.: Management of interface design in HUMANOID. In Proceedings INTERCHI'93. April 93.

4. Macintosh System 7. Apple Computer. 20525 Mariani Ave. Cupertino, CA 95014.

5. Microsoft Windows. Microsoft Corporation. One Microsoft Way. Remond, WA 98052.

6. Moore, J.D., and Swartout, W.R.: Pointing: A way toward explanation dialogue. Proceedings of the Eighth National Conference on Artificial Intelligence, 1990.

7. IBM OS/2. International Business Machines Corporation. Old Orchard Rd., Armonk, NY 10504.

8. Sukaviriya, P., and Foley, J.: Coupling a UI Framework with Automatic Generation of Context-

Sensitive Animated Help. In Proceedings UIST'90, Oct. 1990.

9. Szekely, P, Luo, and R. Neches. Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design. In Proceedings SIGCHI'92. May 1992, pp. 507-515.

10. Szekely, P., Luo, P., and Neches, R.: Beyond Interface Builders: Model-Based Interface Tools. In Proceedings INTERCHI'93. April 93.