

Chapter 3

MODEL-BASED DESIGN OF ONLINE HELP SYSTEMS

Milene Selbach Silveira¹, Simone D.J. Barbosa^{1,2}, and Clarisse Sieckenius de Souza²

¹ PUCRS, Faculdade de Informática (FACIN)

Av. Ipiranga, 668, Prédio 30 – Porto Alegre, RS 90619-900 (Brasil)

E-mail: milene@inf.pucrs.br

URL: <http://www.inf.pucrs.br/~milene>

Tel: +55 51 3203558 – Fax: +55 51 3203621

² PUC-Rio, Departamento de Informática,

R. Marquês de São Vicente, 225, 4^o RDC, Gávea – Rio de Janeiro, RJ 22453-900, Brasil

E-mail: {simone,clarisse}@inf.puc-rio.br

URL: <http://www.inf.puc-rio.br/~simone,~clarisse>

Tel: + 55 21 3114-1500 ext. {4353,4344} – Fax: +55 21 3114-1530

Abstract Online help systems are typically used as a last resource in interactive breakdown situations. In this paper, we present a method for building online help based on design models according to a semiotic engineering approach. We show the benefits of having designers explicitly communicate their design vision to users, and we also point at the need to foster a new culture for online help. We show how this proposal opens a direct communication channel from designers to users, and we hope this will contribute to introducing this new culture.

Keywords: Design models, HCI design, Online help systems, Semiotic engineering.

1. INTRODUCTION

How can we help designers build online help for a computer application? And how can we ensure that it will adequately tell users what to do with the application, what the application is for, why certain design decisions were made, and so on? Typical online help systems do not help their users much. The reasons for this inefficiency may lie in lack of time or planning or in excessive confidence in the intuitiveness of a fail-proof interface, or in a naïve

acceptance of current standards [19]. Following Adler & Winograd's views [1] that attempting to build idiot-proof technology underestimates or hinders the users' intelligence and creativity to learn and transform software according to their needs, we believe the role of the online help system is to open new possibilities and give users resources to understand and go beyond the designer's original ideas, taking the most advantage of the technology. In this view, the construction of the online help system becomes a critical step in the design of human-computer interaction (HCI).

This work is based on Semiotic Engineering [7], which views the user interface as a message sent from designers to users, representing the designers' solution to what they believe is the users' problems. It is through this message that designers tell users what they have interpreted as being the users' needs and preferences, what the answer for these needs is and how they implemented their vision as an interactive system. In Semiotic Engineering, online help is an essential application component. This is where designers will explicitly "speak" to the users, revealing how the application was built, how it can be used and for what purposes.

This paper describes a model-based approach for online help system design, stemming from Semiotic Engineering and driven by two main pillars: communicability and the rhetorical layering technique used in the minimalist approach. In order to illustrate our approach, the design of a real application's help system is presented as a running example.

2. A SEMIOTIC ENGINEERING VIEW OF ONLINE HELP

According to Semiotic Engineering, it is essential that users understand the designers' message so that they may better use and take advantage of the application. One way to make this message explicit is through a careful design of the application's online help system. The goal of our research [21] is to promote a novel perspective on online help design and usage. Users should be able to express more precisely their doubts and needs, and designers should be able to anticipate such doubts and needs, and to organize their response accordingly.

Our proposal for online help design draws from two related works: communicability evaluation [15] and the layering technique used in minimalist documentation [9]. Communicability evaluation is a qualitative HCI evaluation method which reveals potential breakdowns during interaction. These breakdowns are indicated by colloquial expressions which users can associate to their needs for help content. The minimalist approach to technical communication suggests that using small pieces of very relevant contextual-

ized content is one of the best ways of providing information to users [5]. Its benefits are increased by the layering technique [9], through which related pieces of mutually accessible minimalist content are connected, and made available to users.

In our approach, users are offered a set of expressions from which they may choose one to indicate their doubts or trouble during interaction. In doing this, they obtain the contextualized help content associated to the chosen expression. For instance, if they can't see or interpret feedback for an action, users may ask "What happened?".

Some of the expressions used for accessing help were drawn from the communicability evaluation method, and others resulted from an analysis of existing help literature about users' most frequent doubts during interaction [4, 20]. A few of them are frequently found in commercial applications, such as "What's this?" and "How do I do this?". The current set of expressions we use for accessing help is:

What's this?	How do I do this?	What is this for?
Where is...?	Where was I?	What now?
What happened?	Why should I do this?	Why doesn't it?
Who can do this?	On whom does this depend?	Who does this affect?
Oops!	Is there another way to do this?	Help!

Both users and designers benefit from using these expressions: users have a greater chance of getting a relevant help response, and designers have an organisation principle directly driven by communicative breakdowns that they intend to circumvent or solve. Inspired by the layering technique [9], we allow users to access minimal pieces of help content about a certain user interface element or task first, and then access further help material, depending on their needs, and then on to as much further information as required.

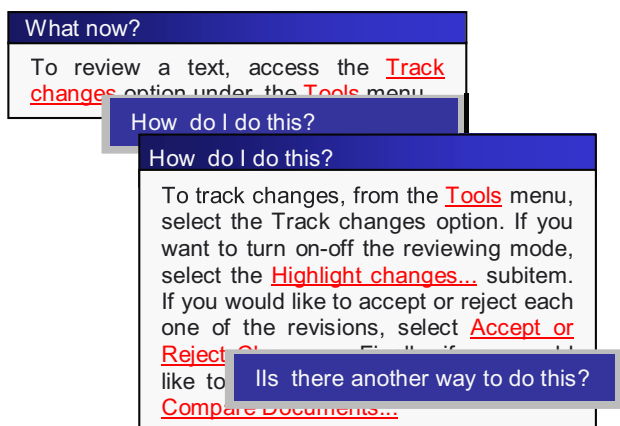


Figure 1. Help responses and their recurrence.

A simple example illustrating minimalist responses and the recurring use of expressions upon these is shown in Fig. 1. This is a fictitious example, based on some help content found in MS Word®, as a response to help requests about ‘tracking changes’. Our proposal includes a standalone help module, where it is possible to find related information about the domain and the application as a whole, as well as usage scenarios, so designers may convey their global design vision in a consistent way. However, in this paper we will focus on the local (contextualised) help only.

3. BUILDING ONLINE HELP SYSTEMS FROM HCI DESIGN MODELS

In this section we describe the extended HCI models involved in designing an online help system. We propose to use these models in a process comprising the steps illustrated in Figure 2.

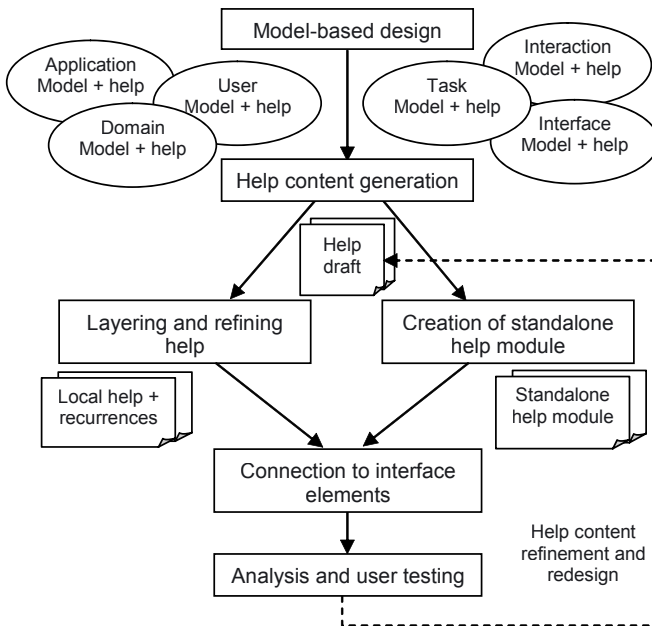


Figure 2. Steps used to build online help systems.

An implicit requirement of our proposal is that we capture design rationale, a task known not to be easy [10]. We should emphasize that the only relevant portions of design rationale for online help systems are those that directly affect the users’ experience. In other words, all decisions that are related to software design, architecture and the like are unimportant for help

purposes. The relevant information must be captured during the application design process. This is one of the reasons why we follow an extended model-based approach to HCI design. In general, model-based approaches in HCI promote the representation of interaction solutions in order for the designer to reflect on and make adequate design decisions. There is a variety of existing models: task models are the most widely used, but user, domain, presentation and dialogue models are also found in the HCI literature, among others [13,14,16,17,18,23,24,25]. We have carried out a series of studies about these models, in order to verify the possibility of reusing the existing information they provide in designing help and to identify the need for extensions. We have selected models for: domain, application, task, user, interaction and interface, and we have extended them to be able to represent information that is specific for help design.

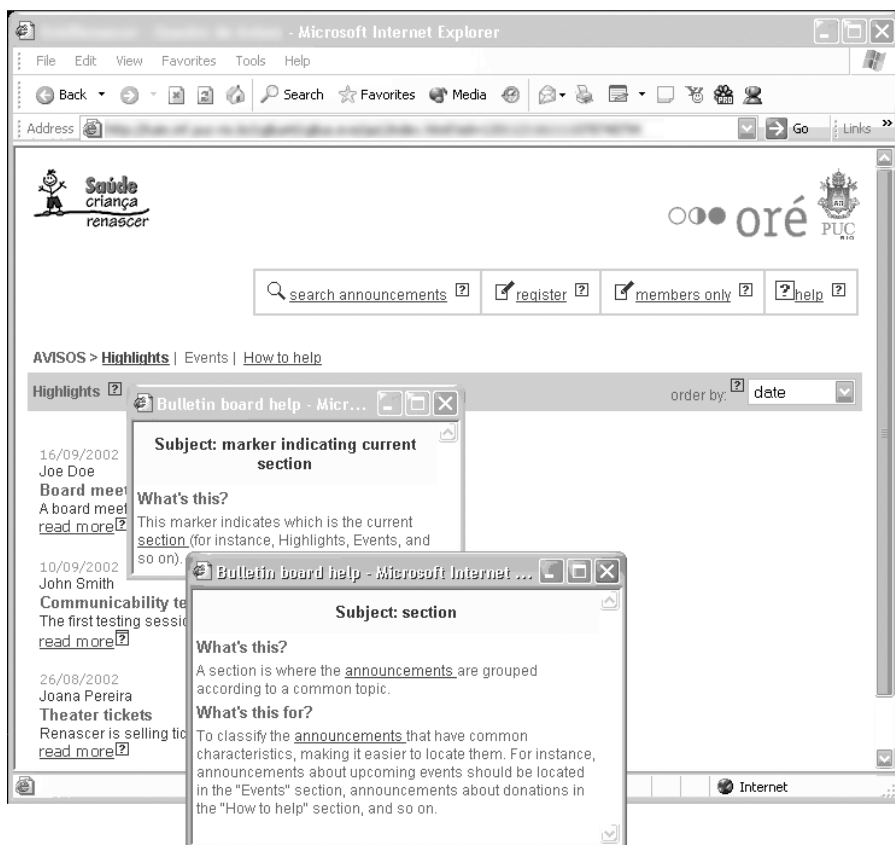


Figure 3. Help request and response.

In order to illustrate our approach, we will describe portions of an application we designed and developed for supporting the volunteer work in a

nongovernmental organization [12]. The module we have chosen for illustration is the Bulletin Board, in which volunteers and employees of this organisation may post and verify announcements related to their work or to the organisation as a whole. These announcements may be classified according to different topics or divisions within the organisation, such as: Administration, Events, Meetings, and so on. Figure 3 illustrates a usage scenario in the actual application, in which a member of the organisation has looked for help about a section marker.

3.1 Model-based Design of Online Help

The information necessary for the construction of local help responses and for the standalone help module is derived from HCI models built during the design process. At this point we would like to emphasize the need for the help designer to participate throughout the design process in order to capture this information necessary for building the online help. His presence is essential to ensure the timely capture of help-specific information. As the models are created, the help designer should record the model components in a database which, towards the end of the process, contains all necessary information for generating the draft help content. This way, the help design process becomes more efficient and provides better results, for the information not only reflects the design product, but also includes the rationale underlying some relevant design decisions. If this information isn't captured throughout the design process, much of it would be scattered across numerous representations, and possibly forgotten. We now describe the information comprised in each model, highlighting (in boldface) the pieces of information that are essentially driven towards help system construction.

- *Domain model*: information related to the application domain, focused on its description, the **nature of work performed**, and the information elements (domain signs¹) that belong to it.
- *Application model*: information related to the application being designed. The focus here lies in the application **description**, its **utility**, **advantages**, supported **activities**, **alternative courses of action**, and the application signs. Besides, it encompasses the roles users may play in this application, and for each role, the tasks related to it, and the necessary **basic knowledge**.

¹ A sign is a technical semiotic term that is usually taken to mean “something that stands for something else for someone”. In this sense, every piece of data represented in a computer application is a sign to the designer, and every user interface element is a sign both to the designer and to the user(s).

- *Task model*: information related to the tasks users may perform. For each possible task, we represent its **description**, **utility**, **reason** why it should be performed (from the designer's point of view), its parent task (considering a hierarchical task decomposition), the operator² that connects it to the following task, which establishes in which way it should be executed, the tasks' preconditions, and the related domain and application signs.
- *User model*: information related to the targeted application users. For each user we represent his name, the roles he may play, and his profile, which indicates the way in which he would like to interact with the application.
- *Interaction model*: information about the possible forms of interaction in the application, that is, about how to effectively perform a certain task in the application. For the execution of each task there may be alternative courses of actions. For each alternative, there is the **reason** why it should be executed (from the designer's point of view), its precondition(s), the indication whether it is the **preferred alternative** (from the designer's point of view) and the actions necessary for its execution. For each action, there is the default value, as well as the way to **undo** it, besides the operator that connects it to the next action.
- *Interface model*: information about the interface elements of the application. For each element, we represent its type, the values it may assume, its default value, its location at the user interface, and the related domain and application signs.

As an example, consider a piece of the domain model, related to a couple of domain signs in our case study:

```

DOMAIN SIGN Marker indicating current section {
  DESCRIPTION (This marker indicates which is the current section (for instance,
  Highlights, Events, and so on.))
  PURPOSE (To quickly indicate the current section.)
}
DOMAIN SIGN section {
  DESCRIPTION (A section is where the announcements are grouped according to
  a common topic.)
  PURPOSE (To classify the announcements that have common characteristics,
  making it easier to locate them. For instance, announcements about upcoming
  events are located in the "Events" section, about donations in the "How to help"
  section, and so on.)
}
and a piece of the task model:

```

² The operators considered in this version are those proposed in [14].

```

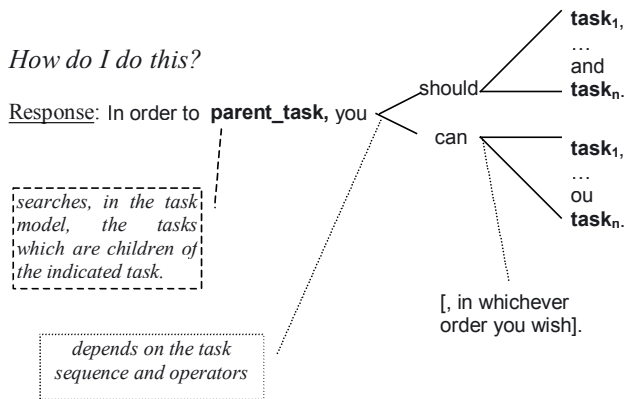
TASK Provide the required information {
  TASK PARENT(Create an announcement)
  OPERATOR (sequence)
  SEQUENCE (1)
  ...
}
TASK Confirm the operation {
  TASK PARENT(Create an announcement)
  OPERATOR (sequence)
  SEQUENCE (2)
  ...
}
    
```

3.2 Help Content Generation using Templates

For each element-expression combination, a minimalist response is designed. In order to generate a draft of this response, we have created a help content template associated to each expression. This template is instantiated with information from the different HCI design models. For instance, for the “What’s this?” expression, the content comes directly from the description of the related (domain or application) sign, which is represented in the corresponding (domain or application) model:

What’s this?
Response: description(**Sign**)

A more elaborate generation may be illustrated by the “How do I do this?” expression, related to a task. In this case, we have the following:



From the information contained in the models and these templates, a draft of the candidate help responses is generated for each pair expression-

element (where element may be a sign, task, alternative courses of actions, and actions). The help designer then selected which responses she would actually include in the application.

Let us consider the marker next to the name of the current section (“Highlights”). A sample response generated from the database is obtained as follows: This marker is a domain sign. The expressions related to (domain) signs are: “What’s this?”, “What is this for?”, and “Where is...?”. Taking as an example the expression “What’s this?”, and using the aforementioned template, the description element was retrieved from the domain sign component, resulting in the following draft answer:

This marker indicates which is the current section (for instance, Highlights, Events, and so on).

With regard to the task model, the response for the expression “How do I do this?”, related to the task “Create an announcement”, would be:

In order to create an announcement you should provide the required information and confirm the operation.

In case of procedural help content, animations may also be generated to show users how the interaction should occur, in a way similar to Cartoonist [22].

3.3 Layering and Refining Help and Creation of Stand-alone Help Module

Based on the interviews with users, domain analysis, and so on, the designer selects those elements which he believes users may have doubts or which he would especially like to explain or describe to users, and the expressions that will be used to access this help content.

As soon as these element–expression pairs are selected, the generated draft responses undergo a refinement process, in which technical communication specialists shape the text in order to better communicate the designer’s message to users. Having done that, each help response is analysed by the designer to verify the possible recurrence points it may comprise. These points indicate the elements in the response to which further help expressions and content may be associated. This content may, in turn, contain additional recurrence points, and so on, deepening the help content about certain interrelated topics.

In our example, the draft text of the selected responses was refined and analysed with the purpose of finding possible recurrence points. For instance, in the response to the expression “What’s this?”

This marker indicates which is the current section (for instance, Highlights,

Events, and so on.)

The designer has verified a reference to another domain sign, in this case “section”. She selected this word as a recurrence point within the response, and associated the expressions “What’s this?” and “What is this for?” to it. The template for “What’s this?” of a domain sign is: description(**Sign**); and the template for the expression “What is this for?” is: purpose(**Sign**). Thus, the response to “What’s this?” is:

A section is where the announcements are grouped according to a common topic.


And the response to “What is this for?”:

To classify the announcements that have common characteristics, making it easier to locate them. For instance, announcements about upcoming events are located in the “Events” section, about donations in the “How to help” section, and so on.

It is important to note that help responses are generated and refined beforehand and embedded in the application as static information, instead of being dynamically generated. This solution avoids execution delays in processing the possible expressions and responses for each element, and makes it possible to manually refine the generated draft responses, so that the manner of speech will seem natural and the communication will be more efficient.

Jointly with the layering and refinement of the help messages, the stand-alone help module may be created. In its most basic form, this module should contain help information about the domain and the application, as well as an explanation about how different kinds of help work (standalone and local). The domain and application portions of this module may be built and refined *pari passu* the construction and refinement of local help content. Afterwards, part of the local help content may also be included in the stand-alone module.

3.4 Connection to Interface Elements

Having refined the local help content, the expressions and their corresponding responses may be made available through the user interface, associated to the elements which, according to the help designer, may raise some kind of user doubt. This is achieved by adding a trigger or link to the corresponding help expressions and content. In our case study, the graphics designer created a symbol to function as a link to local help requests. The chosen symbol was an interrogation mark within a square, such as , placed next to the user interface element associated to the help expression

3.5 Analysis and User Testing

Having built the application prototype, the local help system and the standalone help module, the design team should carry out some preliminary testing in order to verify whether the expressions and the corresponding responses are consistent, as well as the general help information. Every connection to help should be tested, as well as every recurrent point within the responses.

Once the application is implemented it is ready for real user testing through communicability evaluation [15], to verify if the interface is conveying the designer's message and to investigate which problems might occur during interaction. In our case study, we set up a few sessions of communicability evaluation with six users with varying degrees of computer literacy. These users were selected specifically because they represented the majority of the targeted user population, as indicated by the analysis interviews.

In order to better observe interaction during testing and to be able to capture the problems that may occur in help usage, it is interesting that the help designer be present during observations, so that he may focus specifically on help issues. He may not only observe problems in accessing help expressions and in understanding their responses, but also find out user difficulties that hadn't been anticipated (and therefore had no associated help), or whose responses do not address the user's current problem. In our case study, this step made it possible to determine problems in the help content and in accessing help. These problems may be grouped into three classes: help content, "declining" help, and help culture.

Help Content. During testing, the help designer observed users having problems in situations unanticipated by her, which meant that the corresponding help was inadequate or altogether missing.

"Declining" Help. When a user, after many fruitless attempts to use the searching mechanism, made a local help request, she read the explanation, spontaneously said she understood it, but even so she decided to do something different from what was said in the help content, which made it impossible to carry out the task defined in the test scenario.

Help Culture. Most users did not access help, independently of their experience with the application or in using computers. There isn't a culture of asking for online help when you are in trouble. Few were those who asked for help and, when they did, some of them closed the help window before there was time for them to have read the help information. Only one user actually read the help text and followed what the explanation suggested.

4. FINAL REMARKS

In this paper, we have shown why, in Semiotic Engineering, online help system is an essential part of an application. It is through help that the designer can directly communicate with the application users, revealing the reasons underlying his design and how users may make better use of it. In this approach, model-based design is of utmost importance, for it makes it possible to maintain the consistency between the design products built at each phase. This allows the designer to create and convey a cohesive message to users, in order to increase their chances of making sense of his message.

The users may also express their doubts more directly using one of the available local help expressions during interaction. The response will be a fragment of the designer's point of view and rationale when designing the application. Moreover, users may delve deeper into the help content from the recurrence points available at each help response, in an indefinitely long chain of associations driven by their local needs. This process is associated to some fundamental concepts in semiotic theory, namely semiosis and abduction.

In addition to all technical and theoretical efforts, we should also pay attention to introducing changes in the way users perceive help. As a rule, users access help only as a last resort [6]. They may have had frustrating experiences in the past, or not even understand what help is for. So, without fostering a culture for using help, the most carefully designed help won't improve the users' experience with software. Since this proposal opens a direct communication channel from designers to users, we believe it is a first step towards the introduction of this new culture.

By following a model-based approach, the cost of extending current design practices to design help according to our view is reduced. Using design models as a basis for context-sensitive help generation has been explored elsewhere [11]. The main differences in our work rely on the communication-oriented mechanism for accessing help content, made available through communicability expressions. This mechanism empowers users, allowing them to choose between a set of expressions that express their immediate doubts in a more precise way, rather than relying on the designers' judgement of all the help content related to a certain user interface element.

In order to further increase the benefits of our work, our students at PUCRS and PUC-Rio are working on software tools for aiding the design of online help and integrating it to web and GUI applications.

REFERENCES

- [1] Adler, P. and Winograd, T., *Usability. Turning technologies into tools*, Oxford University Press, 1992.
- [2] Barbosa, C.M.A., de Souza, C.S., Nicolaci-da-Costa, A.M., and Prates, R.O., *Using the Underlying Discourse Unveiling Method to Understand Organizations of Social Volunteers*, in E. Furtado, J.C. Leite (eds.), Proceedings of Vth Symposium on Human Factors in Computing Systems IHC'2002 (Fortaleza, 7-10 October 2002), pp. ?.
- [3] Barbosa, S.D.J., de Souza, C.S., de Paula, M.G., and Silveira, M.S., *Modelo de Interação como Ponte entre o Modelo de Tarefas e a Especificação da Interface*, in E. Furtado, J.C. Leite (eds.), Proceedings of Vth Symposium on Human Factors in Computing Systems IHC'2002 (Fortaleza, 7-10 October 2002), pp. ?.
- [4] Baecker, R.M., Grudin, J., Buxton, W., and Greenberg, S., *Readings in Human-Computer Interaction: Toward the Year 2000*, Morgan Kaufmann Publishers, San Francisco, 1995.
- [5] Carroll, J.C. (ed.), *Minimalism Beyond the Nurnberg Funnel*, The MIT Press, Cambridge, 1998.
- [6] Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., and Shneiderman, B., *Determining Causes and Severity of End-User Frustration*, Technical Report, HCIL-2002-11, CS-TR-4371, UMIACS-TR-2002-51, University of Maryland, 2002, accessible at <ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/2002-11html/2002-11.pdf>
- [7] de Souza, C.S., *The Semiotic Engineering of Human-Computer Interaction*. The MIT Press, Cambridge, 2004.
- [8] de Souza, C.S., Prates, R., and Carey, T., *Missing and Declining Affordances: Are These Appropriate Concepts?*, Journal of the Brazilian Computer Society, Vol. 7, No. 1, July 2000, pp.26-33.
- [9] Farkas, D.K., *Layering as a Safety Net for Minimalist Documentation*, In [5].
- [10] Moran, T.P. and Carroll, J.M., *Design Rationale: Concepts, Techniques, and Use*, Lawrence Erlbaum and Associates, Mahwah, 1996.
- [11] Moriyon, R., Szekely, P., and Neches, R., *Automatic Generation of Help from Interface Models*, in Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'94 (Boston, 24-28 April 1994), ACM Press, New York, 1994, pp. 225-231.
- [12] ORÉ Projeto, 2002, accessible at <http://www.serg.inf.puc-rio.br/ore>.
- [13] Pangoli, S. and Paternó, F., *Automatic Generation of Task-Oriented Help*, in Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology UIST'95 (Pittsburgh, 15-17 November 1995), ACM Press, New York, 1995, pp. 181-187.
- [14] Paternó, F., *Model-Based Design of Interactive Applications*, Springer-Verlag, Berlin, 1999.
- [15] Prates, R.O., de Souza, C.S., and Barbosa, S.D.J., *A Method for Evaluating the Communicability of User Interfaces*, ACM Interactions, Vol. 7, No. 1, January-February 2000, pp. 31-38.
- [16] Preece, J., Rogers, Y., Sharp, E., Benyon, D., Holland, S., and Carey, T., *Human-Computer Interaction*, Addison-Wesley, Reading, 1994.
- [17] Puerta, A., *The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development*, in J. Vanderdonckt (ed.), Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur, 1996, pp. 19-36.
- [18] Puerta, A., *A Model-Based Interface Development Environment*, IEEE Software, Vol. 14, No. 4, July/August 1997, pp.41-47.

- [19] Purchase, H. and Worrill, J., *An empirical study of on-line help design: features and principals*, Int. J. Human-Computer Studies, Vol. 56, 2002, pp. 539-567.
- [20] Sellen, A. and Nicol, A., *Building User-Centered On-line Help*, in B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, Reading, 1990, pp. 143-153.
- [21] Silveira, M.S., Barbosa, S.D.J., and de Souza, C.S., *Augmenting the Affordance of Online Help Content*, in A. Blandford, J. Vanderdonckt, P. Gray (eds.), *People and Computers XV - Interaction without Frontiers*, Proc. of the Joint AFIHM-BCS Conf. on Human-Computer Interaction IHM-HCI'2001 (Lille, 10-14 September 2001), Vol. I, Springer-Verlag, London, 2001, pp. 279-296.
- [22] Sukaviriya, P. and Foley, J., *Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help*, Proceedings of ACM Symposium on User Interface and Software Technology UIST'90 (Snowbird, 3-5 October 1990), ACM Press, New York, 1990, pp. 152-166.
- [23] Tarby, J.-Cl, *The Automatic Management of Human-Computer Dialogue and Contextual Help*, in Proceedings of East-West International Conference on Computer-Human Interaction EWCHI'94 (St. Petersburg, 2-6 August 1994), Springer-Verlag, Berlin, 1994.
- [24] Vanderdonckt, J. and Berquin, P., *Towards a Very Large Model-based Approach for User Interface Development*, in N.W. Paton, T. Griffiths (eds.), Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99 (Edinburgh, 5-6 September 1999), IEEE Computer Society Press, Los Alamitos, 1999, pp. 76-85.
- [25] Vanderdonckt, J., Limbourg, Q., and Florins, M., *Deriving the Navigational Structure of a User Interface*, in M. Rauterberg, M. Menozzi, J. Wesson (eds.), Proceedings of 9th IFIP TC 13 International Conference on Human-Computer Interaction INTER-ACT'2003 (Zurich, 1-5 September 2003), IOS Press, Amsterdam, 2003, pp. 455-462.