Building Reactive Web Applications

Federico M. Facca  Stefano Ceri
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Pza Leonardo da Vinci 32, I-20133 Milano, Italy
{facca,ceri}@elet.polimi.it

Jacopo Armani  Vera Demaldé
Istituto di Tecnologie della Comunicazione
Università della Svizzera Italiana
Via Lamberghetti 10 A, CH-6904 Lugano, Swiss
{armanij,demaldev}@lu.unisn.ch

ABSTRACT
The term “Adaptive Web Site” commonly refers to Web applications that evolve automatically according to the user navigational behavior. In this paper we propose a new, high level model for the specification of Adaptive Web applications, and discuss its architecture and prototype implementation. The underlying paradigm proposed for supporting Adaptive Web sites consists of Event-Condition-Action (ECA) rules; however, the rules presented in this paper have a high expressive power, as they enable capturing any arbitrary (and timed) clicking behavior, and then rebuilding the Web application according to an arbitrary hypertext, which substitutes for the current one. The model and architecture proposed are based on the WebML, a conceptual model of Web applications, and on WBM, a model of user behavior. Their combination and interpretation in terms of ECA rules provides a new, general model for specifying and then deploying Adaptive Web applications.

Categories and Subject Descriptors
H.1 [Information Systems]: Models and Principles; D.2.2 [Software Engineering]: Design Tools and Techniques; H.5.4 [Information Interfaces and Presentation]: Hyper-text/Hypermedia—architectures, navigation, user issues

General Terms
Design, Languages, Theory

Keywords
Adaptive Web, Eca Rule, User Modeling, Design Method

1. INTRODUCTION
Adaptive Web is a new research area addressing the personalization of the Web experience for each user. This term was first introduced by Perkowitz and Etzioni as a challenge to the AI community [16], and has recently developed to become a distinct research field. A number of successful applications and frameworks exist that can be regarded as Adaptive Web applications, but most of them are tailored to a specific domain (e.g. e-commerce [1], e-learning [3]), and their modelling is not effectively supported by a complete authoring tool.

We believe that a new technology and an open paradigm for building Adaptive Web sites is very much needed. Therefore in this paper we propose a model and a methodology to easily design Adaptive Web applications, and we illustrate a first prototype of architecture based on the proposed model. According to AHAM [4] adaptive hypermedia systems can be designed using a 3-layered model: (i) the user model (UM), (ii) the Web application model (WAM) and (iii) the adaptation model (AM).

In our proposal we use two existing paradigms, respectively WebML and WBM, for the WAM and UM components. We then introduce an Event-Condition-Action rule paradigm, based on the previously cited models covering the AM component. Although the Event-Condition-Action (ECA) paradigm is well known and largely used paradigm in expert systems and databases, as far as we know, our proposal is the first model to explicitly adopt the ECA paradigm for Web adaptivity.

Adaptive Web Applications normally consider up to three adaptivity dimensions: (i) the user’s behavior (e.g. user’s clickstream, user’s preferences, ...), (ii) the external environment (e.g. time-spatial location, language, ...), and (iii) the technology (e.g. user’s terminal, bandwidth, quality of service, ...).

Our research group previously presented some results related to all the dimensions of the problem [10, 8], but we did not present a unique paradigm encompassing all the three applications, and in general any reactive use of the Web; this is the focus of our paper. As in our previous studies, we follow a reactive approach, i.e., the system reacts to user interaction, instead of a proactive approaches, in which the system tries to anticipate the user’s behavior.

This article is organized as follows: initially we introduce the background models of our research (Section 2); next we introduce our ECA paradigm to model reactive adaptation and we provide some applicative examples (Section 3). In Section 4 we present the architecture of our ReActive Web System; in Section 5 we discuss some experiences that have motivated our architectural choices. Finally in Section 6 we overview some related research works and in Section 7 we address our future directions.

2. BACKGROUND MODELS

2.1 WebML: an overview
WebML (Web Modeling Language) is a conceptual model for Web application design [9], which is an ingredient of a broader development methodology, supported by a CASE...
tool, named WebRatio [9, 18]. WebML offers a set of visual primitives for defining conceptual schemas that represent the organization of the application contents and of the hypertext interface. The WebML primitives are also provided with an XML-based textual representation, which allows specifying additional detailed properties, not conveniently expressible in the visual notation.

For specifying the organization of data, WebML exploits the Entity-Relationship model, which consists of entities, defined as containers of data elements, and relationships, defined as semantic connections between entities.

WebML also allows designers to describe hypertexts, called site views, for publishing and managing content. A site view is a specific hypertext, which can be browsed by a particular class of users. Multiple site views can be defined for the same application.

Site views are internally organized into hypertext modules, called areas. Site views and areas are then composed of pages, and they in turn include containers of elementary pieces of content, called content unit. Typically, the data published by a content unit are retrieved from the database, whose schema is expressed through the E/R model. The binding between the content units (and the hypertext) and the data schema is represented by the source entity and the selector defined for each unit. The source entity specifies the type of objects published by the content unit, by referencing an entity of the E/R schema. The selector is a filter condition over the instances of the source entity, which determines the actual objects published by the unit. WebML offers predefined units (such as: data, index, multidata, scroller, multichoice index, and hierarchical index) that express different ways of selecting entity instances and publishing them within the hypertext interface. WebML also provides a unit (called entry unit) representing entry forms for inputting data.

To compute its content, a unit may require the cooperation of other units, and the interaction of the user. Making two units interact requires connecting them with a link, represented as an oriented arc between a source and a destination unit. The aim of a link is twofold: permitting navigation (possibly displaying a new page, if the destination unit is placed in a different page), and enabling the parameter passing from the source to the destination unit.

Finally, WebML models the execution of arbitrary business actions, by means of operation units. An operation unit can be linked to other operations or content units. WebML incorporates some predefined operations for creating, modifying and deleting the instances of entities and relationships, and allows developers to extend this set with their own operations.

WebML-based development is supported by the WebRatio CASE tool [18], which offers a visual environment for drawing the WebML conceptual schemas, and storing them in XML format. The core component of WebRatio is a code generator, based on XML and XSL, which automatically generates the application code from the XML specification. The generated applications run in a standard runtime framework on top of Java 2 application servers, and have a flexible, service-based architecture allowing components customization.

Recently in [8] WebML paradigm has been extended to support the context-aware adaptivity. In this study one of the most interesting issue is the introduction of a WebML adaptation chain: a sequence of WebML modeled operations that adapt the navigation and content presented to the user in reaction to user context changes.

For further details on WebML and the development methodology, the reader is referred to [9].

### 2.2 WBM: an overview

The Web Behavior Model (WBM) is a timed state-transition automata representing classes of user behaviors on the Web. 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 they constrain the way in which the navigation may occur, both in terms of used hypertext links and of time involved. Every WBM specification, called “script”, has at least an initial state, denoted by an incoming unlabeled edge, and at least one accepting state, denoted by a double circle.

Figure 1 shows a simple WBM script, whose informal semantics is as follows: when the user accesses a link that brings to Page1, the script is initialized with the initial state set to the current state. The user’s navigation to Page2 triggers the transition from Page1 to Page2, an accepting state; in such case, the script terminates successfully. The use of other links not reaching Page2 causes no state change, therefore the script may remain indefinitely in the state labeled Page1, without yielding to termination. This example shows an important aspect of the model, namely, that not all navigation alternatives must be specified (scripts capture some interesting navigations but do not need to specify or distinguish all other irrelevant navigations).

Transitions are constrained by state, link, and time constraints.

- **State constraints.** Entering a state may be subject to the evaluation of a state constraint, expressing a predicate of the pages being accessed or operation being fired; such predicate may express properties of the content displayed by the page or of the operations’ parameters. The state is accessed iff the predicate evaluation yields to true.

- **Link Constraints.** Each link 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 used.

- **Time Constraints.** Each link from a source to a target page may be labeled with a pair $[t_{min}, t_{max}]$ indicating respectively a minimum and maximum number of time units; the transition is fired iff the navigation to the target page occurs not earlier than $t_{min}$ units

![Figure 1: Example of WBM script.](image-url)
and not later than $t_{\text{max}}$ units since the time when the source page is accessed. Either one of $t_{\text{min}}$ and $t_{\text{max}}$ may be missing. Note that if the transition is not fired within $t_{\text{max}}$ then the transition can no longer occur; thus, the use of suitable time constraints may cause the failing of scripts.

Figure 2 shows an example of WBM script with state, link, and time constraints. Entering the state denoted by Page1 is constrained by pred1, which must be true; the transition from the first state to the second state should occur within $t_{\text{min}}$ to $t_{\text{max}}$ time intervals since the time when the script is initiated — thus, if the transition does not occur within $t_{\text{max}}$ time intervals since the start of the script, the script fails. Finally, the transition to the accepting state occurs if Operation1 is fired by using link1.

We allow WBM scripts to be arbitrary graphs, and therefore every state can have more than one exiting transitions, as shown in Figure 3; states labeled as Page2 and Page3 are “competing”, as a browsing activity in Page1 may lead to either Page2 or Page3. We constrain WBM scripts to have only one active state (current state) at time: every triggered transition causes a change of current state. Therefore, in the example above, when the user browses from Page1 to Page2, the transition Page1 to Page2 is triggered, and the script reaches the accepting state denoted as Page2, regardless of any subsequent browsing action. However, in WBM it is possible, and sometimes convenient, to use overlapping states, i.e., states corresponding to overlapping portions of the Web; if two competing states are overlapping, two transitions may trigger simultaneously.

For further details on WBM, the reader is referred to [12].

3. RULE MODEL

We now combine WBM scripts and WebML adaptation chains to build up the ECA rules that make Web applications adaptive. An ECA rule has a general syntax:

$$\text{on event if condition do action}$$

The event part specifies when the rule should be triggered, the condition part assesses whether a given situation has occurred, and the action part states the actions to be automatically performed if the condition holds.

In our model an Event is an user click, a Condition is a set of requirements on the user clickstream (expressed as a WBM script), and an Action is a sequence of adaptivity actions to be forced in the Web Application (expressed as a WebML adaptation chain).

Given that both WBM and WebML have a graphical notation, the rules in In Figure 4; we think that graphical notations are ideal for helping the analysts in the high-level specification of Web applications in general and of reactive Web applications specifically. For each rule, the WBM script, located on the top, represents the Event and Condition parts and the WebML adaptation chain, located on the bottom representing the Action part.

Consider for instance the rule of in Figure 4; the rule reacts to the sequence of user’s visits to Page1 and Page2; the Condition that activates the rule is that the user click stream gets to an accepting state of the script (Page2). When the Condition is satisfied, the Action is executed and the user is forwarded to Page2 after a recomputation of page parameters.

Formally, a rule is a 4-tuple:

$$(\text{scope}, (\text{WBM script}, \text{WebML chain}(\text{priority})))$$

where the scope is the extent where the Action part of the rule can be activated, i.e., a set of WebML pages, areas or site views; the WBM script describes the EC part of the rule; the WebML chain describes the A part of the rule; and, finally, the priority is an optional integer priority number to resolve conflicts when two or more rule are triggered by the same click. Sometimes it’s possible that we want to compute an adaptive action before entering a page regardless of EC part of a rule, in this case we omit the WBM script.

In the proposed Figure 4 the scope is Page2, the WBM script is the top part of the figure, and WebML chain is abstracted in the bottom part of the figure. No priority was specified in the example. In the Section 3.1 we will show how the WebML chain is actually specified.

3.1 Applicative Examples

In this section we propose some examples of ECA rules for Web adaptation. To show that our model is not bounded to any domain, we use quite different domains: e-learning, e-commerce, and e-recruiting. For each domain we provide examples by showing the EC part of the rule (the WBM model), the A part of the rule (the WebML chain of adaptation), and the scope of the rule. In order to not overload...
the user to the
the evaluation of the experience level of the user and redirect
time (e.g. 3 minutes) in the
page contained in the
it newly . Otherwise if the user spent a shorter time than
if the user is still in the
user registration
ence.
when a new student logins, and (ii) when a student is ready
to level up to the next knowledge level, and we respond to
these events with appropriate reactions.

Example 1. Evaluating the Level of User Experience. Let us suppose that, as first step, after a successful
user registration, a Pre-Test page is proposed to new users
of our e-learning environment. If the user spends a sufficient
time (e.g. 3 minutes) in the Pre-Test page, we want to force
the evaluation of the experience level of the user and redirect
the user to the Test Results page, where we present besides
the results of the test a collection of Concepts adapted to
the user experience level computed (see Figure 5a). The *
in the final state of the WBM script specifies a user request
for any page. The Action part of the rule is performed only
if the user is still in the Pre-Test page, or when he accesses
it newly. Otherwise if the user spent a shorter time than
the one required, we simply set the user experience level to
0 (see Figure 5b). In this case the Action is triggered by any
page contained in the Course Area.

3.1.1 E-learning Web application
The educational Web application in this example supports
the student’s progress through the domain concepts by the
mean of an increasing difficulty instructional strategy. Each
student is presented with the topics which are more appro-
priate to her current level of knowledge. The issues at stake
here are: (i) assigning a new student to a stereotype (ii) let
evolve her stereotype as a response to her learning progress.
Therefore in this context we apply our Rule Model to detect
the relevant events of the student’s experience, namely, (i)
when a new student logins, and (ii) when a student is ready
to level up to the next knowledge level, and we respond to
these events with appropriate reactions.

Example 2. Evolving the Level of User Experience.
Figure 6 models an ECA rule to redirect the user to a Test
page for the next experience level when the user visits 3 con-
cepts, i.e., 3 different instances of Concept pages, spending
at least 2 minutes over each page. The WebML adapta-
tion chain of operations is actually performed when the user
asks again for a Concept page. The WebML model in figure
serves the purpose of presenting the user with new questions
and answers allowing her to assess progress.

3.1.2 E-commerce Web application
Let us consider an e-commerce Web application that of-
ters to the user products like CD, DVD and VHS and Books.
The application has for registered user a personalized Web
page (MyHomepage) where most relevant services to the user
are presented. To simplify the model we consider only two
services offered by the Web application: an Auction service
and an advanced Search service. Hence in this context
we apply our Rule Model to detect events that (i) capture
user’s interest in a service and (ii) use the guessed user’s
preferences to adapt services proposed to her.

Example 3. Capturing User Interest in a Service.
Figure 7 model the user interest towards a service by not
only asking the user to use the service, but to use also the
goal for which the service was designed. Hence a user successfully uses the Search service not only if she asks for it, but also if she navigates the Search results page reaching from it a Product details page. In case of the Auction service we want the user to buy an auctioned product. As the success status is not reached immediately when the user asks for the service we need to store a WBM variable $x$ to access it from the WebML chain and increment the number of preferences given by the user to that service. The scope of the presented rule is the whole site view Registered Users: hence the action is triggered by any access to any page inside this site view.

In this example, note the use of a new WebML unit, the Get WBM Variable Unit, that is introduced specifically for supporting rules, and particularly for acquiring values of the variables which are bound by the evaluation of WBM scripts into the page computation environment of WebML

Example 4. Adapting User Home Page to her interests. Let us consider a particular case of adaptation. We want the MyHomePage to contain the service that the user mostly used until now, by exploiting the results of the rule model proposed in Example 3, i.e. the preference number given by the user to such service. We want this adaptation to be performed every time a user accesses the MyHomePage. Hence we can: (i) specify a WBM script with only one state labelled MyHomePage or (ii) omit the WBM script and simply describe the WebML chain of adaptation and its scope. In the second case, as no WBM script is specified, the WebML chain will be performed every time a user is in the scope of the rule. The model in Figure 8 describes the rule proposed using the WebML concept of Alternative Pages, with a Switch Operation Unit determining which of the two possible sub-pages has to be presented.

3.1.3 E-recruiting Web application

In this example we propose a Web application offering e-recruiting services. The main actors of the application are: users searching for a job, and users offering a job. Hence the WebML model of the application realizes one site view for each actor. We want (i) to personalize the application according to the user’s preferences traceable from her navigational choices and (ii) to show how our reactive Web design method can be used to allows interactions between different users through the data layer.

Example 5. Recommending Jobs according to user interests. We detect that a user is interested in a certain category of jobs when she navigates at least three different JobDetails pages presenting three jobs belonging to the same category. In response to this success behavior, the WebML chain stores the preference reported by the user – captured by the Get WBM Variable Unit – in the database and shows the jobs that belonging to the same category, by means of the Suggested Jobs index unit (Figure 9).

Example 6. Recommending Worker according to user offered jobs. Thanks to ECA adaptive rule in the previous example, we can display to the second actor of our web application user interested in the works she has to offer. In this case we do not need any rule or action. As shown in Figure 10 we simply need to access, through a normal WebML page, the data stored by the WebML chain in the Example 5. This paradigm can further extended in future researches to allow a deeper interaction between users not based on the data layer.

4. THE REACTIVE WEB SYSTEM ARCHITECTURE

Our reactive Web architecture requires a new server, called Rule Engine, that adds to the Web and Database Servers, as illustrated in Figure 11. The Rule Engine collects the user http requests in order to track the user navigational behavior and hosts a repository of WBM scripts, which are progressively executed on behalf of individual users.

Disregarding, for the time being, the possibility of conflicts among rules, the behavior of the rule engine is described by the following steps.

1. Url requests as generated by every user click are notified to the Rule Engine.
2. The user request can cause either the instantiation of a new script, or a state change of a given active script; in particular such change of state may lead to an accepting state.

3. When a WBM script reaches an accepting state for a certain user, the Rule Engine changes a record in the 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. Then, if the Web application should generate a page contained within the rule's scope in reaction to the user's request, the Web application recognizes in the modified database state the request of activating an adaptation chain for the current page, and it does so, generating a modified hypertext accordingly.

While the above steps represent the core of the Rule engine behavior, several variants are possible regarding the interaction between the three servers.

The use of a distributed architecture design\(^2\) brings some significant advantages:

1. Since script handling is assigned to a server which is neither the Database nor the Web server, user’s performances are not affected by the time required by rule processing\(^3\).

2. The Rule Engine Server 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.

3. The Web Application can remain operational even if the Engine Rule Server runs slow or crashes.

4. The Rule Engine Server can be recovered aside from the rest of the Web Application.

The Engine Rule Server can also act as a stand-alone system for supporting users in usability analysis of the Web interactions or the validation of given Web applications; this is the application that was intended in the first place when WBM was invented and which is reported in [12]. To this purpose, the Engine must support suitable interfaces, built for a Web behavior analysts.

4.1 The Rule Engine implementation

Our Rule Engine system is implemented as a server receiving information about user http request through a client. Essentially the Rule Engine Server needs to receive a simple message containing: the complete url requested by the User, the timestamp of the request, the code identifier of the User (e.g. the session identifier) and the code identifier of the Web Application\(^4\). Different clients can be implemented according to the different technologies used by the Web Application; In particular, we use a client capable of interacting with WebML applications, which is integrated in a Java Servlet Filter.

4.1.1 Inner Architecture

The implementation of rule engines for active database is a well known and studied topic in literature on database systems. Our problem of handling user sessions and WBM scripts resemble to the problem of handling the transactions and the rules in the active databases.

In the implementation of the architecture of the Rule Engine Server we have been inspired by the Starburst Active Database [19]. The two dimension of the problem – user sessions and WBM scripts – can be exploited to achieve an efficient implementation of the Rule Engine. If we consider the two dimension is clear that there are two possible way to handle the relationship between user sessions and WBM scripts:

1. The system implements a catalog of user sessions where each session contains the collection of the scripts activated by that user session.

2. The system implements a catalog of WBM scripts where for each active script there is the collection of the user sessions that activated the script.

Along [19], we prefer the second solution, as the number of the instances of session, in a heavily used Web application, is much greater than the number of the rules. This allows to reduce the effort needed to maintain the data structures in memory improving the system performance. In our implementation, we use generic classes for page and operation nodes that appear in a WBM script configuration. The Rule Engine server works by inserting tokens, representing user sessions, within the script data structures, then advancing tokens according to script state changes, and firing the rule’s action when token enters an accept state.

An User session is not a permanent object. Due to the HTTP protocol is neither possible to establish when an user

\(^2\)Although of course the Rule Engine, Web, and Database Server can be installed on the same machine

\(^3\)This result requires in addition to instrument the rule engine as an asynchronous process, as it will be discussed later.

\(^4\)This information can be reconstructed as well from the url being requested by the User.
session is expired. This may cause a lot of WBM scripts, that will never be able to end, to be resident in memory. To solve this problem and to handle the time constraints that can be specified in the WBM scripts, the Rule Engine has a garbage collector which performs a cyclic check and clean up of the current script activations and inactive sessions.

The Engine Rule Server stores incoming request in an ordered queue, the request are ordered by the timestamp contained in the incoming request and not by the time of arrival. We adopted this solution to maintain the consistency of the sequence of the users click streams even in case of time lag between the server and the client.

Of course a single user click (Event) can activate or modify more than one script. This fact creates the possibility of conflicts, i.e., rules that relate to the same user and page – the page currently being browsed – and are triggered together. Rules are in total priority order, based on explicit numbering or implicit rule time creation, and so the Rule Engine, in case of conflict, chooses the rule with higher priority. Once that rule is executed, and therefore the users view of the current pages is modified due to adaptation, it makes no sense to execute the lower priority rules; these are thus simply ignored. Of course, there may be a reactive Web scenario where even the low priority rules may be applicable, but we leave this option for future research.

4.2 Synchronous model versus Asynchronous model

Commonly ECA rules (triggers) of database systems try to be strongly synchronized with the database changes that cause their triggering. A similar behavior, called synchronous rule execution, is also possible with our Web architecture. With such model, if a rule is successfully triggered, then the action part of the rule is executed immediately. The Rule Engine Client, installed as a Filter to the Web Application, waits for a response from the Engine Rule Server before giving the control to the Web Application.

This strong synchronization, however, can introduce a slowdown of the user interaction with the Web application. For avoiding such problem, we introduce an asynchronous rule execution model. In such model the Engine Rule Server collects the Events from the user click stream without delaying the Web Application response; thus, the user immediately sees the Web page. However, such page is set to be dynamically refreshed after a given delay. Before refresh, the rule engine has time for running the users’ scripts, and performs a database state change in the case a rule is triggered. Such database is inspected at refresh time; if the data indicates the rule triggering, then the adaptation change is followed. A similar solution was used in order to make Web applications aware of the context [8].

In this way the user navigational choice has the priority on the adaptive design. This is normally a very reasonable way of presenting adaptive pages: first the user request is materialised exactly as expected, and next is refined by means of adaptation.

5. EXPERIENCES

This research work is the result of several previous experiences, narrower in scope, that can be seen as being “generalised” by the current approach.

- At the infancy of WebML, we studied paradigms to build reactive Web applications by using active rules in a classical e-commerce environment [10].
- Within the MAIS project we have developed two demo applications using context-aware adaptation models for museum guided tours and tourist guided tours. In these settings the scripts are quite simple and detect the movement of people within a region (as perceived through mobile devices with GPS).
- We investigated some learning scenarios and exploited the expressiveness of WBM to capture relevant learning events in the context of the student’s progress. WBM was successful in describing the collection of learning experiences that a student needs to perform before starting a new learning experience.

The architecture presented in this paper has been tested, so far, only with the students that have implemented the prototype. The feedback collected so far is quite positive.

6. RELATED WORKS

The ECA rule paradigm was first implemented in active database systems in the early nineties [19, 11] to improve the robustness and maintainability of database applications. Recently they have been also exploited in other context such as XML [2] to incorporate reactive functionality in XML documents (e.g. automatically enforcing document constraints) and Semantic Web [15] to allow reactive behaviors in ontologies evolution.

A variety of design models have been proposed for Adaptive Hypermedia [4, 7, 14]. While most of these methods differ in approach, all methods aim to provide mechanisms for describing Adaptive Hypermedia (see [5, 6] for a survey). AHAM [4], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on the Dexter [13], 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 of the use of an old-fashioned model such as Dexter and it is more suited for e-learning domain.

The model introduced in [7] extends WSDM [17], a Web design method, with an Adaptation Specification Language that allows to specify the adaptive behavior. The extension allows to specify deep adaptation in the Web application model, but lacks of expressive power as regard the specification of the user’s behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided.

[14] proposes an Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way a adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows to express conditions on the user behavior. The visual notation

---

5. We investigated some learning scenarios and exploited the expressiveness of WBM to capture relevant learning events in the context of the student’s progress. WBM was successful in describing the collection of learning experiences that a student needs to perform before starting a new learning experience.

6. Related Works

The ECA rule paradigm was first implemented in active database systems in the early nineties [19, 11] to improve the robustness and maintainability of database applications. Recently they have been also exploited in other context such as XML [2] to incorporate reactive functionality in XML documents (e.g. automatically enforcing document constraints) and Semantic Web [15] to allows reactive behaviors in ontologies evolution.

A variety of design models have been proposed for Adaptive Hypermedia [4, 7, 14]. While most of these methods differ in approach, all methods aims to provide mechanisms for describing Adaptive Hypermedia (see [5, 6] for a survey). AHAM [4], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on the Dexter [13], 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 of the use of an old-fashioned model such as Dexter and it is more suited for e-learning domain.

The model introduced in [7] extends WSDM [17], a Web design method, with an Adaptation Specification Language that allows to specify the adaptive behavior. The extension allows to specify deep adaptation in the Web application model, but lacks of expressive power as regard the specification of the user’s behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided.

[14] proposes an Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way a adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows to express conditions on the user behavior. The visual notation

---

5. We investigated some learning scenarios and exploited the expressiveness of WBM to capture relevant learning events in the context of the student’s progress. WBM was successful in describing the collection of learning experiences that a student needs to perform before starting a new learning experience.

6. Related Works

The ECA rule paradigm was first implemented in active database systems in the early nineties [19, 11] to improve the robustness and maintainability of database applications. Recently they have been also exploited in other context such as XML [2] to incorporate reactive functionality in XML documents (e.g. automatically enforcing document constraints) and Semantic Web [15] to allows reactive behaviors in ontologies evolution.

A variety of design models have been proposed for Adaptive Hypermedia [4, 7, 14]. While most of these methods differ in approach, all methods aims to provide mechanisms for describing Adaptive Hypermedia (see [5, 6] for a survey). AHAM [4], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on the Dexter [13], 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 of the use of an old-fashioned model such as Dexter and it is more suited for e-learning domain.

The model introduced in [7] extends WSDM [17], a Web design method, with an Adaptation Specification Language that allows to specify the adaptive behavior. The extension allows to specify deep adaptation in the Web application model, but lacks of expressive power as regard the specification of the user’s behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided.

[14] proposes an Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way a adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows to express conditions on the user behavior. The visual notation

---

5. We investigated some learning scenarios and exploited the expressiveness of WBM to capture relevant learning events in the context of the student’s progress. WBM was successful in describing the collection of learning experiences that a student needs to perform before starting a new learning experience.

6. Related Works

The ECA rule paradigm was first implemented in active database systems in the early nineties [19, 11] to improve the robustness and maintainability of database applications. Recently they have been also exploited in other context such as XML [2] to incorporate reactive functionality in XML documents (e.g. automatically enforcing document constraints) and Semantic Web [15] to allows reactive behaviors in ontologies evolution.

A variety of design models have been proposed for Adaptive Hypermedia [4, 7, 14]. While most of these methods differ in approach, all methods aims to provide mechanisms for describing Adaptive Hypermedia (see [5, 6] for a survey). AHAM [4], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on the Dexter [13], 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 of the use of an old-fashioned model such as Dexter and it is more suited for e-learning domain.

The model introduced in [7] extends WSDM [17], a Web design method, with an Adaptation Specification Language that allows to specify the adaptive behavior. The extension allows to specify deep adaptation in the Web application model, but lacks of expressive power as regard the specification of the user’s behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided.

[14] proposes an Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way a adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows to express conditions on the user behavior. The visual notation

---

5. We investigated some learning scenarios and exploited the expressiveness of WBM to capture relevant learning events in the context of the student’s progress. WBM was successful in describing the collection of learning experiences that a student needs to perform before starting a new learning experience.

6. Related Works

The ECA rule paradigm was first implemented in active database systems in the early nineties [19, 11] to improve the robustness and maintainability of database applications. Recently they have been also exploited in other context such as XML [2] to incorporate reactive functionality in XML documents (e.g. automatically enforcing document constraints) and Semantic Web [15] to allows reactive behaviors in ontologies evolution.

A variety of design models have been proposed for Adaptive Hypermedia [4, 7, 14]. While most of these methods differ in approach, all methods aims to provide mechanisms for describing Adaptive Hypermedia (see [5, 6] for a survey). AHAM [4], in literature, is often referred to as the reference model for Adaptive Hypertext. It is based on the Dexter [13], 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 of the use of an old-fashioned model such as Dexter and it is more suited for e-learning domain.

The model introduced in [7] extends WSDM [17], a Web design method, with an Adaptation Specification Language that allows to specify the adaptive behavior. The extension allows to specify deep adaptation in the Web application model, but lacks of expressive power as regard the specification of the user’s behavior that triggers the adaptation. No discussion on an architecture implementation of the proposed design method is provided.

[14] proposes an Software Engineering oriented model based on UML and OCL to describe in a visual and a formal way a adaptive hypertext application. The adaptation model is based on a Condition-Action paradigm that allows to express conditions on the user behavior. The visual notation
proposed lacks of immediacy and suffers the use of a visual paradigm born outside the Web area.

7. CONCLUSION AND FUTURE WORK

In this paper we proposed a general purpose model for building data-intensive adaptive web applications. Our proposal is based upon WebML and WBM, and combines these two models into a visual Event-Condition-Action paradigm that opens up the road to the implementation of high-level CASE tools for design adaptive websites.

Many classical aspects of the ECA paradigm remain to be investigated for this new application. For example, we need to develop classes of policies for dealing with priorities and conflicts (in this paper, we use simple and pragmatic policy of choosing the rule at higher priority, but this can be improved at given conditions); we did not consider rule analysis (and in particular the problem of termination, of course such problem arises when the rules trigger each other indefinitely); also, we did not consider dynamic activation and disactivation of rules and of rule groups. Most of these extensions will be considered in future versions of our prototype.

A first prototype of the reactive Web environment has been implemented with the help of several master students. It demonstrates the generality 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 the designers, and we plan to use such prototype for extensive experimentation.

8. ACKNOWLEDGEMENTS

The authors wish to thank Florian Daniel for effective discussions.

9. REFERENCES


