The Role of Designer-Generated Scenarios in Developing Web Applications and their Help Systems

Milene Selbach Silveira[1] **Claudia de Castro Oliveira Monteiro**
Simone Diniz Junqueira Barbosa **Clarisse Sieckenius de Souza**[2]

TeCGraf, PUC-Rio
[1]also FACIN/PUCRS, [2]also DI/PUC-Rio

milene@inf.puc-rio.br, claudia@tecgraf.puc-rio.br, sim@les.inf.puc-rio.br, clarisse@inf.puc-rio.br

PUC-RioInf.MCC09/00 January, 2000

ABSTRACT

Due to the increasing changes in the working practices of organizations brought about by web technology, developers now face not only the new technological challenges, but also the communicative challenges involved in a multidisciplinary development team. Complex communication process involve human-computer interaction, human-human interaction mediated by computers, and human-human interaction *about* computer artifacts.

In this paper, we present a communicative approach that makes use of scenarios produced by designers and developers throughout the development of web applications. We describe a case study, focusing on the communicative aspects involved both in the *process* of software development, and in the users' interactions with the final *product*. The use we made of the scenarios is best illustrated in the final product by the help system, whose architecture is also described.

Keywords

Scenarios, web applications development, help systems, user interface design, semiotics

RESUMO

Devido as mudanças crescentes nas práticas de trabalho das organizações, advindas da tecnologia Web, os desenvolvedores estão enfrentando, atualmente, não só os novos desafios tecnológicos, mas, também, os desafios de comunicação envolvidos em uma equipe de desenvolvimento multidisciplinar. Processos de comunicação complexos envolvem interação homem-computador, interação homem-homem mediada por computadores e interação homem-homem *sobre* artefatos computacionais.

Neste trabalho, apresentamos uma abordagem comunicativa, que faz uso de cenários produzidos por designers e desenvolvedores, no desenvolvimento de aplicações Web. Descrevemos um estudo de caso, com foco nos aspectos de comunicação envolvidos no *processo* de desenvolvimento de software e nas interações dos usuários com o *produto* final. O uso que foi feito dos cenários é melhor ilustrado no produto final por seu sistema de help, cuja arquitetura está também descrita neste trabalho.

Palavras-chave

cenários, desenvolvimento de aplicações Web, sistemas de help, design de interfaces de usuário, semiótica

INTRODUCTION

Web technology is increasingly effecting changes in the way people work and collaborate with each other. No longer are general-purpose single-user applications considered the single most important computational assets in a working environment. In the software industry, this trend brings about the need for new design methodologies and techniques. This is evidenced by the rapid growth of the Computer-Supported Collaborative Work research field indicates.

As the demand for commissioned web applications grows, we need to embrace not only new technological challenges, but also the communicative challenges involved in participatory design [8, 16] — where designers and users must come to terms about which tasks the applications should support and how —, human-computer interaction design and development, and the management of human-human interaction *about* and *mediated by* computer artifacts.

The differences in how developers and users perceive and understand the most relevant and perhaps subtle aspects of the users' tasks encourage a new development approach, strongly based on a semiotic perspective [1, 6, 9, 12, 15], focusing on the communicative aspects permeating both the *process* of software development, and in the users' interactions with the final *product*.

In addition, due to the multidisciplinary nature of web technology, we have included in the development team members from the computing, linguistics, and graphics design communities. Our challenge is twofold: we must first make users aware of the new technology, and then develop an application that is prone to changing their own working processes. Scenarios [4] help communicate the tasks as they will be supported by the application, according to technological features and constraints.

In our approach, we make use of scenarios for developing the user documentation and integrated help system. In doing this, we keep information about the nature and context of each task readily accessible to users while they are interacting with the application.

In this paper, we describe in a case study the development process of a real application in which we adopted an approach that makes use of scenarios throughout the development process. The pervasive use of scenarios will become more evident when we describe our proposed help architecture. We conclude the paper with a discussion about the costs and benefits of our solution in comparison to previous software development processes adopted in our own research laboratory.

THE CASE STUDY

We have developed a workflow system for monitoring and supporting group activities certified by ISO 9000 standards. The application manages services provided by a company, through a series of forms the company technicians and managers must fill in. The goal of the application is to ensure conformity to norms and thus pursue the desired quality standards (Figure 1).

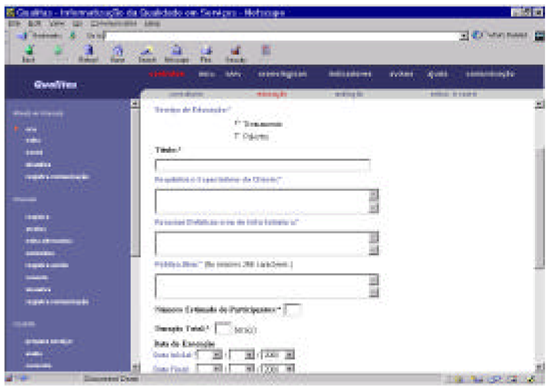

Figure 1 — Sample application page for form-filling.

In order to start modeling the users' domain and tasks, we have adopted a scenario-based approach [4]. Our initial goal was to establish a common understanding between the development team and the client, about two principal issues:

- What are the relevant processes and in which contexts they occur?
- How will these processes occur and change after the application deployment?

Since there was existing documentation about the standards and norms to which the application should conform, we made use of this documentation during development and in meetings with the client. This was done in order to guide and provide a common ground for the discussions about the application, and for the generation of scenarios. It is noteworthy that all scenarios were generated by designers, based on their understanding of users' needs and opportunities, as perceived in requirements gathering analysis.

Mostly, we used narrative scenarios, describing fictitious but plausible characters performing tasks corresponding to business processes within the context of the organization. We also recorded future scenarios, where we glanced at potential changes in the business processes themselves that could spontaneously emerge or be purposefully implemented in a near future.

Due to the strategic importance of the application, the development team at our laboratory found scenarios particularly useful as compared to previous systems analysis techniques. This is due to the fact that it maximizes the success in communicating intentions, requirements, and constraints among team members, and to the client. The client, on the other hand, may choose additional affordances of the product to be delivered, described within a context and at an adequate level of abstraction to ensure proper understanding.

The scenarios played a particularly important role in allowing both client and development team members to envision future extensions to the application and to the business processes. The participatory approach to design was decisive in that it ensured that the development team maintained the users' perspective in producing the scenarios.

Scenarios are frequently critiqued for their vagueness [10]. However, by including linguists in the development team, specializing in discourse analysis [3], we were able to minimize the breakdowns in understanding, phrasing, and communicating task and contextual information to the development team, and to the users.

Our research is motivated by a semiotic engineering perspective [6, 15], in which user interfaces are perceived as one-shot, higher-order messages sent from designers to users about how to exchange messages with the system. We claim that the degree to which a user will be able to successfully interact with an application and carry out his tasks is closely related to whether he understands the designers' intentions and the interactive principles that guided the application design.

THE ROLE OF SCENARIOS THROUGHOUT THE DEVELOPMENT PROCESS

During the early stages of the development process, scenarios are used as a requirements elicitation and system analysis technique. As the team proceeds in the development cycle, scenarios are typically generalized and abstracted in order to focus on the system's behavior [10]. In our approach, however, we use scenarios throughout the whole development process.

In Figure 2, we illustrate the various uses of scenarios along the development cycle. We have omitted the iterations between steps for sake of clarity.

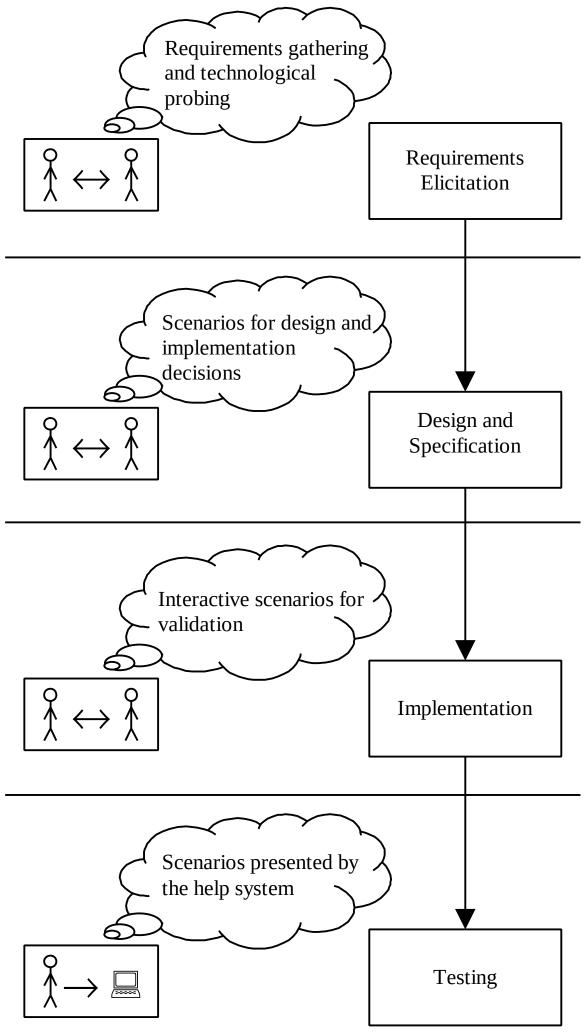
Requirements gathering and technological probing
Requirements Elicitation
Scenarios for design and implementation decisions
Design and Specification
Interactive scenarios for validation
Implementation
Scenarios presented by the help system
Testing

Figure 2 — Different types and roles of scenarios along the development cycle.

Whereas scenarios are usually created by users, the scenarios used in this case study were produced by the development team. We have done this because users generally describe only error-free scenarios, whereas developers are more aware of breakdown-prone situations and error-handling techniques. Therefore, developers have greater chances of uncovering most of the alternative scenarios, including cases of failure. (For a discussion about the importance of breakdowns in design, see [17]). These scenarios serve as a ground for negotiating application features, from general application characteristics to implementation details.

Task Scenarios

Our technique, then, consists of describing actual and potential (future) scenarios, and both normal and alternative courses of action. A typical scenario, describing a normal course of action, was cast as a narrative, including definitions and brief explanations, instantiated in real context of use. Alternative scenarios, however, were more difficult to understand, because the written documentation that served as a base for scenarios did not include exceptions or deviations from normal working practices. It was up to the development team to decide how many scenarios there should be, and how much detail to include in them. Future scenarios were used to call attention to critical aspects of introducing the proposed technology and how this would affect the working practices within the organization.

A summarized[1] example of a scenario describing a *user task* at the early stages of the development process is:

Normal course of action

A client contacts a technician in the company through fax, e-mail or telephone to request a training service. Having identified the need for the service, the technician verifies if the request matches the criteria in norm NG-04-X for the creation of a contrac. This norm states that the client should be a company division and the requested service must involve previous preparations.

After verifying the accordance to the criteria, the technician records the request as a contract proposal, using the "Services Contract" form. This form should contain, at least, the following information: client's expectations and objectives, intended audience, estimated number of participants, necessary resources, date, duration, and place of the service.

Alternative course of action 1

The request does not match the criteria established by norm NG-04-X:

This is a case where the client simply wants more information about the service(s). When this happens, the response is immediate, and there is no need to formalize a contract or enter data into the system.

Alternative course of action 2

Not all of the mandatory form fields are filled in:

If the technician does not input all the mandatory data, he should be unable to proceed with the creation of the contract, notified about the mistake, and prompted for the missing information.

Recording Design and Implementation Decisions

During the design and implementation stages, shorter scenarios were produced, in order to ensure proper interpretation of users' processes and tasks, and to record important design decisions. For instance, we had the following scenario that conflicted with the alternative course of action #2 described previously, and thus indicating a need for revising the design:

[1] The actual scenarios included much more details and contextual information, such as the names of the characters portrayed, the exact pieces of information provided and requested, as well as richer depiction of circumstances in which they took place.

The technician would like to use the application to record incomplete contract data, because he often gathers the information about a contract in different moments in time. He would like the forms to be checked for mandatory input only when he is about to complete the task.

Training Scenarios

In the training sessions, scenarios were used both in narrative and video formats. This technique helped users reach a different perspective on their own business processes. They gained new insights and clearer models of their *praxis*, which they had not been able to attain by reading the written processes documentation alone. The fact is that our scenarios represent their processes in a more expressive way than the actual manuals about these processes. This was so because they included and emphasized the technological consequences of having the system in a real situation. These characteristics played an important role in users' acceptance of and satisfaction with the application.

An example of a short scenario presented in video can be transcribed as:

A technician visits a client and gives a talk about environmental impacts of electrical plants. At the end of the talk, he distributes evaluation sheets to all participants. As he leaves the building, he trips up and all sheets fall in a mud pool.

As he gets back to work and enters the system to conclude the contract, he is prompted to input the evaluation data. Nowhere does the system allow him to explain what happened to the evaluation sheets. Thus, he quits the contract data entry module, and switches to a justification module. There he registers what happened and sends a message to his superiors. All this information is automatically attached to the contract he is working with and, as he gets back to the contract module, he is finally able to conclude it.

This scenario illustrates the designers' vision of exceptions that might occur in the process and how the implementation handles them.

Validating Scenarios

Right after deployment, an evaluation was made, in order to validate the new system. Two members of the development team (a computer analyst and a linguist) and one client representative generated a set of interactive scenarios to be performed through by users. Three users from different areas of the company were selected, in order to cover all the processes supported by the application.

A scenario used for validation consists of a numbered sequence of *interaction* steps, which the users should follow, as in a script. Users can make observations for each task, if needed. For instance, one of the scenarios included the following set of tasks, in which the user was the main character:

1. *Open a late contract (request made after the service date).*
2. *Select a pre-existing client.*
3. *Fill in the service data:*
   i. Define the initial date (prior to today)
   ii. Define the final date (prior to today)
4. *Fill in the schedule (service stages) with more than three stages*
   i. Consult warning messages
5. *Fill in the acceptance date (prior to the service date)*

6. *Indicate related documents (include verification list and the evaluation sheet)*
   i. Fill in the verification list
   ii. Record the results of the evaluation in the evaluation sheet
   iii. Record successful results and opportunities for improvement

7. *Conclude the contract*
   i. Inform the contract products thinking about frequent situation
   ii. Conclude the contract

8. *Termination*
   (The contract will be sent for consideration and termination to the area manager).

One of the tests presented the following observation related to step 5:

> *Although the task described in the script indicated that the acceptance date was to be prior to the service date, the interface did not present any such information, and in a real situation I wouldn't have got it right.*

This observation and many others of the kind motivated a change in the application design, resulting in a more sophisticated user support by different kinds of help information and access, as will be described in the next section.

As a result of this evaluation, we could communicate through the tests the consequences of introducing the new technology. Furthermore, users could re-think their own work practice. The application was gradually qualified and tailored to suit users' newly perceived needs. Moreover, users felt the need to modify their own procedures in order to take more advantage of the technological advances, and some potential enhancements were discovered.

Although there were only a few users in this evaluation, we adopted a position similar to that of discount usability evaluation [14, 13], which states that only a few users are necessary to reveal most of an application's problems.

Now that we have described the role of scenarios throughout the development process, let us proceed to discussing their impact on the final product. We have chosen to present the help system architecture, because this portion of the application gives the best evidence of the effects of using scenarios in the way we have described.

HELP ARCHITECTURE FOR WEB APPLICATIONS

A major concern in help systems' development is to keep in focus the users' needs for understanding both system functionalities and how users can employ them for their own particular purposes within a real context of use.

In order to capture these needs, we have engaged in an investigation task whose main goal was to evaluate the usability of existing processes manuals and documentation, in order to guide the development of the help system [5]. This evaluation was made by interviewing users. They answered general and specific questions and participated in simulations of real situations of use by means of scenarios. The questions were classified in three major groups: user characterization or profiling; evaluation of the documentation usability; and assessment of the users' knowledge of the business processes and norms included in the documentation.

The results of this preliminary evaluation emphasized the usability and applicability [7] of the documentation, and motivated its use as a basis for the construction of the on-line help system. The help system design should be consistent with the application design, and the following issues were identified as desirable to be included in it:

- explanations about the correct form-filling procedures, indicating mandatory and optional fields;
- an on-line version of the written process documentations, including an index and links to previous versions;
- direct and contextualized links to the on-line process documentation, from within the application;
- frequently asked questions about the system use;
- clear indication of the kind of service referred by each help topic;
- context-sensitive help for interactive features.

Taking this analysis as a starting point, the proposed architecture was developed, emphasizing local and contextualized help access. Our main goals are to save users' time and effort in locating the necessary information, and to guide them through the interaction. In addition to a traditional help module, which contains the application user's guide, technical and functional features, we inform users the next steps to be carried out within the current task context, and provide local help for each input field in a form.

Our architecture is an attempt to minimize the major problems users report with respect to existing help systems [11]:

- difficulty in switching between the help system and the application;
- help systems don't provide the specific information desired;
- help information is not available when needed;
- help information is not accurate or is incomplete;
- help information is difficult to understand;
- help systems are confusing to use.

The next subsection describes the architecture used in our case study, and which is currently being extended and refined.

Proposed Architecture

The proposed help architecture is composed of the following modules, where either users or system initiate the dialogue.

(a) *Main Help Module*: this module (Figure 3) is activated when the users select the <help> option from the application's main menu. It is presented in a new window — apart from the application . It contains the application's user manual (available functions, technical specifications, form-filling instructions, etc.), as well as documentation relative to the institutional policies and internal procedures and mechanisms for communication via e-mail with the support team. These functions are accessible via menus and hypertext links.

Besides including traditional help information in the user manual, we add interaction scenarios in the format of narrative and movies. The movies present an actual user interaction with the application when carrying out a given operation, including mouse movements, the selection of options, form-filling, etc. This technique allows users to understand how to carry out a sequence of tasks in a real situation.

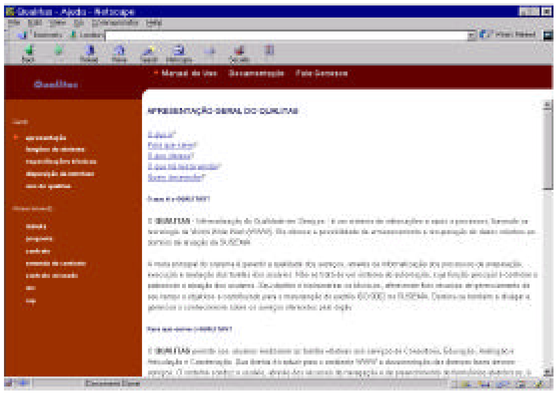
Figure 3 — Main Help Module.

Furthermore, this module includes links from the explanations about the application's form-filling to the corresponding process documentation, in order to present not only syntactic information about a given input element, but also about the meaning of the information to be provided by the user. As described earlier in this paper, this documentation is included in its entirety in the help module, and accessible through an index that contains a brief description of each norm or procedure, as well as links to its last versions and official revisions.

The communication mechanism is illustrated in Figure 4. It provides users with a means of asking direct questions to the application support team. These questions may be categorized as "system problems", "suggestions or comments", "future requirements", etc. Each of these categories represent a kind of user need the development team anticipated. For other kinds of needs, users can describe the specific kind of need they have, which can be used in the future for creating a knowledge-base for an explanation system.

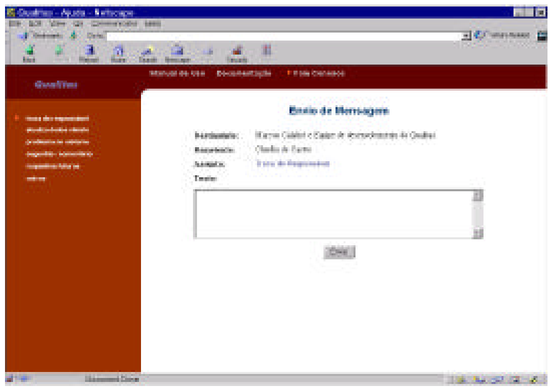
Figure 4 — Screen-shot illustrating communication mechanism.

(b) *On-demand Hints on Interactive Features*: this module (Figure 5) is activated when the user clicks on some field label afforded as a hypertext link. This action opens a small pop-up window on top of the current application window, with an explanation about how to fill in the corresponding field in terms of syntax (for instance, "the field must be filled with alphanumeric characters in the format XA234") and/or semantics (the domain task to which the field is related).

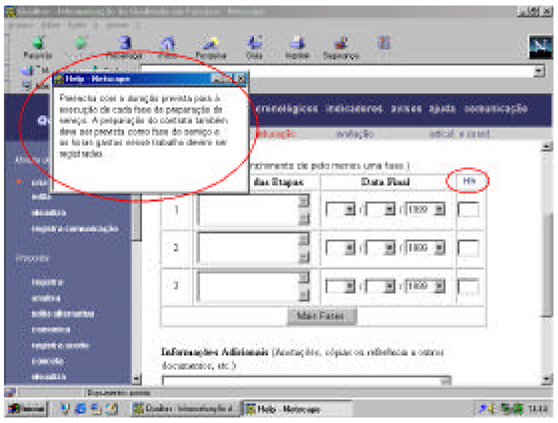

Figure 5 — Hints for the HH field and corresponding

The availability of this module ensures faster access to focal information, also present in extended form in the main help module. When this is the case, a direct link to the corresponding topic within the main help module is provided. Such links may occasionally point to organizational documents, as well as to system functionality details and scenarios themselves.

(c) *Explicit Next-Step Instructions*: this module (Figure 6) is not activated by the user, but directly afforded by the application itself. It presents direct messages or comments from the designers to the user, about how he should proceed with the interaction in a given situation. Such comments are presented in the bottom-part of the screen. Sample comments would be "To go on, click on the **analyze** item, within **proposal**, on the left frame" and "Please wait for the endorsement of the supervisor. If you are authorized to endorse it yourself and wish to do so, click on the **endorse** item, on the left frame".

This module helps maintain the context of the process by indicating the sequence of its constituent tasks. Whenever the user needs to wait for an action to be performed by another user, or a task stage is concluded, designer messages are presented, indicating the next step to be carried out. This avoids the situation where the user waits for a feedback from the application, while the application is waiting for an input from the user.

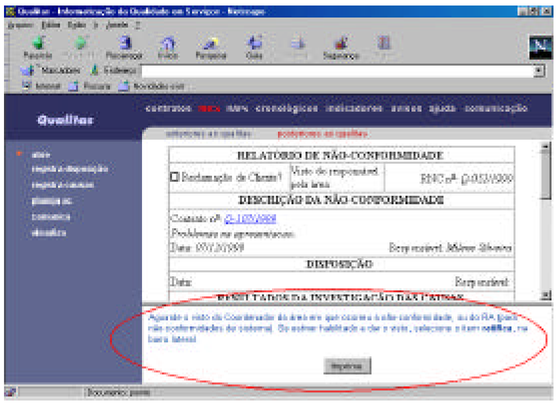

Figure 6 — Next-Step Instructions.

Analysis of the Proposed Architecture

Besides the topics detected by analyzing the aforementioned users interviews, Table 1 presents a comparison between the most frequent doubts users have [2], and how these are addressed by the proposed architecture:

Types of Question	Sample Questions	Proposed Architecture
Informative	*What kinds of things can I do with this program?*	**main help**
Descriptive	*What is this? What does this do?*	**hints**
Procedural	*How do I do this?*	**main help + hints**
Interpretive	*What is happening now? Why did it happen? What does this mean?*	n/a
Navigational	*Where am I? Where have I come from and where can I go to?*	**next step instructions**
Choice	*What can I do now?*	**next step instructions**
Guidance	*What should I do now?*	**next step instructions**
History	*What have I done?*	n/a
Motivational	*Why should I use this program? How will it benefit me?*	**main help**
Investigative	*What else should I know? Did I miss anything?*	**main help**

Table 1 — Users' doubts and the corresponding help modules.

The architecture supports users with faster response to simple doubts through the local help module, and also gives them a sense of confidence in the process sequence, because of the designer's messages about the next actions to be carried out.

Our current research topics include extending the architecture to provide help modules for dealing with the questions not addressed in the current proposal. We will also build a help editor where the original application designer may guide the application maintenance team in what they should do and how they should do it. In other words, we would like to make the inclusion of hints and next step instructions easier, as well as the inclusion of new functions in the user's manual, in order to guarantee the evolution of the help information in accordance with the application evolution, taking into consideration the designer's intents.

EARLIER DEVELOPMENT APPROACH

A previous application was developed using traditional system analysis techniques, such as requirements elicitation and system modeling. Although the earlier application also followed a prototypical approach, the communication between the development team and the client was not as successful as the more recent approach, using scenarios.

The development of the previous application did not follow a participatory design method, nor did it use scenarios as a means of communication between developers and client/users. As a result, the client could not see the opportunities brought about by the new technology, whereas the clients of the system developed using scenarios detected many opportunities for improvement of their own working practices, as well as potential extensions to the application. This discrepancy was aggravated by the absence of regulating documentation in the previous one, where it was the system that helped create a manual about the company's processes.

Another important difference between the two approaches was due to the homogeneity of the previous development team, consisting of computer analysts and programmers alone (a graphics designer came into the process too late to effect real changes in the application design), as opposed to the multidisciplinarity of the second development team, with members of other areas, such as Linguistics and Graphics Design.

With respect to the resulting web applications, we noticed a striking improvement in the help system, related to content as well as to structure. The architecture supports just-in-time help information, in context, with fast local access, and with direct messages from the designers to the user about the task he is currently carrying out.

The most remarkable differences between the two development approaches are summarized in Table 2:

	Earlier Approach	**This case study**
methodology	traditional systems analysis, using prototypes	scenario-based-like, with scenarios produced by the development team, as well as prototypes
development team	computer analysts and programmers (and a graphics designer late in the process)	computer analysts and programmers, linguists, and a graphics designer
user involvement	users took part on the initial requirements analysis, and later during prototype evaluation	users participated in all meetings about the design, throughout the development process
help system usability and applicability	traditional help only, without a study about its usability	development of a help architecture, with specific modules for different users' needs of information and access
effect of the introduction of the new technology	short-vision about the application, without an insight about the technology itself	insights about technological opportunities for future enhancements and extensions

Table 2 — Comparing the earlier approach to our proposed scenario-based one.

DISCUSSION

In this paper, we have shown how the use of scenarios throughout the a real application development process contributed to the quality of the final product. The qualitative comparison between the approach used in this case study and an earlier one gave us an indication of the benefits of using scenarios developed by a multidisciplinary team within a participatory design model, culminating in a new help architecture for increased user support.

A noteworthy aspect of our approach is that the developers themselves produced the scenarios. This is desirable because the developers can uncover alternative scenarios and error conditions that users would not consider *a priori*. On the other hand, it was very important to have a client representative as a member of the development team, as in participatory design, because it maintained the users' tasks and needs as a priority in design. This ensures that developers will not impose to users their own view of the technology, detached from the user context.

The multidisciplinarity of the development team contributed to the successful creation of the scenarios, since discourse analysis techniques proved to be crucial for minimizing inconsistencies and misunderstandings in communication among the members of the development team, and between them and the client.

The use and evolution of a set of scenarios throughout the development process, distinctly from other scenario-based approaches, helped users have more insight not only about the application, but also about the technology involved and the nature of their tasks and work practices.

The effects of our approach in the final product are made evident by the integrated help architecture. We believe our help system is the most powerful mechanism of designer-to-user communication. The architecture provides a variety of modules for specific users' needs for information and ease of access, including a direct communication mechanism that can be used for referring to the help system itself, and making it possible to revise the help content and structure based on actual users' needs.

Due to the hypertextual nature of web applications, the help system also provides links between the related pieces of information. Following a communicative perspective, we make use of these links to create a hyper-discourse about the application.

ACKNOWLEDGMENTS

The authors would like to thank TeCGraf/PUC-Rio, PUCRS, FAPERJ, and CNPq for providing support to this work. They would also like to thank the Semiotic Engineering Research Group at PUC-Rio and the Qualitas Team for their contributions to some of the ideas presented in this paper.

REFERENCES

1. Andersen, P.B. *A Theory of Computer Semiotics*. Cambridge. Cambridge University Press. 1990.
2. Baecker, R.M. et al. *Readings in Human-Computer Interaction: toward the year 2000*. Morgan Kaufmann Publishers, Inc., San Francisco, 1995.
3. Brown, G. e Yule, G. *Discourse Analysis*. Cambridge University Press, 1983.
4. Carroll, J. (ed) *Scenario-based Design: Envisioning Work and Technology in System Development*. John Wiley and Sons, New York, NY, 1996.
5. Cunha, C.K; Dahis, G. *Avaliação da documentação da SUSEMA como fonte para um sistema de Help On-line*. Unpublished manuscript, 1998.
6. de Souza, C.S. The Semiotic Engineering of User Interface Languages. *International Journal of Man-Machine Studies*. No. 39. pp. 753-773. 1993.

7. Fischer, G. “Beyond ‘Couch Potatoes’: From Consumers to Designers”. In *Proceedings of the 5th Asia Pacific Computer–Human Interaction Conference*. IEEE Computer Society. 1999. pp.2–9.
8. Greenbaum, J. & Kyng, M. *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum, 1991.
9. Imaz, M. & Benyon, D. “How Stories Capture Interactions”. *In Proceedings of Human-Computer Interaction – INTERACT’99*. IFIP, 1999.
10. Jacobson, I. “The Use-Case Construct in Object-Oriented Software Engineering” In Carroll, J. (ed) *Scenario-based Design: Envisioning Work and Technology in System Development*. John Wiley and Sons, New York, NY, 1996.
11. Kearsley, G. *Online Help Systems: design and implementation*. Ablex Publishing Corporation, Norwood, 1988.
12. Nadin, M. “Interface Design and Evaluation - Semiotic Implications”. In Hartson, R. and Hix, D. (eds.), *Advances in Human-Computer Interaction*, Volume 2, 45-100. Norwood, NJ. Ablex. 1988.
13. Nielsen, J. “Scenarios in Discount Usability Engineering”. In Carroll, J. (ed) *Scenario-based Design: Envisioning Work and Technology in System Development*. John Wiley and Sons, New York, NY, 1996.
14. Nielsen, J. *Usability Engineering*. Academic Press, 1993.
15. Prates, R.O.; Leite, J.C; and de Souza, C.S. (1998) “Semiotically-Based User Interface Design Languages”. *Proceedings of IHC'98 -- I Workshop de Fatores Humanos em Sistemas Computacionais*. Maringá, PR, Brasil. October 12-13, 1998.
16. Schuler, D. and Namioka, A. (eds.) *Participatory Design: Principles and Practices*. Hillsdale, NJ, Lawrence Erlbaum, 1993.
17. Winograd, T.A. and Flores,F. *Understanding computers and cognition: a new foundation for design*. Reading, MA. Addison-Wesley. 1987.