Visual Modeling of ReActive Web Applications

Federico Michele Facca and Florian Daniel

Dipartimento di Elettronica e Informazione, Politecnico di Milano
P.zza Leonardo da Vinci 32, I-20133 Milano, Italy
{facca,daniel}@elet.polimi.it

Abstract. In our previous research we have proposed a new high-level model for the specification of Web applications that takes into account the way in which users interact with the application in order to actively react and supply appropriate contents or gather profile data. In this context, we have realized a proper execution framework to develop ReActive Web applications specified by means of the novel modeling paradigm. In this paper we discuss an e-learning case study implemented using our framework and introduce the visual design tools we have extended and developed in order to support the development of ReActive Web applications.

1 Introduction

Reactivity on the Web is becoming a hot topic, and aims at addressing new issues within emerging e-business and e-learning Web applications, where retrieval and update of data plays an essential role. In this context, monitoring the behaviors of users may enable Web applications to react to such behaviors and to improve the users navigation comfort and interactivity with the application. This paper is based on our previous research on ReActive Web applications [1, 2]; the result of this research is a visual Event-Condition-Action (ECA) paradigm to describe reactivity triggered by a user's interactions with the Web application.

The ECA rule paradigm was first implemented in active database systems in the early nineties [3] to improve the robustness and maintainability of database applications. Recently, it has also been exploited in other contexts, such as XML [4] to incorporate reactive functionality in XML documents, Semantic Web [5] to allow reactive behaviors in ontology evolution, Web applications [6] to realize reactive behaviors involving distributed applications on the Web.

In our framework it is possible to specify arbitrary composite and timed behaviors depending on a user's navigations and to react by adapting the Web application and the application's data. A composite behavior is a behavior including different user interactions with the Web application, i.e., user's inspections of different portions of the Web application and data instances; a timed behavior is a behavior where the time gaps between the the users' actions are specified by time constrain. In this paper we highlight the potentialities of our framework by presenting an e-learning case study and by introducing the visual

design tools we used to automatically generate the application presented in the case study.

This paper is organized as follows: Section 2 introduces the two background models that we adopt for modeling ReActive Web applications: WebML and WBM. Section 3 combines WebML and WBM for defining proper ECA rules. Section 4 illustrates an applicative example, and Section 5 introduces some details on the framework used to develop ReActive Web applications. Finally, in Section 6 we address future research efforts and draw some conclusions.

2 Background Models

Our ReActive framework is based on two models, WebML [7] and WBM [2], which are properly combined to obtain a visual paradigm for ECA rules, enabling the specification of reactivity of user behaviors.

2.1 Web Modeling Language

The *Web Modeling Language*, WebML, is a conceptual model for the design of Web applications [7], supported by a proper CASE tool [8]. The WebML method fosters a strong separation of concerns, by separating the information content from its composition into hypertext, navigation, and presentation, which can be defined and evolved independently. The modeling language offers a set of visual primitives for defining structural schemas that represent the organization and navigation of hypertext interfaces on top of the application data, while for specifying the organization of data the well known Entity-Relationship model is adopted. Also, primitives for specifying data manipulation operations for updating the site content or interacting with arbitrary external services are provided.

For further details on WebML, the reader is referred to [7].

2.2 Web Behavior Model

The *Web Behavior Model*, WBM, is a timed state-transition automaton for representing classes of user behaviors on the Web. Graphically, WBM models are expressed as labeled graphs, allowing for an easy to understand syntax (cf. Figure 1). A *state* represents the user's inspection of a specific portion of hypertext (i.e., a page or a collection of pages). State labels are mandatory and correspond to names of pages or page collections. A *transition* represents a navigation from one such portion to another and, thus, the evolving from one state to another. 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 double border lines. Initial states cannot also be accepting states. 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 must occur in order to cause a state transition.

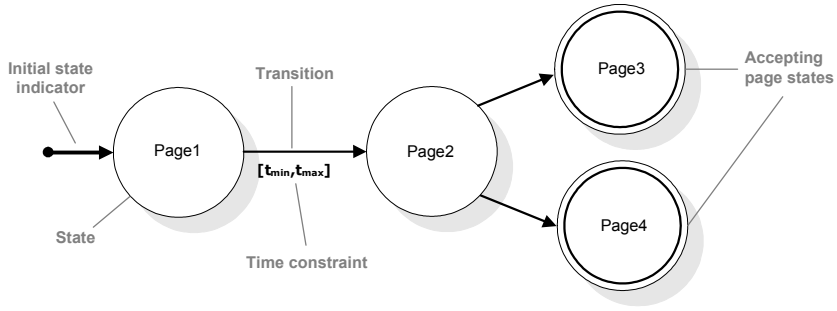

Fig. 1. Example of WBM script with state, link, and time constraints and multiple exiting transitions from one state.

Finally, the expressive power of WBM has been augmented to better describe WebML-based applications: *state constraints* – referring to data contained within a page – and *link constraints* – referring to a particular incoming or outgoing link – have been introduced. For further details on the WBM model and its tailoring to WebML, the reader is referred to [2].

3 Reacting to User Behaviors

In order to be able to react to observed behaviors and to adapt the running application to novel requirements, we combined WebML and WBM. In our framework possible reactions comprise: (i) adaptivity of contents published by specific pages; (ii) automatic execution of navigation actions toward other pages; (iii) automatic execution of operations or services; (iv) adaptivity of the overall hypertext structure.

Although independent from one another, expressing adaptation as a combination of WBM scripts and WebML adaptivity constructs intrinsically leads to a high-level ECA paradigm for specifying adaptivity. Commonly, ECA rules respect the 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. When specifying behavior-aware Web applications, the event consists of a *page* or *operation request*, the condition checks the state of the WBM scripts associated with the current page, and the action part specifies some actions to be forced on the Web application which are expressed as a WebML *operations chain*, i.e., a sequence of WebML operation units. The condition is satisfied and, thus, actions are performed, only when the page's scripts reaches an accepting state. Actions are executed only when pages associated with the respective rules are accessed. ReActive pages are labeled in the WebML hypertext model with `A`, standing for "Active". To avoid multiple rule activation conflicts a priority can be associated with each rule, thus if the

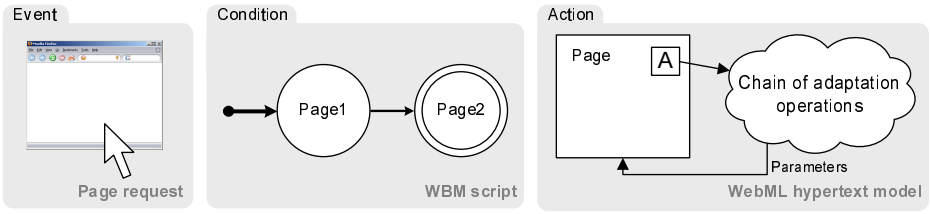

Fig. 2. High-level ECA rule components.

condition holds for two ore more rules, only the one with the highest priority is fired.

Figure 2 graphically summarizes the outlined rule construct: The rule reacts to a user's visit to *Page1* followed by a visit to *Page2* at some stage after his/her visit to *Page1*. The expressed rule condition only holds when the script reaches the accepting state *Page2*. In this case the operations associated to that page (abstracted as the cloud in Figure 2) are executed and possible reactions may be performed.

For further details on WebML operations chain and adaptivity in WebML, the reader is referred to [9].

4 An E-learning Case Study

This section provides an e-learning reference scenario modelled in WebML. Later we will enrich the scenario with proper reactivity constructs by means of visual ECA rules as shown in Section 3. This way of presenting the Web application fully reflects our development method.

The non-adaptive application allows users to browse courses according to their personal expertise level on the topic of the course and to test the acquired knowledge by answering related questions, thus possibly enhancing their knowledge level on a topic. Figure 3 depicts the E/R schema underlying the e-learning

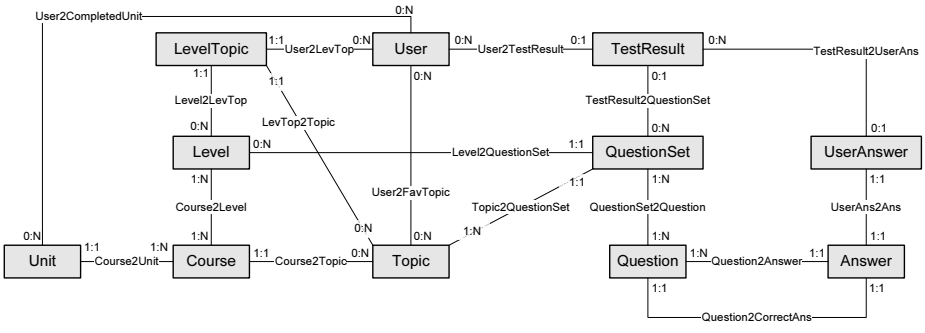

Fig. 3. Entity-Relationship schema for the ReActive e-learning application.

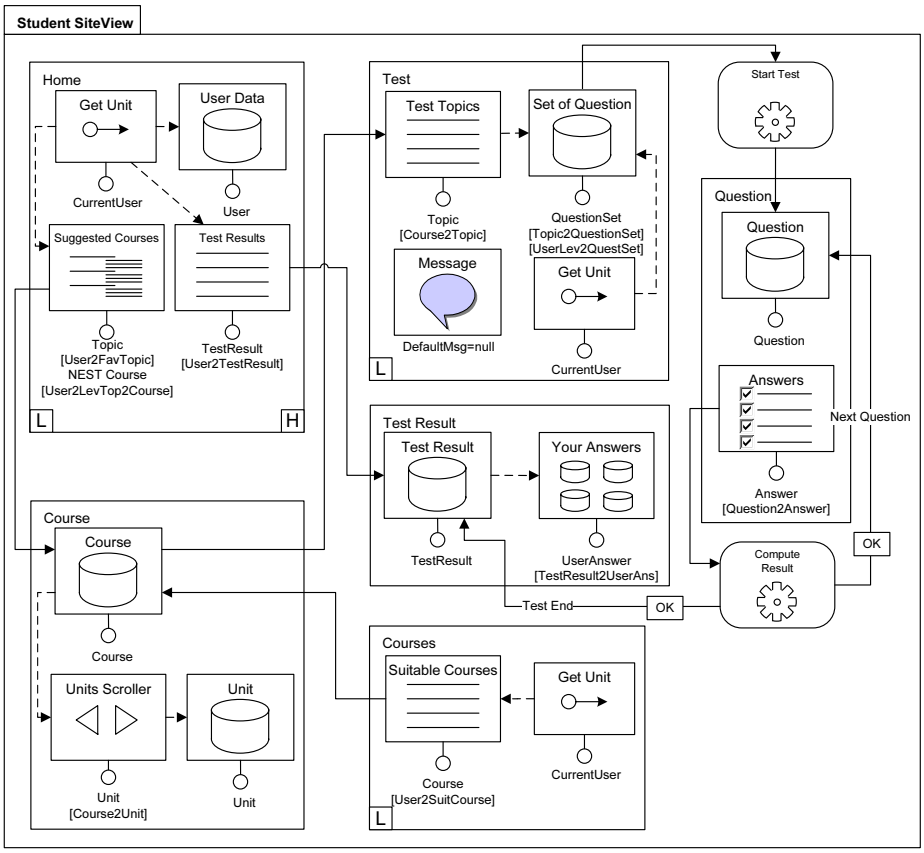

Fig. 4. The WebML model of the proposed educational Web site.

application: each user has a set of favorite topics and an associated level of expertise for each topic. Each course is related to one or more levels of expertise and to one topic. For each level of expertise and each topic there is a set of questions; each question is associated to a set of possible answers and to one correct answer. Results achieved by users and their answers to each question are stored. To simplify the diagram, derived relationship are not shown.

A simplified WebML model for the *Student* siteview of the e-learning application is depicted in Figure 4. The *Home* page contains *User Data*, a list of *Suggested Courses*, grouped by topic and selected according to the user's knowledge level on the topic[1] and a list of the *Test Results* scored. The *Get Unit* allows accessing the user's identifier, while the selector conditions below the units allow binding a unit to a data entity and personalizing the displayed items by applying filter conditions. From the *Home* page the user can ask for the *Courses* page or

[1] When a new student registers for the first time to the Web application, his/her level of knowledge is assumed to be 0 for each topic.

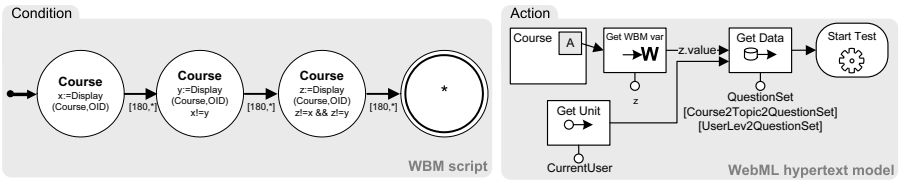

Fig. 5. An ECA rule to trigger the evaluation of a student's knowledge level. The event part (user click) is omitted for simplicity.

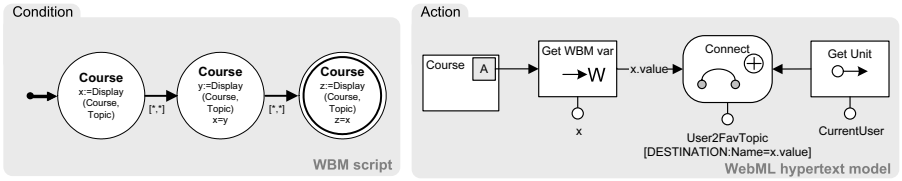

Fig. 6. An ECA rule to profile user preferences.

the *Test Result* page. L-labeled pages – *Home*, *Test* and *Courses* – are landmark pages and can be accessed from any page within the hypertext. The *Test* page presents a list of *Topic* and for each of them a *Set of Question* is selected according to the individual knowledge on the selected topic. Once the user has selected a topic on which he/she wants to test his/her knowledge, he/she can start a test. Hence a *Question* is presented with the relative set of possible *Answers*. Submitting an answer, its correctness is evaluated and a score is associated to the the user by the *Compute Result* operation unit. Then, this unit sends the user to the next question, or, finally, computes the new expertise level of the user on the topic he/she wanted to test and redirects the user to the *Test Result* page. In this page, the *Test Result* scored is reported to the user together with the set of answers he/she selected during the test. In the *Courses* page, *Suitable Courses* according to the user's knowledge level are presented. From here, the user can browse a (*Course* page) where each *Course* is organized into a set of smaller contents that can be scrolled.

In the sequel we describe some examples that add a ReActive layer to the Web application.

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 different Course page. The $*$ in the final state of the WBM script specifies the acceptance of any arbitrary page. The WebML operation chain for adaptation is thus performed when the user asks again for a *Course* page. When the chain is activated, the appropriate question set is retrieved by the `Get Data` unit using parameters passed by the `Get` unit and the `Get WBM Variable` unit, hence the test starts.

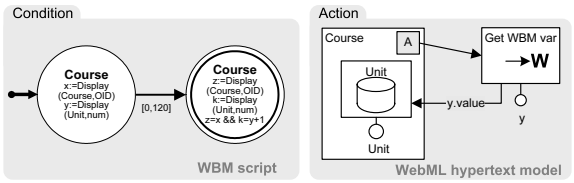

Fig. 7. An ECA rule to oblige student to spend enough time on a course unit.

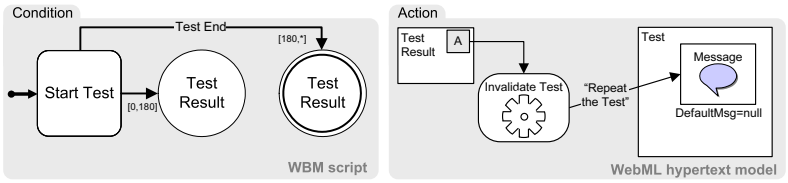

Fig. 8. An ECA rule to invalidate too long tests.

Example 2. Student Profiling. Suppose we want to personalize the application according to the user's preferences traceable from his/her navigational choices (cf. Figure 6). The script detects that a user is interested in a certain topic, whenever he/she navigates at least three different *Course* pages presenting three courses belonging to the same topic. The identified topic is stored within the variable x. In response to this behavior, the WebML operation chain stores the derived preference: the value of the variable x is retrieved by the `Get WBM Variable` unit and the identified topic is associated to the current user. Now, when the user enters the *Home* page, courses belonging to the same topic are automatically presented by means of the *Suggested Courses* unit (cf. Figure 4).

Example 3. Imposing time of page browsing. Figure 7 depicts an ECA rule to oblige students to spend enough time on a course unit before accessing the next one: the WBM script tracks users that access a course *Unit* within the *Course* page and then ask for the next course *Unit* of the same course in less than 2 minutes. In such case, we suppose the user has not carefully read the unit, and hence the action part of the rule forces him to stay on the same course *Unit*.

Example 4. Invalidating a Test. Suppose it is not enough for us that a student successfully passes a test to improve his/her level, but we also want him/her to pass it within a certain amount of time (cf. Figure 8). The WBM script reaches a success state only if the user starts a test and reaches the *Test Result* page in more than 180 seconds, in such case he/she is redirected to the *Test* page where the *Message* unit asks him/her to repeat the test.

Further ECA rules that can be applied to the e-learning Web application may include: dynamical increase/decrease of the difficulty level of the question

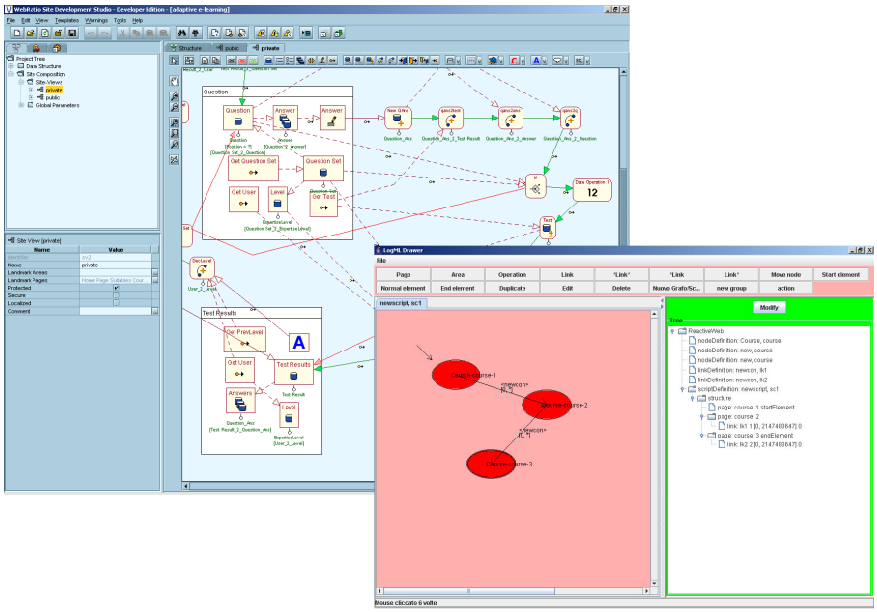

Fig. 9. The WebRatio modeling tool, extended with the new reactivity-supporting units and the WBM CASE tool.

according to the answer time of a user to each single question; forcing students to browse the same contents as the teacher (a sort of "collaborative" reaction); monitoring of effectively active students on the Web application (not only logged into the application but also actively interacting with it); determination of effectively completed course units (units where the student spent at least the minimum required time).

5 A Framework for Building ReActive Web Applications

In our framework, Web application code generation is based on WebRatio [8], a CASE tool for WebML that supports the visual design of the application schema and the automatic code generation, starting from WebML schemas and using a proprietary, extensible runtime engine. Automatic code generation is based on parametric code components corresponding to WebML units. Parametric components are configured at runtime using XML descriptors that contain SQL queries and parameters for retrieving contents from the application data source. The implementation of the extension introduced in this paper exploits WebRatios native extension mechanisms that allow adding new features by means of so-called custom units, a mechanism that has already demonstrated its power when extending the CASE tool to support other functions. The so achieved extension fully reflects the proposed (visual) design method, and supports the

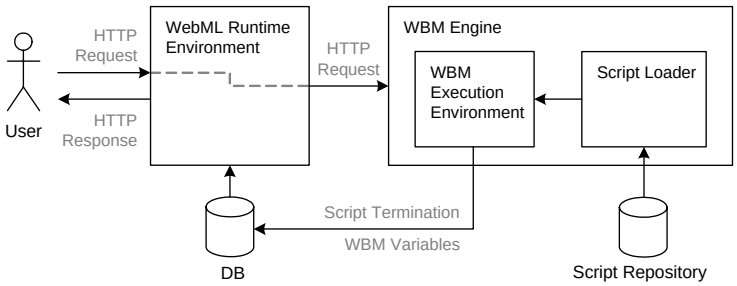

Fig. 10. Functional architecture of the overall behavior-aware system.

automatic generation and deployment of ReActive Web applications. We implemented the ReActive pages as described in Section 3 and introduced a new unit to retrieve WBM variables (`Get WBM Variable`). The implementation of ReActive pages required an extension of the page logic, yielding a further new unit (called `Active` unit), to be used in place of the `A`-label associated to reactive pages. This unit triggers the operation chains, indicated by the outgoing links from the `A`-label, when a WBM script associated to a page and to the current user terminates successfully (cf. Figure 9).

To support WBM script's modeling based on WebML schemas, we developed a visual tool that can import the XML representation of a WebML schema and use retrieved data to design WBM script (see Figure 9).

Executing Web applications reacting to users' behaviors – in addition to the standard WebML runtime environment – requires proper runtime support for WBM scripts. The implementation of rule engines for active databases is a well known and studied topic in the literature on database systems. Our problem of handling user sessions and WBM scripts resembles to the problem of handling transactions and rules in active databases. For more details on the implementation of the WBM engine refer to [2].

5.1 ReActive Architecture

Figure 10 reports the architecture of our framework: HTTP requests toward the Web application are automatically forwarded to the WBM engine by the WebML runtime environment, which hosts the actual application. Users interact only with the Web application itself and are not aware of the WBM engine behind it.

The WBM engine collects and evaluates tracked, user-generated HTTP requests for (i) instantiating new scripts at runtime, and (ii) enhancing the states of possible running WBM scripts, as well as (iii) communicating possible script terminations. Script instantiation is managed by a proper *Script loader* module and the set of scripts that can be instantiated for a particular application is retrieved from a *Script Repository*. Finally a dedicated *WBM Execution Environment*, is in charge of progressing instantiated, running scripts. Once a script reaches its

accepting state, the execution environment communicates the successful termination to the Web application by modifying suitable data structures within the shared database.

After the successful termination of a WBM script, the Web application possesses all the necessary data for executing the possibly associated actions. As soon as a user requests one of the pages within the scope of the high-level rule whose condition is satisfied by the terminated WBM script, the Web application executes the operations associated to the requested page. For this purpose, page computation starts by checking whether scripts connected to the page have terminated or not, before proceeding with the actual rendering of the page. If there are terminated scripts for that page, one or more rules could be executed. Thus, computation proceeds with the determined adaptation operations, producing effects as described in Section 3. Only afterward, if no automatic navigation actions are triggered, computation continues with the actual page, and a suitable HTTP response is produced.

6 Conclusion and Future Work

In this paper we introduced a practical case study showing a potential application of the general purpose approach for building ReActive Web applications. Furthermore, we presented our current implementation of the WBM engine and the CASE tools we used to design advanced ReActive Web sites. The adopted CASE tools prove that combining WebML and WBM yields a very powerful visual ECA model, with adequate expressive power for capturing highly sophisticated Web dynamics and providing suitable reactivity mechanisms.

In our future work, we are planning to enrich the proposed ECA paradigm, including not only events related to user behaviors but also data events and other Web events. We also intend to better integrate the two CASE tools presented for providing a unique and complete tool to easily design and deploy Web applications reacting to user behaviors.

Acknowledgement
We acknowledge the commitment of Luca Sonzogni and Cristian Rossi to the implementation of the WBM engine and the prototype application, and Luca Cavagnoli who developed the WBM CASE tool.

References

1. F. M. Facca, S. Ceri, J. Armani, V. Demaldé, Building reactive web applications, in: Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters, ACM, 2005, pp. 1058–1059.
2. S. Ceri, F. Daniel, , F. M. Facca, Modeling web applications reacting to user behaviors, to appear on a Special Issue of the Computer Networks journal on *Web Dynamics*.

3. J. Widom, The Starburst Active Database Rule System, IEEE Trans. Knowl. Data Eng. 8 (4) (1996) 583–595.
4. A. Bonifati, S. Ceri, S. Paraboschi, Active rules for XML: A new paradigm for E-services, The VLDB Journal 10 (1) (2001) 39–47.
5. G. Papamarkos, A. Poulovassilis, P. T. Wood, Event-Condition-Action Rule Languages for the Semantic Web, in: Proceedings of SWDB'03, Berlin, Germany, September 7-8, 2003, 2003, pp. 309–327.
6. F. Bry, P.-L. Pătrânjan, Reactivity on the web: Paradigms and applications of the language XChange, in: Proceedings of ACM Symposium on Applied Computing, Santa Fe, New Mexico, USA (13th–17th March 2005), 2005.
7. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, M. Matera, Designing Data-Intensive Web Applications, Morgan Kauffmann, 2002.
8. WebModels srl: WebRatio. `http://www.webratio.com`.
9. S. Ceri, F. Daniel, M. Matera, 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, pp. 225–233.