Towards a User-Centered Design of Web Applications based on a Task Model

Marco Winckler[1], Jean Vanderdonckt[2]
[1]*LIIHS-IRIT Toulouse*, [2]*Université catholique de Louvain*
winckler@irit.fr, vanderdonckt@isys.ucl.ac.be

Abstract

Since more than a decade, several methods for engineering and developing web applications have been introduced and extensively used. Since these methods often focus on data and related processes, their approach to conceptual modeling of web applications is centered on the notions of data, objects, functions, processes, and services. In this paper, we show that these methods could be expanded by modeling the user interface of such web applications by adopting a user-centered approach based on a task model. A task model represents the user's viewpoint on how to manipulate these data and trigger these functions so as to reach the goal associated to the task. Depending on the user, several different task models could be elaborated and each task model may lead in turn to different user interfaces for the same data and processes, as opposed to a single user interface as produced by traditional development methods. For this purpose, a case study is presented that demonstrates how a task can be modeled so as to represent the user's viewpoint in the user interface and to refine the dialog of the application.

1. Introduction

The development of Web applications is known to be as a complex activity due to many factors that need to be considered simultaneously: the evolving nature of applications, the multidisciplinary nature of development team, the competitive points of views for the application, the complexity and incompleteness of user requirements, the tight schedules for delivering the Web application [1, 6, 13]. In particular, many currently existing Web applications must follow predefined business logics and complex transactional operations and services requiring integration with distributed databases and legacy systems. Notwithstanding, the early Web was born as a hypertext/hypermedia system and it still preserves many of its hypermedia influence. As a result of this hybrid heritage, the development of many Web applications may tend to prefer considering aspects typically addressed in document management systems (e.g., information architecture), software engineering (e.g., functional architecture) [9].

To cope with this complexity, the Web Engineering community has investigated and defined models and notations intended to support the design activity of web applications that surpass the capabilities and the aims and goals of merely those found in document processing and software engineering. Several models for the development of Web applications have been proposed and extensively used in the recent years such as the OO-H method [8], the Object-Oriented Hypermedia Design Method (OOHDM) [15] and WebML [3]. Such models sometimes consist of an adaptation from previous work on Hypermedia Systems, Object Oriented methodology and Formal Methods. Those models which have been influenced by Hypertext Systems and Object Oriented methodology, propose solutions based on the concept authoring-in-the-large; which means they provide abstract models describing the overall classes of information elements and navigation structures without much concern for implementation details [7].

Most development methods existing for web applications base their conceptual modeling on the objects (or data) and their related methods, functions, or services. The consequence of this hypothesis is that the types of task that can be derived from these models frequently adopt the traditional CRUD (Create, Read, Update, Delete) pattern: tasks are limited to basic operations on objects and their relationships. When some methods go beyond this simple pattern, the dialog of the resulting user interface is assuming a transactional scheme: all data that are manipulated by the task are supposed to be entered and all methods that are required to accomplish the task should be triggered by the user. But no order is assumed, thus resulting in a dialogue where no contextual consideration is supported. For instance, such methods cannot model fine-grained dialogues such as: if the user selects this radio button, activate dynamically this push button associated to that service, enter information with this order that can dynamically change according to the user's preference, dynamically change a form according to user's reply, display this window according to the previously done operations. In other words, the dialog is often restricted to navigation between pages and screens, not to fine dialog.

Work on "classical" interactive systems has shown the central role played by task analysis for designing usable and useful systems [2, 11]. Task models allow the description of high-level user requirements in terms activities that must be performed by the user and/or by the system in order to reach some goal. The modeling produced with task models leads to many different implementations. One advantage of this is that we can compare different design options prior to the implementation, thus saving time.

Most of the user's tasks over the Web concern the navigation [6]. By navigation we understand here all activity allowing users to move from a Web page to another, which covers supplying information through forms (identified as part of electronic procedure using a database), following a navigation paths or freely exploring the information space.

This paper argues that the use of task modeling can be employed synergistically with navigation modeling to improve the usability of Web applications designs. The Section 2 describes a general method centered on user tasks. Section 3 presents a case study for a Digital Library which demonstrates our approach. Section 3.1 presents the user roles for the application. Section 3.2 presents the modeling of user users' tasks by the means of the ConcurTaskTree notation (CTT) [14]. Section 3.3 introduces the StateWebCharts notation (SWC) [18] and presents the corresponding navigation models for the digital library. Section 3.4 shows how to drive from navigation models with SWC to the prototyping of the digital library. Section 4 presents a discussion and related work. Lately, Section 5 presents the conclusions.

2. A Method for Web User-Centered Design

We assume that the phase of requirements engineering has been already started and designers have identified the users' profile, their informational needs as well as the underlying data model. The method presented here does not impose any particular notation. We used to describe the user profile and informational requirements by textual scenarios. The underlying data model is described by the means of a UML class diagram. The approach for modeling is made up by following these steps:

1) Identify the different roles performed by the user;
2) Create a task model for each role;
3) For each task model create an individual navigation model;
4) Create relationships in the navigation model for any further informational requirements;

The step one 1 is performed in the very early phases of development when the target audience for the Web application is set up. The tasks of each user role are specified by the means of a task model at the step 2. In the step 3 task models guide the process of navigation modeling; in addition, designers can include relationships to describe system behaviors as reaction to user interaction, which is not described by task models. The step 4 is made up by detailing the navigation in order to include relationships based on informational requirements (e.g. allow users to return to the main page of the Web site whenever the page they are navigating). Additional transitions and states can be included into the model to represent single pages, external relationships, index, guided tours or any other navigational requirements.

3. Case Study: Informal Description

Our case study is the digital library of theses for the French Association on Computer-Human Interaction (AFIHM[1]). The main aim with this Web application is allowing users to navigate, search and update a digital library of theses on the HCI field. The general idea behind this Digital Library (DL) is to allow users to feed the database with little effort and control by the AFIHM. Since some users could create unexpected records (e.g. supplying incomplete/inappropriate information, mistakenly changing a record) the Web application should support a kind of review process. This review process is made up by a system administrator who decides to give or not his authorization to publish a thesis in the catalogue after have been notified by mail each time a user submit his/her thesis. The informational and functional requirements for the application can be summarized as follows:

a) To provide searching and browsing facilities;
b) All visitors can create for free their own account
c) Only users having an account can submit and download thesis from the DL;
d) To ease updating the catalogue's contents by users;
e) To support a fast review process of thesis;
f) Allow users to navigate from the Digital Library to the AFIHM's Web site;
g) Notify users about the status of their submission (i.e. accepted, refused, etc.).

Due to space restriction, only a portion of the task models and navigation models produced for the case study are presented below.

3.1 User Roles and Tasks

Table 1 presents the roles which have been identified for the application ("everyone" and "system administrator") and their corresponding (allowed) tasks. The role "everyone" corresponds to any Web site visitor, which covers the profiles "Not logged in" and "registered user". The browsing and searching facilities of the database are available to everyone but a user only can submit or download a thesis in the electronic format if s/he is logged into the system. The user role "system administrator" refers to someone who is responsible for supervising the submissions.

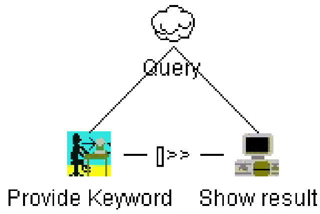
[1] AFIHM is the French acronyms of "*Association Francophone d'Interaction Homme-Machine*". More information available at: http://www.afihm.org/

Table 1. Target audience for the AFIHM's theses Web site.

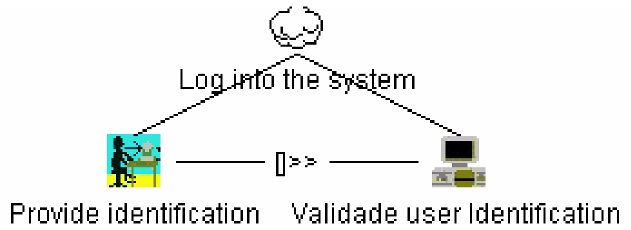
Role	User Profile	Pre-conditions	(allowed) Tasks
Everyone	Not logged in	none	Query (Search Digital Library) Create an account Log into the system
	Registered user	Logged in	Query (Search Digital Library) Download thesis Update account information Submit a thesis to the catalogue
System administrator	Have access rights Full control over the catalogue	Logged in	Review submissions

3.2 Modeling User Tasks for the Digital Library with the CTT Notation

We start by modeling individual the tasks without in-between dependencies such as "Query" and "Log into the system". Fig.1 presents these tasks by using the CTT notation [14]. In CTT, tasks are organized in a hierarchy; for example in Fig. 1.a, the task "Query" is accomplished when its subtasks "Provide Keyword" and "Show result" have been completed. The relationships between tasks are based on LOTOS operators such as enabling (operator >>), enabling with information passing (operator []>>), task interruption (operation [>), etc. CTT notation allows modeling 4 types of tasks: abstract task, user tasks, application task and interactive task. The Abstract tasks in CTT are tasks which require compound tasks such as "Query" (Fig.1.a) and "Log into the system" (Fig.1.b). User tasks are entirely performed by the user without interacting with the system (not used in this case study example). Interactive tasks are performed by the user with the system such as tasks "Provide Keyword" (Fig.1.a) and "Provide identification" (Fig.1.b). Application tasks describe actions performed by the system without user intervention, for instance "Show result" (Fig.1.a) and "Validate User Identification" (Fig.1.b). In the sequence, we designer can create more complex relationships between tasks. For example, Fig. 2 shows a complete task modeling for download a thesis form the Digital Library.



a) CTT model for the task "Query"



b) CTT model for the task "Log into the system"

Fig. 1. CTT models of individual tasks without in-between dependencies.

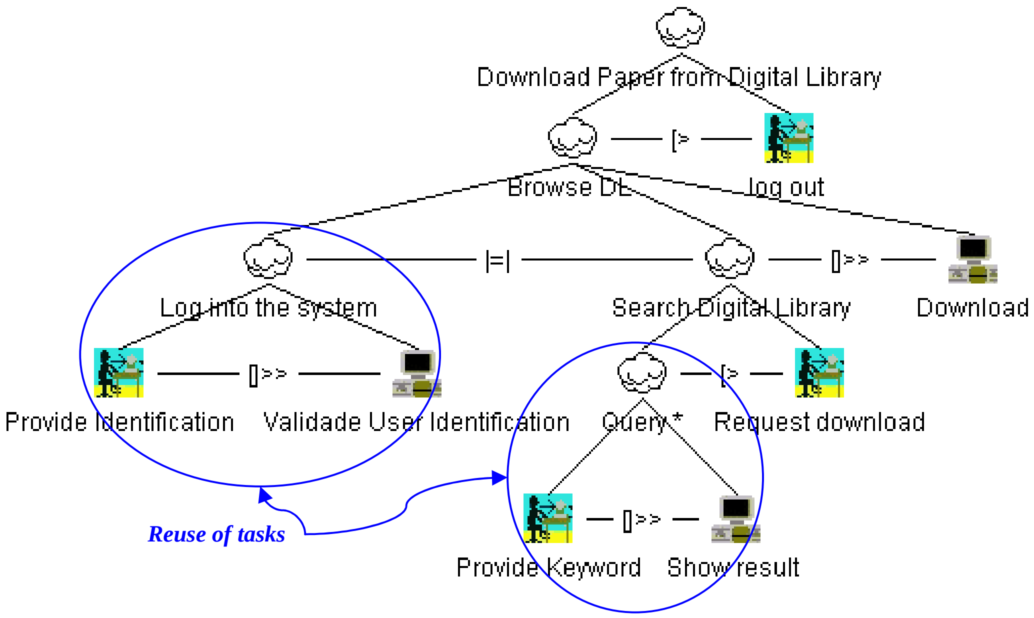


Fig. 2. CTT modeling for a download a thesis from the Digital Library.

It worth noting in Fig. 2 the reuse of tasks “Log into the system” and “Query”. The relationship (|=|) means that these tasks can be performed in any order. The system only performs the task “Send the file” after s/he has searched the digital library and get logged in the system can. In this modeling, the task “Query” is iterative (represented by the symbol *). To allow the interruption of this iteration the user can perform the task “Request download”. The task “Log out” was added to allow the users to exit the application at any time. The operator “[>” indicates the interruption of the task.

By exploiting to the task model above it is possible to perform several scenarios (see Table 2). The scenarios presented in Table 2 are used to evaluate all alternative sequences for the task “download a thesis from the digital library”. The scenarios below do not impose any particular implementation that means user tasks can be better understood without to have to planning how to support them by the system. This kind of analysis is made possible because user tasks are considered from the point of view of the users need for the application and not how to represent the user activity with a particular system.

Table 2. Some possible scenarios for the task “download a thesis from the digital library”.

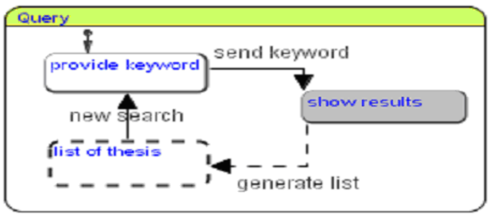
	Scenario 1	Scenario 2	Scenario 3
Sequence of task execution	Provide Identification	Provide Keyword	Provide Keyword
	Validate User Identification	Show result	Show result
	Provide Keyword	Provide Identification	Provide Keyword
	Show result	Validate User Identification	Show result
	Request download	Request download	log out
	Download	Download	

3.3 Modeling the Navigation for the Digital Library with the SWC Notation

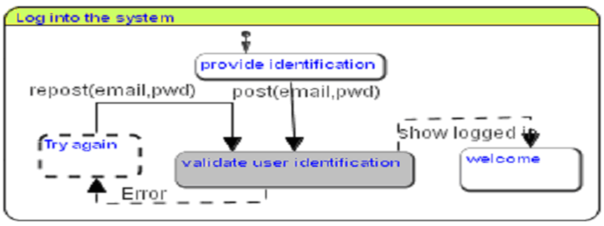
As described in previous section, some tasks require users to provide a keyword for querying a database or to provide their identification for getting access to private documents. In these cases, navigation models must represent what happens if the user identification fails or if the database records do not match to the keyword. These issues are better represented by navigation models than task models.

For navigation modeling we have proposed the StateWeb-Charts notation (SWC) [18]. SWC is a formalism based on StateCharts [10], which has been extensively used to model complex/reactive systems. StateCharts can be defined as a set of the states, transitions, events, conditions and variables and their inter-relations. In SWC, states are abstractions of containers for objects (graphic or executable objects). For Web applications such containers are usually (but not only) HTML pages. States in SWC are represented according to their function in the modeling. States can be static, dynamic, transient or external. Static states represent static content while dynamic states represent pages generated dynamically by the system. Transient states describe a non-deterministic behavior in the state machine; they are needed when a single transition cannot determine the next state for the state machine. External states represent external modules for the application of external Web sites. In a similar way, a SWC transition explicitly represents the agent activating it. Transitions whose event is triggered by a user are graphically drawn as continuous arrows while transitions triggered by system or completion events are drawn as dashed arrows. 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 operational semantic for a SWC is: current states and their content are visible to the users while non-current states are hidden. Users can only navigate outgoing relationships (represented by the means of transitions) from current states. When a user selects a transition the system leaves the source state which becomes inactive letting the target state to be the next active state in the configuration. SWC also provides the pseudo-states as those found in StateCharts (i.e. shallow history, deep history, end state and initial state). More details about SWC can be found in [18].

In order to exemplify the elements of the SWC notation, Fig. 3 presents the SWC modeling for doing a query over the digital library and logging into the system which correspond to the tasks “Query” and “Log into the system” presented by Fig.1.a and Fig.1.b, respectively.



a) Navigation model for the task “Query”



b) Navigation model for the task “Log into the system”

Fig. 3. SWC modeling to log into the system (a) and query (b).

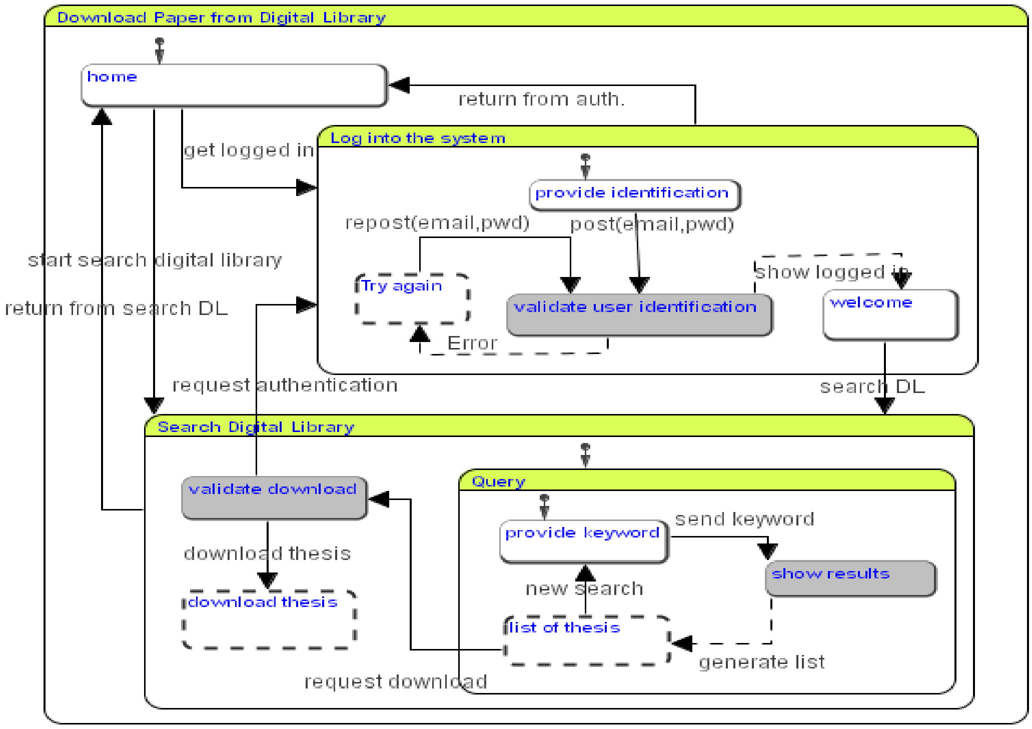


Fig. 4. SWC modeling to log into the system (a) and query (b).

In Fig. 3.b, the states "provide identification" and "validate user identification" correspond to the subtasks in Fig.1.b. The state "validate user identification" is a transient state which does have a visual representation to the user. This state is associate to dynamic state "Try again" which is dynamically generated and presented to the user if the login fails. Otherwise, the transient state present the welcome page represented by the static state "Welcome". We can observe that these states correspond to the subtasks in Fig1.b but they also include new states (i.e. "Try again" and "Welcome") and transitions (i.e. "Error", "Show logged in" and "Repost(mail,pwd)" which are required to describe the behavior of the application in response to user tasks.

The Fig. 4 shows the complete navigation model for download a thesis from the Digital Library including the support for the tasks "Log into the system" and "Query" (the digital library). Similar to the previous examples, this SWC modeling includes transitions that complete the description of the system's behavior. In addition, it features some transitions supporting content-based navigation such as "return from auth.", "get logged in", "return from search DL", "start search digital library" and "Search DL". Notice that these transitions link static states such as "home" and "welcome" which correspond to static documents. When linking up states by the means of transitions we can create all the navigation required by the users whether it concerns content-based navigation or navigation required to follows a specific procedure.

3.4 Prototyping the Web Application

The edition, simulation and prototyping of SWC models are supported by the tool SWCEditor [17] (see Fig 5). After we have verified that the navigation built with SWC holds in our requirements we can start by creating the Web pages that correspond to the SWC model. The Web pages were built using a visual environment independent from SWCEditor since, at the present, it does not integrate a Web page editor. Once we have prototyped each individual Web page for the application, we returned to SWCEditor and we associated each state to a Web page. Each state visible at the user presentation is associated to a Web page which includes the content and the graphical presentation for the objects. The SWCEditor supports the simultaneous simulation of SWC models and the execution of the corresponding Web pages. Fig. 6 provides a view at glance of this process. The navigation modeling for the digital library of AFIHM is presented at left highlighting the current state in the simulation (i.e. the state "home"). At right, Fig 6 presents the corresponding implementation of the home page. We also can observe at left of Fig 6 a dialog window showing a list of transitions going out from the state "home". These transitions are translated to links at the home page. The arrow links indicates this translation.

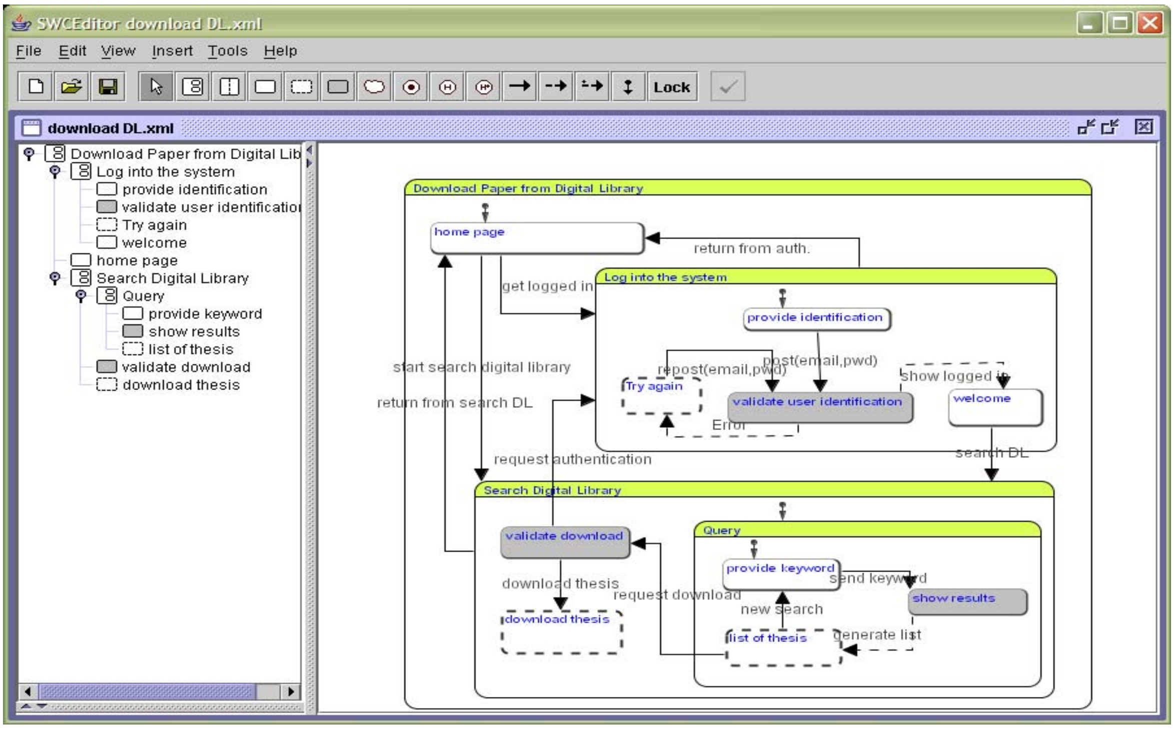


Fig. 5. SWCEditor: edition of the model “Download paper from DL Library”.

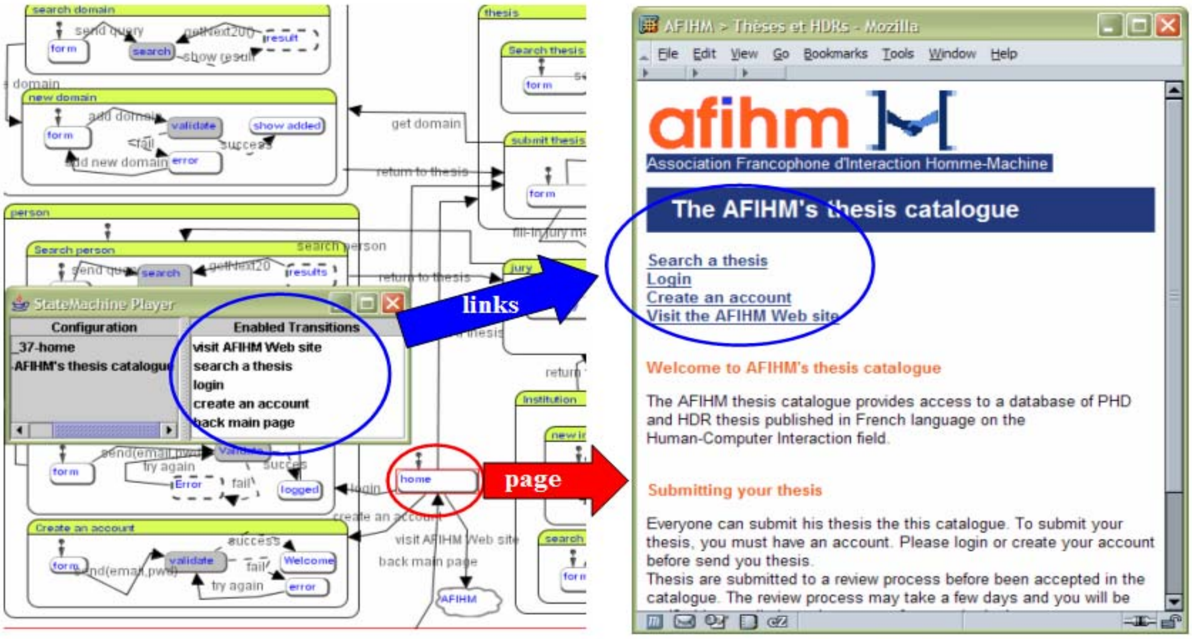


Fig. 6. Prototyping the Digital Library for the theses of the AFIHM.

4 Discussion and Related Work

When designing Web applications, we have to pay attention to the users' tasks and to the users' mental model about the information space in order to help users to navigate efficiently the application. The efficiency of Data-driven approaches is limited to the navigation one can extend from an underlying database. The main problem of such as an approach is that the underlying database does not necessarily (and quite often doesn't) represent the user mental model for the Web application. Even though task modeling is widely considered as helpful activity which let design to analyze the user activity without influence of technological constraints, the actual use of task models for the design of Web applications is underestimated mainly because current approaches for the design do not provide any guidance on how to integrate task models into the design process.

We assume that tasks models are not suitable for representing part of the system because the way users have access to information is part of the system specification not part of the user task [19]. Keep task models independent from system models allows the analysis of user needs for tasks and the transformation of such as models according to the modality and any other implementation constraints. For example, a task model should not inform how many pages a user must visit to accomplish a task because the number of the pages is on the system domain which can adapt the number of pages in the presentation dynamically according to parameters such as the device employed, the preferred modality for user interaction (graphic, sound/voice, etc), and so on. Thus both task models and system models (in the case of the Web navigation models) must be employed synergistically to produce appropriated User-Centered Designs.

As discussed in the first section, most development methods existing for web applications base their conceptual modeling on the objects (or data) and their related methods, functions, or services, and they derive tasks from the traditional CRUD (Create, Read, Update, Delete) pattern: tasks are limited to basic operations on objects and their relationships. When some methods go beyond this simple pattern, the dialog of the resulting user interface is assuming a transactional scheme: all data that are manipulated by the task are supposed to be entered and all methods that are required to accomplish the task should be triggered by the user. These development methods focus on the designer's point of view about the content and the navigation of the web application. Moreover, when the user's perspective is taken into account it is often introduced very informally. When dealing with large web application this informal process reaches its limits and often leads to usability failures.

Inappropriate navigation design of applications as one of the main sources of usability problems related to the navigation [4]. The hypertext interconnections in Web applications can be extremely complex and designers could benefit from tools and guidelines to support and assist them. Tauscher and Greenberg [16] describe some patterns of navigation but their results don't explain which tasks are engaged while these patterns are used. 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 or how a task model can be exploited within the development process of a web application.

Only a few works have been addressed the problem of model user tasks for the Web Design. The WSDM method [5] tackles many usability concerns and user requirements which are quite often neglected by other methods. However WSDM don't provide a way to model and analyze user tasks without having to take into account the system model. SOHDM approach [12] relies upon scenarios to guide all the design activities concerning the development of Web applications but it does not take into account non-functional aspects that are relevant for a user-centered design process. As mentioned before, both content-based and task-based navigation should be supported by navigation modeling methods.

5 Conclusions

This work has presented a Task-Centered Approach for navigation modeling. Our aim is to demonstrate how to describe user requirements by the means of task models and scenarios and how to transform them into navigational paths. For this purposes we have employed the CTT notation [16] to represent user tasks because it enable us to explicitly represent the tasks that are performed by the user, by the application and those which are interactive (i.e. require both system and user intervention). For navigation modeling we have employed the SWC notation [18] because it provides non-ambiguous descriptions for the navigation and it explicitly where user and system act changing the state of the application. Both notation presented are supported by tools which facilitate the edition and simulation of models.

The precise modeling of user tasks provides a deeper understanding about the user needs for the application. The mapping of task models to navigation models allows designers to explore many possible solutions for the implementations.

The role of navigation models is to create appropriate views of information space (by grouping entities from an underlying data model or a document database) and provide navigable relationships in-between according to the users' needs. Since task models are high-level description of the user activity they are not suitable to represent all aspect concerning the navigation over Web applications. For this reason the mapping between task models and navigation models is required to complete the design.

The use of formal description techniques, such as SWC, provides a clear and non ambiguous description of the navigation supporting user tasks by the system. In addition, the appropriate tool support can alleviate the effort of modeling and support rapid prototyping, as shown in section 3.4. Even thought we wish do not discuss in this paper the verification of the SWC model, we can just mention that some tools exist to support the simulation and verification of properties of the model that can be assimilated as usability problems (e.g. deep navigation paths, dead links, and so on).

Acknowledgments

This work is partially supported by the project SPIDER Web (Capes/Cofecub n. 399/02).

References

[1] Balasubramanian, V., Bashian, A. Document management and Web technologies: Alice marries the Mad Hatter, Communications of the ACM 41(7), pp. 107-115 (1998)

[2] Benyon, D. Task Analysis and System Design: the Discipline of Data. Interacting with Computers, 4(1), pp. 102-123 (1992).

[3] Ceri, S., Fraternali, P., Bongio, A. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. WWW9 Conference (2000)

[4] Cockburn, A., Jones, S. "Which way now? Analysing and easing inadequacies in WWW navigation," International Journal of Human Computer Studies pp. 105-129, 1996.

[5] De Troyer, O., Leune, C. J. WSDM: A User Centered Design Method for Web Sites. 7th International World Wide Web Conference (1998).

[6] Fleming, J. Web Navigation: Designing the User Experience. O'Reilly & Associates Press. (1998)

[7] Garzotto, F., Paolini, P., Schwabe, D. HDM—a model-based approach to hypertext application design. ACM Transactions on Information Systems (TOIS) vol. 11, no. 1, pp. 1-26, (1993)

[8] Gómez, C.C. J., Pastor, O. Extending an Object-Oriented Conceptual Modelling Approach to Web Application Design. CAiSE'2000 pp. 79-93, (2000)

[9] Gu, A., Henderson-Sellers, B., Lowe, D. Web Modelling Languages: The Gap Between Requirements and Current Exemplars. 8th Australian World Wide Web Conference (AusWeb'2002) 2002.

[10] Harel, D. StateCharts: A Visual Formalism for Complex Systems. Science of Computer Programming vol. 8, no. 3, pp. 231-274 (1987)

[11] Johnson, P. Human-Computer Interaction: Psychology, Task Analysis and Software Engineering, Mc-Graw Hill, Maidenhead, UK, 1992

[12] Lee, H., Lee, C., Yoo, C. A scenario-based object-oriented hypermedia design methodology. Elsevier Information & Management no. 36, pp. 121-138 (1999)

[13] Murugesan, S., Deshpande, Y. Web Engineering: Managing Diversity and Complexity of Web Application Development. LNCS 2016. Springer-Verlag, Berlin, p. 355 (2001)

[14] Paterno, F., Mancinii, C., Meniconi, S. ConcurTaskTrees: a Diagrammatic Notations for Specifying Task Models, In Proceedings of IFIF T13 Conference INTERACT 97, Sydney, Chapman&Hall. (1997) pp. 362-69

[15] Schwabe, D., Rossi, G. Building Hypermedia Applications as Navigational Views of information Models. 28th Hawaii International Conference on System Sciences pp. 231-240 (1995)

[16] Tauscher, L., Greenberg, S. How people revisit web pages: empirical findings and implications for the design of history systems. Int. J. Human–Computer Studies vol. 47, pp. 97-137 (1997)

[17] Winckler, M., Barboni, E., Farenc, C., Palanque, P. SWCEditor: a Model-Based Tool for Interactive Modelling of Web Navigation. ACM Computer-Aided Design of User Interfaces (CADUI'2004), Funchal, PT. (2004)

[18] Winckler, M., Palanque, P. StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'2003), Funchal, PT. (2003).

[19] Palanque, P.; Bastide, R.; Winckler, M. Automatic Generation of Interactive Systems: Why A Task Model is not Enough. 10th International Conference on Human-Computer Interaction – HCI International'2003, Héraklion, Greece, June 2003.