Task-Based Assessment of Web Navigation Design

Marco Winckler, Philippe Palanque, Christelle Farenc, Marcelo Pimenta

Task-Based Assessment of Web Navigation Design

Marco Winckler
Philippe Palanque
Christelle Farenc
LIHS – IRIT Université Toulouse 3
31062 Toulouse Cedex 4, France
winckler@univ-tlse1.fr
{palanque, farenc}@irit.fr

Marcelo S. Pimenta
UFRGS/Informática
Caixa Postal 15064
CEP 91501-970
Porto Alegre – RS, Brazil
mpimenta@inf.ufrgs.br

Abstract

Even though the importance of Web applications is still increasing, design practice in this area is still mainly based on informal techniques and methods. Currently, few descriptive techniques are available to support web modeling and in practice, it is unusual to model web applications even though modeling is usually the corner stone of "classical" computer applications. Models not only formalize requirements but also they can help to assess the design in all the stages of the development process. In this paper we present notations dedicated to web navigation modeling and how they can affect the design process. In addition we discuss how task models could be used to assess navigation models according to user's activity. Our aim with this kind of evaluation is to ensure (prior to implementation) that important users tasks can (or cannot) be performed with the system under construction.

KEYWORDS: WEB DESIGN, NOTATIONS, NAVIGATION MODELING, TASK MODELING, USABILITY EVALUATION.

INTRODUCTION

In early development of web applications, little importance was given to interface usability and user requirements and, as consequence, a lot of web applications have failed to meet their users' expectations and this was mainly due to bad designs. Lately, the use of informal user-centered design techniques has provided considerable help to improve usability of web sites by taking into account user requirements on the interface. For example, card sorting [19] and questionnaires [6] have been made popular as tools to identify user requirements and to structure information in a natural way i.e. according to users' point of view. Although simple, it is now recognized as a significant improvement with respect to previous practice.

However, web design is still made in an informal way. For example, there is little support to describe in complete and unambiguous manner navigation in web applications even though navigation is considered as a critical element. Currently, designers have some tools for supporting requirements analysis, such as storyboards and sitemaps [13], but they don't have support to translate these requirements into site navigation and design. This is acceptable for small web applications such as personal web sites as designers can cope intuitively with the small number of requirements but that approach does not hold for larger web sites [7].

Problems with informal approaches are quite well known and the most significant ones are:

i) To detect and to cope with ambiguous requirements;

ii) To go from design to implementation phases in a structured and reliable way;

iii) To cope with the complexity of the design that increase exponentially with the size of the web applications;

iv) To cope with modifiability that is a critical point for web applications.

Currently, only few descriptive techniques such as StateWebCharts [22] are available to model navigation for Web applications. Designers can consider modeling navigation as a cumbersome task but it provides them with many benefits. Those benefits are the same as the ones for modeling dialogue in interactive systems; modeling allows:

i) To cope with navigation in an abstract way i.e. without coping too early with details;

ii) To separate design from implementation issues;

iii) To reason about models in order to check properties (e.g. is it always possible to reach pages in one click, is it always possible to come back to the home with less than 3 clicks, etc.);

iv) To provide developers with a complete and non ambiguous description of the navigation thus avoiding design choices at implementation time;

v) To compare navigation models with other models built in the early stages of the development process such as task models or scenarios.

Task models are claimed to describe not only how users perform their tasks but also when and why those tasks are performed. Even thought task analysis has proven its effectiveness for interactive software development only few studies have dealt with users' task for web applications. Tauscher and Greenberg [23] describe some patterns of navigation followed by users, such as how users revisit pages on a web site, but their results don't explain which tasks are engaged while these patterns are used. Byrne et at [3] have created a taxonomy of user tasks for the web, based on analyses of most frequent tasks performed by users while using web applications. These

studies try to describe user tasks at a high and generic (activity) level but don't provide any information about how task modeling could be performed for a specific site or how a task model can be exploited within the development process of a web application.

We argue that both navigation and task modeling can help designers to build more usable and more reliable web sites. In this paper we precisely describe how StateWebCharts (SWC) notation dedicated to navigation modeling can be synergistically used together with task models in order to identify some usability problems in various stages of the development process. Our main focus is on coupling the two models by using task modeling to assess navigation models. As we can verify if a specific task can be performed on a navigation model we can show that the navigation model supports effectively users' activities.

DEVELOPMENT PROCESS FOR WEB APPLICATIONS

In general, Web designers start by gathering user requirements for the Web application. This information is then structured in a hierarchical way (sometimes based on card sorting results) and is at the basis of navigation design. Low-fidelity prototypes, based on paper and pencil and parallel design approaches, could also be employed to test and to improve designs in the early stages of development.

In previous work [18], Scapin et al have shown that development process of Web applications should be heavily based on formal notations and should follow an iterative process where the traditional requirements engineering phase should be followed by site specification.

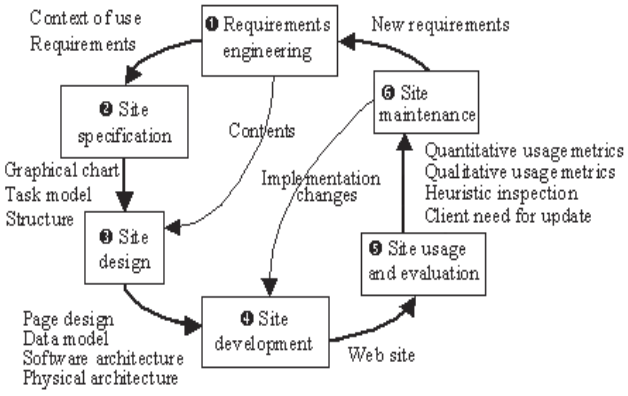


Figure 1. Development life cycle of a Web site.

The site specification (phase 2, figure 1) produces a comprehensive description of the site (through the construction of requirements, tasks and navigation models) can help in many ways the site development by formalizing user requirements, guiding the design and development of the site, providing useful information during usability evaluation and site maintenance phases.

In order to evaluate the web application one or more usability evaluation methods could be employed. Most evaluation methods require an in-advanced-stage prototype or feedback from users. As for "classical" interactive systems, it is critical for the development process if usability problems found during the evaluation step require significant changes on the web application. This calls for methods allowing usability evaluation during early stages development of Web sites.

In next section we present the main characteristics of the SWC notation that has been designed for describing navigation in a complete and unambiguous way. Task modeling for web applications is then presented. Before exploiting these two models for assessing the usability of web application we present the use of these two notations on a simple but real case study.

NAVIGATION DESIGN

Two main kinds of tasks are relevant to web design: search and navigation, and in this paper we only deal with the latter. Indeed, search task is mainly supported by search engines, a kind of stereotyped software system with a very simple user interface. Navigation design deals both with the organization of information on the web site and provides support for user tasks that could be accomplished by exploring the web site.

Small web applications such as personal homepages (less than 10 pages) could be created and maintained without major problems but navigation complexity increases very quickly on larger web applications. Broken links, ghost pages, long paths and complex navigation are frequently reported on usability testing and are symptomatic of the difficulty to design efficient navigation for large web applications.

In order to deal with this difficulty, some navigation models have been proposed such as OOHDM [20], WebML [5], extensions based on Petri nets [21] and statechart [24]. By using a navigation model designers can manage complexity and plan efficient strategies for navigation. However, none of the descriptive techniques introduced above represent all the important requirements for modeling Web applications, such as:

- ***Modeling dialog control***; that means, who is dispatching events over the interface (user or system);
- ***Borderline for design***; distinguish clearly the frontiers for Web design i.e., which parts of the application "*belongs*" to the designer (are part of the designer work) and which ones does not (such as external links);
- ***Designing Client-server activity***; which parts of the interface are processed on the client side and which ones are processed on the server side;
- ***Taking in account direct access***; support design taking into account that individual parts of the interface can be reached directly by users (without following a predefined path).

In this paper, we use SWC notation to model navigation because it copes with all the requirements above. SWC is an extended model of traditional statecharts [9], which is a state-based, event-driven notation. Each individual Web

page is considered a container for objects and each container is associated to a state. Links and interactive objects causing transition are represented by events. The semantic for a state is: current states and their containers are visible for users while non-current are hidden.

In SWC notation (figure 2) *Basic states (a)* represent states with normal HTML content; *Server states (b)* represent information being processed on the server side and, therefore, not visible for users; *Dynamic states(c)* represent states for which the content is dynamically generated by the system; and, *External states (d)* refer to states outside current design. The notation also includes two aggregation states, *XOR (e)* and *AND (f) states*. Inside *XOR states* only one sub-state is visible at a time. For *AND-states,* for each area separated by a dashed line, one state may be visible. *XOR states* are used to represent hierarchy of states while *AND states* are used to model multiple visualizations.

In order to represent special behavior such as those found in state charts, SWC provides the following pseudo-states *(g) initial, (h) final, (i) shallow* and *(j) deep history*. These pseudo-states don't have any container associated to them. In addition, SWC represents through transitions both user and system activity over the interface. Continuous arrows represent *user transitions (h)* and dashed arrows represent *system/completion transitions (l)*. Both *completion* and *system transitions* describe system's activity according completions or system events associated to transitions.

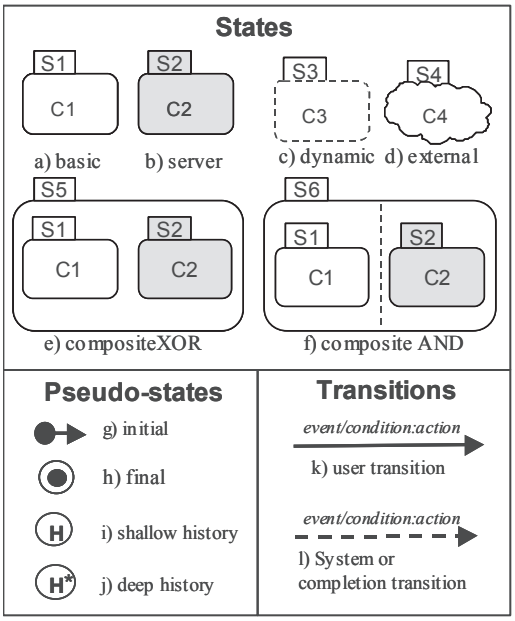


Figure 2. Graphical representation of StateWebCharts

TASK MODELING FOR WEB

Work on "classical" interactive systems has shown the central role played by task analysis for designing usable and useful systems [1, 2, 10]. However, even though task modeling is widely considered as helpful activity, the actual use of task models for the design of web applications remains an open question**.** Indeed, traditional approaches for designing web applications do not provide any guidance on how to integrate task models into the design process.

When designing navigation, we have to pay attention to users' mental model of the application as well as to provide efficient navigation for the most important users' tasks. The problem with the web is that there are many potential users and as many ways to use the web site. This is one of the reasons why it is so complex to determine the set of user's task that have to be considered in the design phase.

In fact, current approaches usually focus on the designer's point of view about the content and the navigation of the web application. The design process typically starts from an informal description of the content or a flat hierarchy, then the design is "implemented" in a try-and-error cycle heavily supported by tools like editors (for example MS FrontPage) and driven by guidelines [11]. User's perspective is thus only introduced informally and usually in an implicit way through testing and interviews. When dealing with large web application this informal process reaches its limits and often leads to usability failures. In order to include the perspective of users in a formal way we propose hereafter a design process that includes explicitly task modeling in the early phases.

GENERIC TASK MODELING FOR WEB

This section presents various results of task analysis for web applications. Users' tasks for web application belong to two main categories: high-level and primitive tasks.

- High-level tasks generally are very close to user's goals and may be performed in several ways. This kind of tasks enables designers to understand the fundamental aspects of user's activity and is usually independent from the way they are performed.
- Primitive tasks are more detailed as they correspond to a given activity of the user on the system. These tasks are generic for most web applications such as the ones engaged by a user while browsing. As initial set of such primitive tasks, we consider the taxonomy proposed by Byrne et al [3].

Byrne et al. have studied user tasks on the web and they have created taxonomy for labeling the most frequent ones. This classification includes the following categories:

- *Use information*: describes what users do with found information e.g. print page; save to disk; etc.
- *Locate on page*: describes users' strategies to identify pieces of information on web pages e.g. locate something interesting; find an image; etc.
- *Go to page*: describes any activity that causes the browser to display a particular page, e.g. select a hyperlink on a page, provide URL, use bookmark, etc.
- *Provide information*: describes users' activity when sending information to a web application e.g. send query string; fill in a form; etc.

- *Configure browser*: describes tasks for configuring web browser e.g. adds new bookmark; change cache size; etc.
- *React to environment*: describes tasks for which the browser requires something from users e.g. respond to dialog; respond to display change; etc.

The categories *use of information*, *configure browser* and *react to environment* describe activities that are completely independent from web design. So, from the set of tasks identified by Byrne et al. we have only selected the primitive tasks relevant for navigation design, i.e. *locate on page*, *go to page* and *provide information*.

It is important to note that we use the word "abstract" for the tasks described, as they are relevant in most of web applications. The models describing those tasks are detailed enough to be considered as concrete tasks models as they include very low level tasks such as, for instance, "inform url" in Figure 3. This detailed description of the abstract tasks allows us matching task models and system descriptions to check their conformance and compatibility.

CONCURTASKTREE NOTATION

For describing user's tasks we are using ConcurTaskTree notation [16] (CTT). Even though originally designed for describing user's activity for "classical" interactive systems it is suitable for web applications [17].

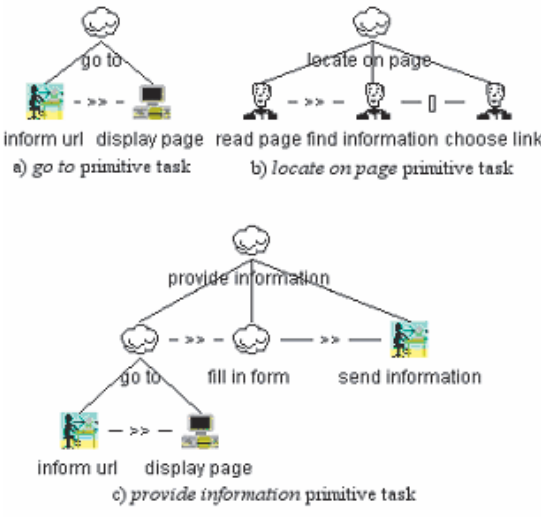


Figure 3. Task primitives in ConcurTaskTree notation

In this section we are using CTT to formally describe the primitive tasks *go to, locate on page* and *provide information*. ConcurTaskTree notation features a graphical representation for modeling tasks in hierarchical structures. Its expressive power, allows representing interactive activities and more precisely i) the concurrent ones related to the Web usage, where many windows/applications may be opened at a time; and ii) possible interruptions associated to browsing activities.

Figure 3 contains the task primitives described using CTT.

One of the objectives of this modeling is to allow reusing these primitives to describe more complex user tasks. For instance, in Figure 3 primitive task *go to* is reused for representing the primitive *provides information*.

CTT presents 4 types of tasks: *abstract, user, system* and *interactive* task. *Abstract* tasks are tasks with sub-tasks for example *go to* task and *locate on page* (figure 3). In CTT *User* task means cognitive or manual tasks performed by users, such as *read page, find information* and *choose link* for example (figure 3.b). *System* tasks are tasks performed by the system, for example task *display page* (figure 3.a and 3.c). *Interactive* tasks are used for representing activity for which both the user and the system are engaged, for example *inform URL* and *send information*.

ConcurTaskTree features a set of operators to indicate temporal relationships among tasks [16]. A set of tools (called CTT Environment) has been developed to edit, simulate and analyze task models in ConcurTaskTrees. CTTE has been used to build our modeling.

A CASE STUDY

In this section we present a case study for the HIBAM (Hidrografia da Bacia Amazônica) web application, whose purpose is to provide on-line information about almost 10-years of hydrographic research about Amazonian region (Brazil). This case study aims at:

- Showing how StateWebChart notation can be used for describing several strategies for navigation,
- Showing how CTT and primitive tasks models can be used for describing high-level tasks,
- Showing that it is possible to exploit synergistically these two notation in order to check whether a navigation model supports a given task model.

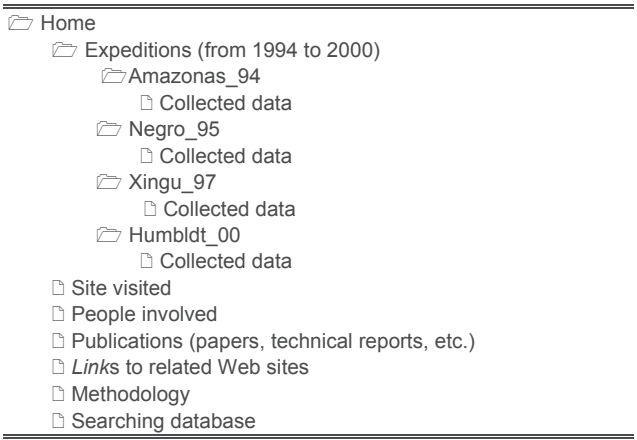


Figure 4. Hierarchical model for HIBAM web site.

Initially, card sorting was used to produce a hierarchical model describing the information structure of the site. For space reasons Figure 4 only presents a simplified hierarchy of HIBAM web site. The structure is the same as in the real site but repetitive elements such as the number of expeditions (more than 20) have been removed.

Navigation Modeling

Following a parallel design process, we have built two different designs (*A* and *B*) for HIBAM website navigation, presented in figure 5 and 6, respectively. We don't have any assumption about which design could easier to use or preferable. Our aim is to discuss how to consider different design alternatives during the design process and how the notations used can help us assessing the designs.

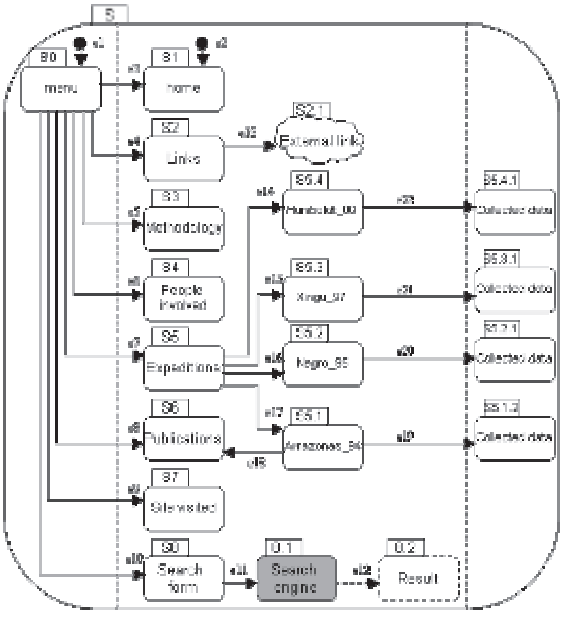
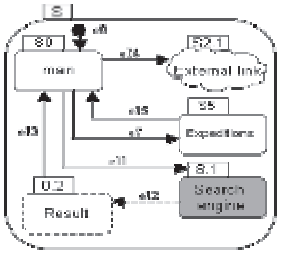


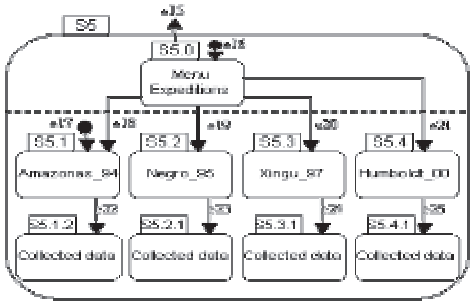
Figure 5. Design A for website navigation.

Figure 5 presents the first alternative for navigation that is based on two-frames design with a menu giving access for information. Menu (state *S0*) is always visible in this alternative. The startup configuration is states *S0* and *S1,* defined by default events *e1* and *e2*. When a particular expedition is selected, additional information is presented concurrently in a new window (see events *e19, e20, e21* and *e22* to states *S5.1.2, S5.2.1, S5.3.1, S5.4.1*, respectively).

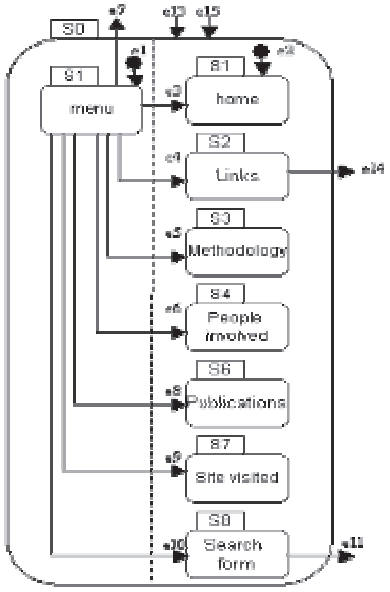
A second alternative (figure 6) was built by hiding menu when users navigate deeply in the hierarchy. This strategy cuts off part of information and forces users to pay attention to a specific content. StateWebCharts provides a special feature for describing hierarchies by hiding complexity in a multilevel system of states. In figure 6.a at the highest level (*level* 0), users start in state *S0* (main menu), which content is detailed in level 1 (figure 6.b). State *S5* presenting menu for expeditions is also detailed in level 1 (figure 6.c). When the user selects *expedition* by triggering the event *e7,* the current configuration is changed to a new one with two visual areas (a menu for all expeditions and another one for a particular expedition). Note that in state *S5* only the content related to expeditions is shown.



a) HIBAM - **Level 0**



b) Main - **Level 1**



c) Expeditions - **Level 1**

Figure 6. Design *B* for website navigation.

TASK MODELING

For this case study, we have select two common user tasks for HIBAM users:

i) *Visit external link*; the user must find a paper (for which the direct url is unknown) in another web site. However, the user is knows that HIBAM web site points to this document. In such a case, user's goal is to *use HIBAM web site as a starting point to find out other documents*;

ii) *Find publication from expedition*; for this task the user's goal is to *write a literature review for publications that were produced as a result of an expedition.*

These tasks are described using ConcurTastTree (see figure 7 and 8). In figure 7, task model considers that users can use the main page to get access to other websites. User must visit the page *home* at first (*go to home*), locate information on page (*select an external link*) and then *go to* the selected link. The task model in figure 8 is more complex than the previous one. Users have to see all publications (reports, papers, etc.) that were produced for a specific expedition. So, typically users have to select a specific expedition from the list of expeditions and then visit it. Then, users will find the associated publications.

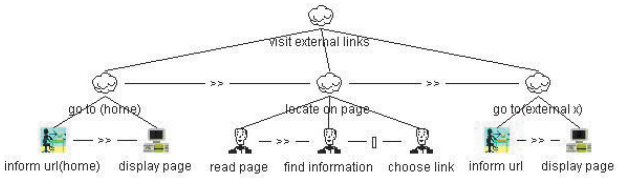


Figure 7. Task modeling for *visit external link*.

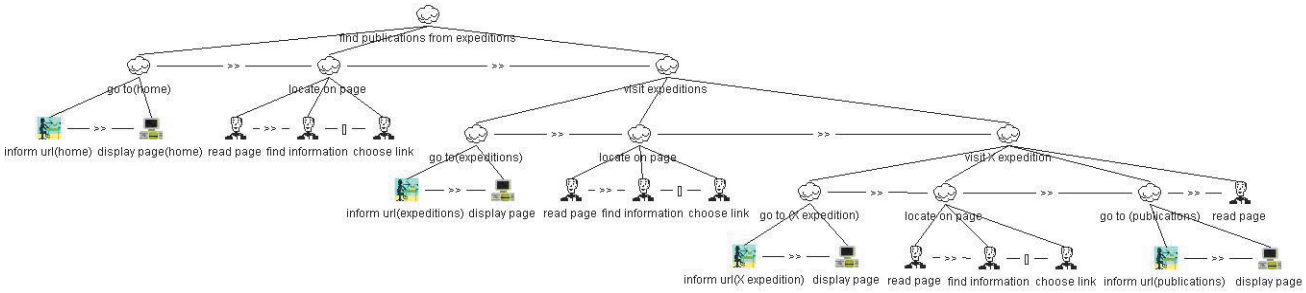


Figure 8. Task modeling for *visit publication from an expedition*.

ASSESSING NAVIGATION USING TASK MODELS

We have shown how we can describe navigation and user tasks using SWC and CTT for web applications. As stated above, those models are useful as they provide designers with explicit representations of both expected user's activity and navigation for the web application under construction. Besides, additional benefits can be obtained by cross validating these two descriptions of a same world.

More precisely the cross validation can check whether or not users can execute a specific task on a given web application. Scenarios extracted from task models are at the basis of the cross validation process. Using CTTE simulator it is possible to execute a task model. This execution results in a set of actions performed by the user and called scenario. As CTT allows for representing choice or concurrency in user's activity several different scenarios can be extracted from a given task model.

The assessment consists in executing one or more scenarios over a navigation model. Action by action the scenario is played and at each step visual changes are performed according to the navigation model. In order to perform the assessment the following steps must be carried out:

1) To extract scenarios from the task model;
2) To remove all non-interactive actions from the scenario. Figure 9 presents the scenario extracted from the task *visit publication from an expedition*. All bold elements in Figure 9 represent interactive tasks of the scenario. We can observe that we have only 4 interactive tasks: *inform URL – (home), inform url (expedition), inform url (x expedition), inform url (publications)*. Other tasks performed by the system such as *display page* or they are made on the users mind, for example, *choose link* and *read page* were removed.
3) To relate scenario and navigation model. Each interactive task in the scenario is associated to one state (or several states if concurrent visualization) in the navigation model. For example, the high-level task *inform url (home) (figure 8)* is associated to states *S0* and *S1* (figure 6). The result of this phase is a set of concrete scenarios as presented in Figure 10.
4) To check consistency between tasks and navigation models. This validation is done by running all concrete scenarios (figure 10) over the navigation model (figure 6). According to the structure of the navigation model a scenario might or might not be executed. For the scenario in Figure 10 be executable it must exist a transition in the navigation model in Figure 6 between state S5 and S5.1.

Inform url(home)
display page(home)
read page
locate on page
choose link
inform url(expeditions)
display page
read page
locate on page
choose link
inform url(x expedition)
display page
read page
locate on page
choose link
inform url(publications)
display page
read page
locate on page

Figure 9. A generic scenario extracted from the task *find publication from expeditions*.

Interactive Task	*Target state in navigation model*
Inform url(home):	**S0, S1**
inform url(expeditions):	**S5**
inform url(x expedition):	**S5.1**
inform url(publications):	**S6**

Figure 10. Concrete scenario associated to states in the navigation model.

ASSESSMENT AND THE DESIGN PROCESS

One of the contributions of the assessment proposed above is to clearly identify how many actions on the system are needed to perform a scenario. This is made possible because, a scenario could be matched to different paths in the navigation model and because we can explore all useful paths supporting the same scenario. Contrarily to other works that analyze navigation paths, this procedure allows designers comparing navigation paths to real tasks, which is supposed to give deeper insights about the use of the web application's user interface.

Assessing task models and navigation models can provide various benefits according to various phase of the development process (see Figure 1).

Early phases: These phases correspond to steps 1, 2 and 3 in Figure 1. We propose to use both notations (SWC and CTT) to explicitly describe and to document the design. Prior to implementation task-based assessment should be employed to verify if all previously identified tasks could be performed over the navigation model. In addition, the kind of checking could be synergistically used to verify many kinds of user tasks in different contexts of use, without having to change the navigation model. Thus, designers can compare how different tasks could be accomplished using the same user interface. On the other hand, given a set of predefined tasks, designer can verify the paths produced by playing the various scenarios over several design options. This compatibility checking is an efficient tool that allows designers choosing the best design option (that could be produced during a parallel design).

Usability Evaluation: This phase corresponds to step 5 in Figure 1. The task-based assessment could be employed during the usability evaluation step in conjunction with traditional evaluation methods. In this case, evaluators can, at the same time, play task scenarios over models and actual implementation. This allows designers to see visual aspects of the interface that cannot be seen while running the navigation model alone. Another advantage is that we can evaluate dynamically generated navigation. Indeed, SWC notation have a special type of state (*dynamic state*) that allows for representing the fact that dynamic content is added to site but it doesn't allow for representing the content itself. Lastly, during the *usability evaluation* step we can give to users the scenario and analyze how they perform scenario over the interface. Paterno and Ballardin [17] have already proposed this kind of remote evaluation.

Redesign: This phase corresponds to step 6 in Figure 1. The proposed approach is useful in many ways, but we believe that it is more particularly useful for website redesign which is one of the most critical phase of the development process of a web application. Designers have to ensure that the changes performed over the application not only support new users' tasks but also the previously identified ones. Redesign could promote changes by:

a) Taking into account new tasks for an application. If so it is important to check whether or not the navigation has to be altered;
b) b) Changing site structure and navigation. In this case designers must ensure that the tasks can still be performed over the new site structure.

In the steps of *early phases* and *redesign* the assessment is predictive, i.e. we can check design over the navigation model itself and thus prior to implementation.

RELATED WORK AND CONCLUSION

In this work we have presented how web application design could be fruitfully supported by navigation and task modeling. We have also presented how these two models could be synergistically exploited to improve the development process of web applications.

Model-based approaches for interactive systems have been proposed for many years but for web applications such concerns are only emerging [8]. Recently, some model for describing web interface have been published [5, 20, 22] but they mainly concern the system perspective of the application. User-centered design has been claimed as a mean to include user perspective during development process but even if tools exist to support UCD, it remains an informal approach [6, 10]. The studies concerning user task for web applications explore more user activity than task modeling and they provide little support to the design process [3, 13].

In the second part of this paper we have presented a method for the assessment of navigation design based on task modeling. This method is similar in many ways with our previous work for assessment of interactive systems through scenarios [12] and the work of Paternó and Ballardin [17] for web applications reengineering. The assessment kernel is the same in all case, that is, to match scenarios produced from task model with an interface. In [17] they propose to apply the assessment as part of remote usability testing and, for that, they match log files of user activity on the interface with tasks models. However, the evaluation only concerns the implementation of the web application while in the work presented here we perform evaluation over a navigation model. While their approach is helpful to identify usability problems on ready-to-use interface, it cannot be applied in earlier stages of the development process.

Navarre et al. [12] have developed a tool to execute CTT-scenario over applications built using the ICO formalism [15]. This is conceptually similar to the work developed in this paper but we believe it could not be applied for web applications. ICO formalism represents states in and

implicit way, as the current state of the application is represented by the value and the distribution of tokens over the models. States are central to web applications and thus must be at the core of the notation. This is one of the reasons for using Statecharts as the basis of the notation used. Web applications belong to the category of business application as they mainly process data. For this reason a notation such as ICO, able to model real-time complex safety critical applications embeds inherent mechanisms that would not be used for web applications and might produce complex models and thus jeopardize the effective use of the notation by designers.

Lastly, this work describes the use of the assessment technique in various phases of an iterative development process while related work [12], [17] was more focused on specific phases. In order to support the development process presented above a tool suite is currently under development, the first being the production of an editor for SWC and its inclusion in more generic tools supporting UML notations.

ACKNOWLEDGMENTS

We thank CNPq, Brazilian Council for Research and Development that has partly funded this research.

REFERENCES

1. Benyon, D. *Task Analysis and System Design: the Discipline of Data*. Interacting with Computers, 4(2), 246-59.
2. Bodart, F. et alli. *A Model-Based Approach to Presentation: A Continuum from task Analysis to Prototype*. Proc. of International Eurographics Workshop on Design, Specification and Verification of Interactive Systems, Bocca di Magra (La Spezia), 8-10 juin 1994.
3. Byrne, M.; John, B. E.; Wehrle, N. S.; Crow, D. C. The Tangled Web We Wove: A Tasknomy of WWW Use. CHI99 15-20 May, 1999. P. 544- 551.
4. Card, S.; Robertson, G.; York, W. The WebBook and the Web Forager: An Information Workspace for the World-Wide Web. CHI'96, April 13 - 18, 1996, Vancouver Canada.
5. Ceri, S.; Fraternali, P.; Bongio, A. Language (WebML): a modeling language for designing Web sites. 9th WWW Conference, Amsterdan, May 15-19, 2000.
6. Dong, J.; Martin, S. Interative Usage of Customer Satisfaction Surveys to Assess an Evolving Web Site. Proceedings of the 6th Conference on Human Factors and the Web, Austin, TX, USA, June 19, 2000.
7. Fowler, S. L.; Novack, A. J.; Stillings, M. J. The Evolution of a Manufacturing Web Site. 9th WWW Conference, Amsterdan, May 15-19, 2000.
8. Fraternali, P.; Paolini, P. Model-Driven Development of Web Applications: the Autoweb System. ACM Transactions on Office Information Systems vol. 18 (4), 2000.
9. Harel, D. Statecharts: a visual formalism for complex systems. Science of Computer Programming 8, P. 231-274. 1987.
10. Johnson, P. Human-Computer Interaction: Psychology, Task Analysis and Software Engineering, Mc-Graw Hill, Maidenhead, UK, 1992.
11. Lynch, P. J; Hortson, S. Web Style Guide: Basic Design Principles for Creating. Web Sites. Yale Univ Press. 1999. 164 p.
12. Navarre, D. ; Palanque, P. ; Bastide, R. ; Paternó, F. ; Santoro, C. "A tool suite for integrating task and system models through scenarios" in 8th Eurographics workshop on Design, Specification and Verification of Interactive Systems, DSV-IS'2001, June 13-15, 2001, Glasgow, Scotland.
13. Newman, M. W.; Landay, J. A. Sitemaps, storyboards, and specifications: a sketch of Web site design practice. Proceedings of DIS 2000, ACM Press New York, NY, USA. Pages: 263 – 274.
14. Nielsen, J. Failure of Corporate Websites. Alertbox, 98. http://www.useit.com/alertbox/981018.html
15. Palanque, P.; Bastide, R.; Paternò, F. Formal Specification as a Tool for Objective Assessment of Safety-Critical Interactive Systems. IFIP TC13 conference on Human Computer Interaction, Interact'97, Sydney, July 1997, Chapman et Hall, pp.323-330.
16. Paterno, Mancinii, Meniconi "ConcurTaskTrees: a Diagrammatic Notations for Specifying Task Models", Proc. of INTERACT 97, pp 362-69, july 97, Sydney, Chapman&Hall.
17. Paterno, F.; Ballardin, G. Model-Aided Remote Usability Evaluation. INTERACT'99, pp.434-442, IOS Press, Edinburgh, September'99.
18. Scapin, D. et al. Transfering Knowledge of User Interfaces Guidelines to the Web. In. Tools for Working with Guidelines. London: Springer; 2001; pp. 293-304.
19. Scholtz, J.; Laskowski, S.; Downey, L. Developing Usability Tools and Techniques For Designing and Testing Web Sites. 4th Conference on Human Factors & the Web, June 5, 1998,Basking Ridge, USA (1998).
20. Schwabe, D. et al. Engineering Web Applications for Reuse, IEEE Multimedia, Spring 2001, pp. 2-12.
21. Stotts, D. P.; Furuta, R. Petri-net-based hypertext: document structure with browsing semantics. ACM Trans. Inf. Syst. 7, 1 (Jan. 1989), pp. 3-29.
22. Winckler, M.; Farenc, C.; Palanque, P.; Bastide, R. Designing Navigation for Web Interfaces. In Proceedings:… IHM-HCI2001, Lille, França, 10-14 Sep. 2001 (Short paper).
23. Tauscher, L. ; Greenberg, S. (1997). Revisitation patterns in World Wide Web navigation. In Human Factors in Computing Systems: Proceedings of CHI97 (pp. 399-406). New York: ACM Press.
24. Zheng, Y.; Pong, M. Using statecharts to model hypertext. In. Proceedings of the ACM conference on Hypertext. Milan, Italy. 1992.