An Approach to User-Behavior-Aware Web Applications

Stefano Ceri[1], Florian Daniel[1], Vera Demaldé[2], and Federico M. Facca[1]

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano
P.zza Leonardo da Vinci 32, I-20133 Milano, Italy
{ceri,daniel,facca}@elet.polimi.it
[2] Istituto di Tecnologie della Comunicazione, Università della Svizzera Italiana
Via Lambertenghi 10 A, CH-6904 Lugano, Switzerland
demaldev@lu.unisi.ch

Abstract. The *Adaptive Web* is a new research area addressing the personalization of the Web experience for each user. In this paper we propose a new high-level model for the specification of Web applications that take into account the manner users interact with the application for supplying appropriate contents or gathering profile data. We therefore consider entire *processes* (rather than single *properties*) as smallest information units, allowing for automatic restructuring of application components. For this purpose, a high-level *Event-Condition-Action* (ECA) paradigm is proposed, which enables capturing arbitrary (and timed) clicking behaviors. Also, a possible architecture as well as a first prototype implementation are discussed.

1 Introduction

As the Web is a steadily growing environment and users, rather than navigating relatively simple (static) Web sites with structures that evolve only very slowly in time, nowadays users are more and more faced with complex Web applications, dynamically generated contents and highly variable site structures. Continuously, they are confronted with huge amounts of non pertaining contents or changed interaction paths. As a consequence, users may feel uncomfortable when navigating the Web.

Several techniques have been introduced that aim at augmenting the efficiency of navigation and content delivery. Content *personalization* allows for more efficiently tailoring contents to their recipients by taking into account predefined roles or proper user *profiles*. The relevance of information to be presented is derived from both user profile data and explicitly stated preferences.

Context-aware or *adaptive* Web applications [1, 2] go one step further and aim at personalizing delivered contents or layout and presentation properties not only with respect to the identity of users, but also by taking into account the context of the interaction involving users and applications. Besides proper user profiles, the term *context* usually refers to *environmental* (e.g temperature, position) or *technological* (e.g. device, communication channel) factors.

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 417–428, 2005.

Along a somewhat orthogonal dimension, *workflow-driven* Web applications address the problem of showing the right information at the right time by explicitly modeling the hidden (business) process structure underlying determined usage scenarios, especially within business-oriented domains. Whereas several commercial workflow management systems [3] exist that allow specifying processes by means of proper visual modeling tools and also support Web-based user interfaces, Brambilla et al. [4] propose a hybrid solution that weaves the necessary process logic into high-level, conceptual WebML site models [5].

Eventually, usability studies and Web log analysis efforts [6] try to examine the usability and thus ergonomics problem by means of an ex-post approach with the aim of deriving structural weaknesses, checking assumptions made about expected user navigations and mine unforeseen navigation behaviors for already deployed Web applications. Final goal of these approaches is the incremental enhancement of the application under investigation in order to meet the newly identified requirements.

We believe that a new approach and an open paradigm that combines adaptive and process-centric perspectives can open new ways for both (coase-grained) application adaptation and (online) usability analysis. In this paper we propose a model and a methodology to easily design *behavior-aware* Web applications that allow performing actions in response to the user's fulfillment of predefined navigation patterns. Our proposal is based on the conceptual framework provided by WebML, but we also propose a new formalism, WBM (*Web Behavior Model*), a simple and intuitive model for describing navigation goals. The two models are combined to form a high-level *Event-Condition-Action* paradigm providing the necessary expressive power for capturing the way users interact with applications. Despite the adoption of WebML for hypertext design, the proposed solution is of general validity and can thus be applied to arbitrary Web applications.

The paper is organized as follows: Section 2 introduces WebML and WBM as conceptual background; in Section 3 we define the ECA paradigm by combining WebML and WBM. In Section 4 we discuss a possible SW architecture, Section 5 illustrates an applicative example, and Section 6 outlines experiences gained so far. In Section 7 we discuss related research work and, finally, in Section 8 we address future research efforts.

2 Background Models

2.1 WebML: An Overview

WebML (Web Modeling Language) is a conceptual model and development methodology for Web application design [5], accompanied with a CASE tool [5, 7] and an automatic code generation mechanism. WebML offers a set of visual primitives for defining conceptual schemas that represent the organization of contents into hypertext interfaces. Visual primitives are also provided with an XML-based textual representation, which allows specifying additional properties, not conveniently expressible in the visual notation. For specifying the data

structure, upon which hypertexts are defined, WebML adopts the well known Entity-Relationship model (ER).

WebML allows designers to describe hypertextual views over the application data, called *site views*. Site views describe browsable interfaces, which can be restricted to particular classes of users. Multiple site views can be defined for the same application. Site views are organized into hypertext modules, called *areas*. Areas and site views contain *pages* that are composed of containers of elementary pieces of content, called *content units*. Content units are linked to entities of the underlying data schema, holding the actual content to be displayed, and restricted by means of proper selector conditions. There are several predefined units (such as *data*, *index*, *multidata* or *entry* units) that express different ways of publishing or gathering data within hypertext interfaces; also, proprietary units can be defined. Arbitrary business actions can be modeled by means of so-called *operation units* and performed through navigating a relative input link. WebML incorporates some predefined operations for creating, modifying and deleting data instances and relationships among entities, and allows developers to extend this set with own operations.

Finally, pages or units can be connected by means of directed arcs, the *links*. The aim of links is twofold: permitting users to navigate contents (possibly displaying a new page, if the destination unit is placed in a different page), and passing parameters from a source unit to a destination unit for providing possible selector variables with respective values. For further details on WebML, the reader is referred to [5].

Recently, WebML has been extended to support the design of context-aware or adaptive Web applications [1]. Adaptive pages are continuously refreshed and, according to possible changes within the model of the application's context, content or layout adaptations as well as automatic navigation actions can be performed before rendering the HTML response. WebML operation chains are associated to adaptive pages and express proper actions to be carried out.

2.2 WBM: An Overview

The *Web Behavior Model* (WBM) is a timed state-transition automata for representing classes of user behaviors on the Web. Graphically, WBM models are expressed by labeled graphs, allowing for an easily comprehensible syntax; cf. Figure 1.

A *state* represents the user's inspection of a specific portion of Web hypertext (i.e., a page or a collection of pages), which is loaded on his browser, or the user's activation of a specific Web operation, such as "buy" on an e-commerce site, or "download" of a given file. A *transition* represents the navigation from one state to another. State labels are mandatory and correspond to names of pages or page collections or operations; transition labels are optional and express constraints enabling or disabling transitions, in terms of both used hypertext links and time. Each WBM specification, called *script*, has at least an initial state, indicated by an incoming unlabeled arc, and at least one accepting state, highlighted by dou-

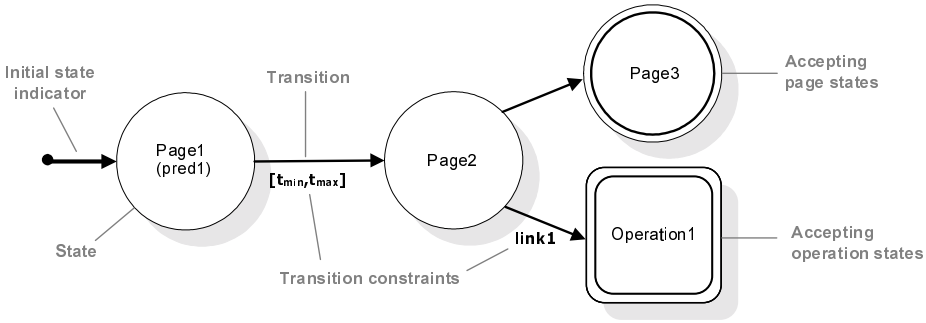

Fig. 1. Example of WBM script with state, link, time constraints and multiple exiting transitions from one state. Basic WBM primitives are named: page states are expressed by circles, operation states by rectangles

ble border lines; Figure 1 provides an overview of WBM primitives. Transitions can be constrained by state, link, and time constraints as follows.

- **State constraints.** Entering a state may be subject to the evaluation of a state constraint, expressing a predicate over properties of the pages being accessed or operation being fired. Such predicate may refer to contents displayed within pages or to operation parameters. The state is accessed iff the predicate evaluation yields to true.
- **Link constraints.** Each transition may be labeled with the name of a link entering the page or enabling the operation. The state is accessed iff the specified link is navigated.
- **Time constraints.** Each transition from a source to a target state may be labeled with a pair $[t_{min}, t_{max}]$ expressing a time interval within which the transition can occur. Either t_{min} or t_{max} may be missing, indicating open interval boundaries. If a transition does not fire within t_{max} time units, it can no longer occur; on the other hand, navigation actions that occur before t_{min} are lost. The use of suitable time constraints may thus cause the invalidation of running scripts.

One important aspect of WBM models is, that not all navigation alternatives must be covered. As the aim of WBM is to capture a concise set of user interactions, describing particular navigation goals and respective "milestones", only a subset of all possible navigation alternatives is relevant. E-commerce Web sites, for example, make heavy use of so-called *access-pages* that only serve the purpose of providing users with browsable categories for retrieving the actual products offered. Furthermore, Web sites usually provide several different access paths toward their core contents. Therefore, by concentrating only on those interactions that really express navigation goals, WBM allows both abstracting from unnecessary details and defining small and easily comprehensible specifications. Only performing specified target interactions – in the modeled order – may thus cause WBM state changes.

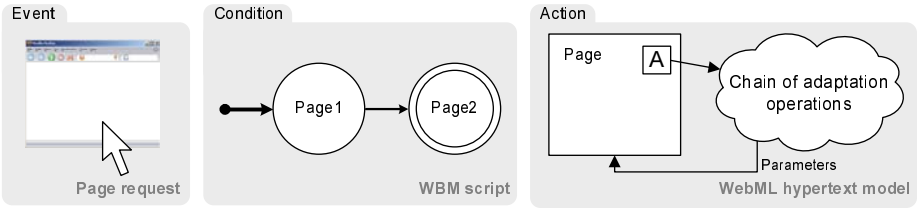

Fig. 2. High-level ECA rule components

Figure 1 shows an example WBM script. Entering the state denoted by *Page1* is constrained by *pred1*, which must evaluate to true. The transition from the first state to the second state must occur within t_{min} and t_{max} time units from the moment the script has been initiated, otherwise the script fails. The script in Figure 1 also presents two exiting transitions from state *Page2*. States labeled *Operation1* and *Page3* are "competing", as a browsing activity in *Page2* may lead to either *Operation1* or *Page3*. We constrain WBM scripts to have only one active state at time, only transitions may cause changes to it. Therefore, either a user browses from *Page2* to *Page3* (the transition *Page2* to *Page3* is triggered) and the script reaches the accepting state denoted by *Page3*, or the transition to accepting state *Operation1* occurs if *Operation1* is performed by using *link1*.

Despite the use of WebML for specifying example hypertext structures, WBM as adopted in this paper can be used to describe navigation behaviors on top of arbitrarily developed hypertexts. For further details on WBM and its propositional logic, the reader is referred to [8].

3 ECA Rule Model

To build the ECA rules that finally make Web applications aware of predefined user behaviors, we now combine WBM scripts and WebML adaptation mechanisms. Commonly, the ECA paradigm describes a general syntax (`on` *event* `if` *condition* `do` *action*) where the event part specifies *when* the rule should be triggered, the condition part assesses whether given *constraints* are satisfied, and the action part states the *actions* to be automatically performed if the condition holds.

In our view, the event consists in a *page* or *operation request*, the condition is a set of requirements on the user navigations (expressed as a *WBM script*), and the action part specifies some adaptivity actions to be forced in the Web application and expressed as WebML *operation chain*. Events are generated only for explicitly labeled pages (`A`-label) denoting the *scope* of the rule, and proper rule priorities resolve possible conflicts among concurrently activated rules. Figure 2 graphically represents such ECA rules.

Consider for instance the rule of Figure 2. The rule reacts to a user's visit to *Page1* followed by a visit to *Page2*. Thus, the expressed condition only holds when the script gets to the accepting state *Page2*. Once the accepting state is

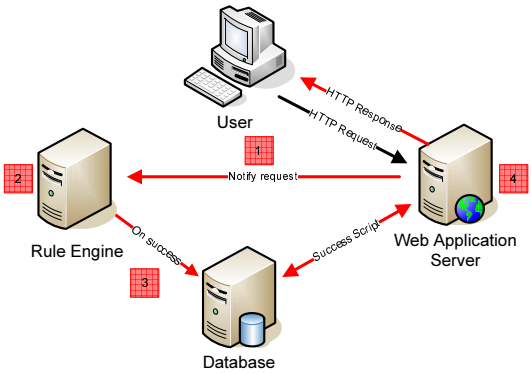

Fig. 3. A schema representing the architecture of the ReActive Web System

reached, the actions (expressed as cloud in Figure 2) are executed and, after a re-computation of page parameters, possible adaptations may be performed.

4 The ReActive Web System Architecture

Our so-called ReActive Web framework requires an extension of standard Web architectures: a new server, called *Rule Engine*, is introduced as illustrated in Figure 3. It collects and evaluates HTTP requests in order to track the user's navigational behavior, and hosts a repository of WBM scripts, which can be executed on behalf of individual users.

The behavior of the Rule Engine is described by the following steps (cf. Figure 3).

1. URL requests as generated by user clicks are notified to the Rule Engine.
2. A request can cause either the instantiation of a new script, or a state change of a given running script, or nothing.
3. When a WBM script reaches an accepting state for a certain user, the Rule Engine changes a record in the shared database, storing the information about the completed script and the user's session. Also, variables used by the WBM script are stored in the database.
4. Finally, if the request refers to a page contained within the rule's scope, the application interprets the modified data record as request for activating the adaptation chain associated to that page. Accordingly, it executes the operation chain, possibly generating a modified hypertext.

While the above steps represent the core of the Rule Engine's behavior, several variants are possible regarding the interaction between the three servers. For example, the Rule Engine server can act as stand-alone system for usability analysis or validation of given Web applications. The use of a distributed architecture offers some significant advantages:

- Since script handling is assigned to a dedicated server, which is neither the database nor the Web server, the overall application's performance is not affected by the time required for rule processing[1].
- The Rule Engine is not bound to the technology used to develop the Web application, and a single Rule Engine can handle rules for more than a single Web application.
- The Web application remains operational, even in case of Rule Engine slowdowns or crashes.
- The Rule Engine can be recovered independently from the rest of the Web application.

Synchronous as well as *asynchronous* rule execution models can be achieved with the presented architecture. In the synchronous case, if a rule is successfully triggered, the action part of the rule is executed immediately at the first page request. To avoid possible performance slowdowns (due to time spent for script evaluation), the asynchronous configuration defers rule evaluation to the next (automatic) page refresh. This allows for parallel tasks and short response times. The strong decoupling of application server and Rule Engine allows for independent resource management and parallel and scalable configurations[2].

5 Case Study: An E-Learning Web Application

In this section we introduce a case study to explore some aspects of the potential of our approach. In particular, we chose the e-learning domain due to the large possibility of personalization and adaptation possibilities it offers. A sketch of the e-learning Web application model – without the adaptation layer – is depicted in Figure 4. When a user logs in to the application, he is forwarded to the *Home* page, where *User Data* and *Suggested Courses* – only if there is any suggestion for the current user in the database – are displayed. From the *Home* page the user can ask for the *Courses* page. When requesting that page, if there is no *ExpertiseLevel* (a value corresponding to the user's current level of knowledge) available for the current user, he is redirected to the *Test* page. In this case, a multiple choice test for the lowest level of knowledge is proposed to the user. When he submits the filled test to the Web application, his new *ExpertiseLevel* is computed and he is redirected to the *Courses* page, where suitable concepts for its level are presented. From here, the user can browse new contents (*Course* page) or navigate to the *Test* page and perform a new test to verify if his level is increased after having studied new contents.

In the following we introduce two examples that add an adaptation layer to the presented Web application.

[1] This result requires, in addition, to instrument the Rule Engine as asynchronous process, as will be discussed later

[2] Further details on the implementation of the Rule Engine can be found at `http://dblambs.elet.polimi.it`

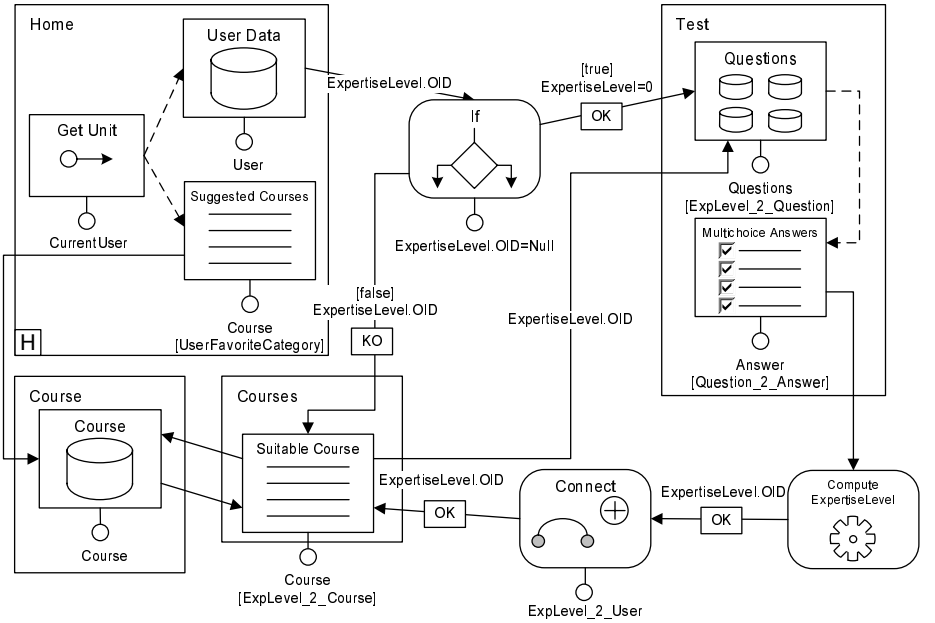

Fig. 4. The WebML model of the proposed educational Web site

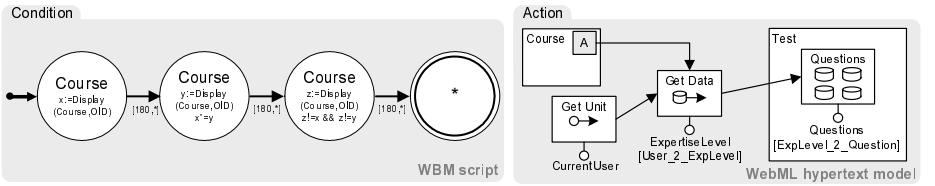

Fig. 5. An ECA adaptive rule to trigger evaluation of a student's knowledge level

Example 1. Evolving the Level of User Expertise. Figure 5 models an ECA rule to redirect the user to the *Test* page for the next experience level after having visited 3 courses; i.e., 3 different instances of *Course* pages, spending at least 3 minutes over each page. The WebML operation chain for adaptation is actually performed when the user asks again for a *Course* page. The * in the final state of the WBM script specifies the acceptance of any arbitrary page. The WebML model in figure serves the purpose of providing the user with new questions and answers allowing him to assess progress.

Example 2. User Profiling. Suppose we want to personalize the application according to the user's preferences traceable from his navigational choices (cf. Figure 6). We detect that a user is interested in a certain category of courses when he navigates at least three different *Course* pages presenting three courses belonging to the same category. In response to this behavior, the WebML chain

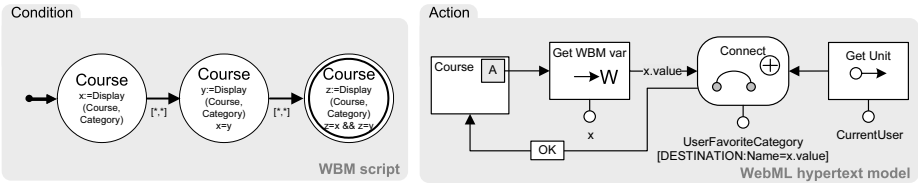

Fig. 6. An ECA adaptive rule to profile users according to their navigational behavior

stores the preference reported by the user – captured by the `Get WBM Variable Unit` – in the database. Now, courses belonging to the same category are presented to the user by means of the *Suggested Courses* unit in the *Home* page.

6 Experiments

A first prototype of the presented architecture has been developed and tested by implementing the reactive Web application of the previous case study (see Section 5). So far, we have fully implemented only link and time constraints provided by WBM, while only few of the mechanisms required by WBM state constraints have been realized. At this step, we are using a Web service for the interaction between Web application and Rule Engine. First feedback from experiments are quite positive: experiments proved that the whole mechanism is feasible and that the use of the asynchronous execution model effectively avoids Rule Engine response times to impact on user navigation. Besides positive initial considerations, experiments revealed a problem of performance in the architecture. We observed an excessive lag between the start of a notification of a page request and the final computation of the new state by the Rule Engine (around 2.5 seconds to manage 100 user requests). Further studies proved that the bottleneck of the system was not the Rule Engine (a stand-alone version of the Rule Engine can process the same 100 requests in less than 60 milliseconds). The actual bottleneck was found to be the time needed to generate the SOAP request by the client. We will fix this problem in the upcoming prototype and test the impact on performance caused by the introduction of complex state constraints[3].

7 Related Works

The ECA rule paradigm was first implemented in active database systems in the early nineties [9, 10] to improve the robustness and maintainability of database applications. Recently, they have been also exploited in other contexts such as

[3] Experiments were realized using an AMD AthlonXP 1800+, 512MB of RAM and with Tomcat as web server. WBM scripts used were more complex than the ones described in this paper

XML [11], to incorporate reactive functionality in XML documents, and the Semantic Web [12], to allow reactive behaviors in ontology evolutions. Our research explicitly adds ECA-rules to web application systems to allow adaptation based on user behavior.

A number of paradigms to model the user's interaction with the Web have been proposed [13, 14]. Indeed, the proposed approaches model the user interaction from the Web application's point of view: they have been designed to describe the navigational model of Web applications and not to model the user interacting with the application. Nevertheless, they can be adapted to model the user behavior disregarding the navigational design of the Web application. The main advantage of these models is their strong formal definition, as they are based on well known formal models like Petri Nets [13] or UML StateCharts [14].

WBM, on the other hand, is a general purpose model to describe at high level the user's interaction with Web applications, focusing only on navigational alternatives able to capture navigation goals. Besides that, WBM has an easy visual paradigm that allows designers to specify arbitrary user's behavior models.

A variety of design models have been proposed for Adaptive Hypermedia [15–17]. While most of these methods differ in approach, all methods aim to provide mechanisms for describing Adaptive Hypermedia (see [18, 19] for a survey). Most of them do not use a full ECA paradigm and rather deal with a CA paradigm [17]. Others [15] are not conscious to use an ECA paradigm and hence do not refer directly to it or do not propose a formal model, based on such a well-known paradigm. Some of them focus only on the adaptation, disregarding an effective description of the user's behavior that should trigger the adaptation [20]. A comprehensive overview of commercially available solutions is presented in [21]. The author points out that commercial user modeling solutions are very behavior-oriented: observed user actions or action patterns often lead directly to adaptations without an explicit representation of the user characteristics.

AHAM [15], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on Dexter [22], an early model for hypertext, and uses maps of concepts. The model presents many valid ideas (e.g. the 3-layer model capturing all the adaptive semantics) but suffers for the use of an old-fashioned model such as Dexter and is more suited for e-learning domain. The model introduced in [20] extends WSDM [23], a Web design method, with an Adaptation Specification Language that allows specifying the adaptive behavior. In particular, the extension allows specifying deep adaptation in the Web application model, but lacks expressive power as regards the specification of the user's behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided. In [17] the authors propose a Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way an adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows expressing conditions on the user's behavior. The proposed visual notation lacks of immediacy and suffers the use of a visual paradigm born outside the Web area. Likewise, [2]

proposes an interesting framework for context-aware adaptivity based on ECA rules, but it fails in proposing a real design model. It is not even clear if the original idea has been really used in experimental applications and with which results.

Compared to the cited researches, our model allows for an easy specification of user-behavior-driven, adaptive Web applications, and we heavily exploit expressive power derived by ECA rules. Furthermore, we supported our ideas by implementing and testing the proposed architecture underlying the model, allowing for automatic code generation from the Web application design model.

8 Conclusion and Future Work

In this paper we proposed a general purpose model for building behavior-aware Web applications. Our proposal is based upon WebML and WBM, and combines these two models into a visual ECA paradigm that opens the road to the implementation of high-level CASE tools for designing advanced Web sites. In this context, we are currently investigating the use of well-known modeling primitives, such as UML statecharts, for expressing WBM and its notation, as they are already supported by proper CASE tools.

Within our future work, we will develop proper policies for dealing with priorities and conflicts. Currently, we adopt the simple policy of always choosing the rule at highest priority, but this can be improved. Furthermore, we did not consider the problem of rule termination yet, which might arise when rules trigger each other. Also, dynamic activation and deactivation of rules and of rule groups will be considered.

A first prototype of the reactive Web environment has been implemented. It demonstrates the applicability and power of the approach, as it supports rules of arbitrary complexity and therefore can build arbitrary reactive applications. The implementation of a second generation prototype is ongoing, with optimized rule management and offering full graphic user interfaces to designers.

References

1. Ceri, S., Daniel, F., Matera, M.: Extending webml for modeling multi-channel context-aware web applications. In: Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03), Rome, Italy, December 12 -13, 2003, IEEE Press (2003) 225–233
2. Finkelstein, A.C., Savigni, A., Kappel, G., Retschitzegger, W., Kimmerstorfer, E., Schwinger, W., Hofer, T., Pröll, B., Feichtner, C.: Ubiquitous web application development - a framework for understanding. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI). (2002) 431–438
3. IBM: MQ-Series Workflow. `http://www-306.ibm.com/software/integration/wmqwf/`.
4. Brambilla, M., Ceri, S., Comai, S., Fraternali, P., Manolescu, I.: Specification and design of workflow-driven hypertexts. Journal of Web Engineering (JWE) **1** (2003) 163–182

5. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kauffmann (2002)
6. Facca, F.M., Lanzi, P.L.: Recent developments in web usage mining research. In: Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2003, Prague, Czech Republic, September 3-5,2003. Volume 2737 of Lecture Notes in Computer Science., Springer (2003) 140–150
7. WebModels srl: WebRatio. `http://www.webratio.com`.
8. Armani, J., Ceri, S., Demaldè, V.: Modeling of user's behaviors in web applications: a model and a case study. Technical report, Institute of Communication Technologies, Università della Svizzera italiana (2004)
9. Widom, J.: The starburst active database rule system. IEEE Trans. Knowl. Data Eng. **8** (1996) 583–595
10. Ceri, S., Fraternali, P., Paraboschi, S., Tanca, L.: Active rule management in Chimera. In: Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, San Francisco, California (1996)
11. Bonifati, A., Ceri, S., Paraboschi, S.: Active rules for xml: A new paradigm for e-services. The VLDB Journal **10** (2001) 39–47
12. Papamarkos, G., Poulovassilis, A., Wood, P.T.: Event-condition-action rule languages for the semantic web. In: Proceedings of SWDB'03, Berlin, Germany, September 7-8, 2003. (2003) 309–327
13. Stotts, P.D., Furuta, R.: Petri-net-based hypertext: Document structure with browsing semantics. ACM Transactions on Information Systems **7** (1989) 3–29
14. Turine, M.A.S., de Oliveira, M.C.F., Masiero, P.C.: A navigation-oriented hypertext model based on statecharts. In: HYPERTEXT '97: Proceedings of the eighth ACM conference on Hypertext, New York, NY, USA, ACM Press (1997) 102–111
15. Bra, P.D., Houben, G.J., Wu, H.: Aham: a dexter-based reference model for adaptive hypermedia. In: Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots. (1999) 147–156
16. Casteleyn, S., Troyer, O.D., Brockmans, S.: Design time support for adaptive behavior in web sites. In: Proceedings of the 2003 ACM symposium on Applied computing. (2003) 1222–1228
17. Koch, N., Wirsing, M.: The munich reference model for adaptive hypermedia applications. In: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer-Verlag (2002) 213–222
18. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction **11** (2001) 87–110
19. Cannataro, M., Pugliese, A.: A survey of architectures for adaptive hypermedia. In Levene, M., Poulovassilis, A., eds.: Web Dynamics. Springer-Verlag, Berlin (2004) 357–386
20. Casteleyn, S., Troyer, O.D., Brockmans, S.: Design time support for adaptive behavior in web sites. In: Proceedings of the 2003 ACM symposium on Applied computing. (2003) 1222–1228
21. Kobsa, A.: Generic user modeling systems. User Model. User-Adapt. Interact. **11** (2001) 49–63
22. Halasz, F., Schwartz, M.: The Dexter hypertext reference model. Communications of the ACM **37** (1994) 30–39
23. Troyer, O.D. In: Audience-driven Web Design. Idea Group (2001) 442–462