

MODEL-DRIVEN WEB USAGE ANALYSIS FOR THE EVALUATION OF WEB APPLICATION QUALITY

PIERO FRATERNALI, PIER LUCA LANZI, MARISTELLA MATERA, ANDREA MAURINO
*Dipartimento di Elettronica e Informazione - Politecnico di Milano, Piazza L. da Vinci, 32
20133 - Milano - Italy
[fraterna,lanzi,matera,maurino]@elet.polimi.it*

Received March, 2004
Revised October 31, 2004

So far, conceptual modeling of Web applications has been used primarily in the upper part of the life cycle, as a driver for system analysis. Little attention has been put on exploiting the conceptual specifications developed during analysis for application evaluation, maintenance and evolution. This paper illustrates an approach for integrating the use of conceptual models in the lower part of the application life cycle. The approach is based on the adoption of *conceptual logs*, which are Web usage logs enriched with meta-data deriving from the application conceptual specifications. In particular, the paper illustrates how conceptual logs are generated and exploited in Web usage evaluation and mining, so as to achieve a deeper and systematic quality evaluation of Web applications. A prototype tool supporting the generation of conceptual logs and the evaluation activities is also presented.

Keywords: Web Usage Analysis, Web Usage Mining, Quality Evaluation, Conceptual Modeling, WebML.

Communicated by: R Baeza-Yates

1 Introduction

Current Web applications are very complex and high sophisticated software products, whose quality, as perceived by users, can heavily determine their success or failure. A number of methods have been proposed for evaluating their usability [21]. Some of them focus on analyzing the correctness and consistency of design specifications [19, 23]. Such techniques certainly allow designers to improve the quality of the final product. However, they focus on a static description of the application, and do not take into account dynamic usage aspects, that can be only revealed by monitoring and analyzing the behavior of users interacting with the application. On the other hand, a menagerie of products exists that permits Web masters to analyze the Web server logs and extract information on application usage [21]. However, these tools are unaware of the conceptual schema of the application (if its specification does exist), and thus understanding the relationships between the tool's output and the structure of the application is a non-trivial task, especially if the application is large and complex.

The main contribution of this paper is a model-driven framework that integrates the model-based design and development of Web applications with quality evaluation, based on the static (i.e., compile-time) analysis of conceptual schemas and on the dynamic (i.e., run-

time) collection of Web usage data that are automatically analyzed and contrasted with the conceptual schema of the application.

In some previous papers [10, 18], we have shown how, given the conceptual schema of a Web application, it is possible to automatically analyze it with respect to some attributes for the quality of conceptual specifications, and anticipate at design time the identification of weaknesses that reduce the usability of the final application. In this paper we concentrate more on the analysis of usage data, dynamically collected during the Web application's life.

The main novelty of our work with respect to previous approaches to Web usage analysis is the integration of common Web logs with additional log data, related to hypertext components the users interact with (from basic content units to pages and to areas clustering pages), and the contents dynamically extracted from the application data source for populating such components. The resulting log data are called *conceptual logs*, because the enrichment is operated through the integration of concepts deriving from the application conceptual schema. They are specified in XML, and are analyzed by an XSL-enabled tool, which is also fed with the XML representation of the conceptual schema of the application. The tool is therefore able to interpret the log conceptual enrichments, and provide feedbacks to the conceptual designer by referencing explicitly conceptual schemas produced during model-driven design.

Our model-driven approach is therefore based on the adoption of conceptual models for application design and development. So far, its experimentation has been conducted over Web applications developed with WebML (Web Modeling Language) [7], a modeling language for data-intensive Web applications, and its supporting CASE tool, WebRatio [8]. However, we believe that the illustrated results are of general validity and apply to any application that has been designed using a model-driven approach, provided that the conceptual schema is available and the application runtime architecture permits the collection of customized log data.

The paper is organized as follows: Section 2 introduces the motivation behind our work, and illustrates some background concepts about the WebML model and its associated development process. Section 3 introduces our evaluation framework, by shortly describing the *Design Schema Analysis*, the original core of the framework, and then introducing the *Web Usage Analysis* technique, able to elaborate rich log data for verifying design soundness against user behavior, and the *Web Usage Mining* technique, able to extract interesting associations and sequential patterns from the same rich logs. Section 4 illustrates the architecture of the software tool supporting the automatic execution of the three evaluation techniques. Section 5 shows the framework at work for the evaluation of `webml.org`, the official Web site of the WebML team. Section 6 illustrates the main features of some relevant related works. Finally, Section 7 draws our conclusions.

2 Rationale and background

Modern software development methodologies advocate an agile approach, whereby a small team of designers works side by side with the application stakeholders to implement fast prototypes and evolve them into an application that meets the user's requirements [22, 11]. This process demands for pervasive evaluation, in an iterative process enabling the continuous identification of modifications necessary to cope with changed or new requirements. Since the boundary between the development phase and the operational life of applications is fuzzy,

evaluation is required both at design time, through inspection methods that reason on the application design, and after application deployment, when real usage data become available.

Iterative development methods have gained substantial benefits from the application of conceptual modeling techniques, which let developers express application requirements and/or design schemas at a high level, and automatically generate (part of) the application code [7, 20, 27]. However, the conceptual specification resulting from the initial design activities are rarely used to support quality assessment, due to the lack of model-driven evaluation methods and tools [23].

We have defined a model-based evaluation framework, able to support the analysis of Web applications, both at design time and after the application is deployed. The framework is model-based because it exploits the conceptual schemas produced in the design phase in two complementary ways: *i*) at design time, when real usage data are not available, the application conceptual schema is used for identifying design errors and inconsistency that potentially reduce the usability of the final application [10, 18]; *ii*) after the application is deployed, the conceptual schema is used for enriching Web usage logs with information referring to the hypertext conceptual schema and to the contents extracted from the application data source for computing hypertext pages. These reach logs are called *conceptual logs*; they are used for precisely reconstructing user navigation and user access to information objects, as well as for mining interesting associations among visited pages and contents.

Our evaluation framework has been defined in the context of a specific conceptual model, WebML [7], and has been implemented by extending a commercial CASE tool [8, 29]. However, we claim that the overall approach is of general validity, and can be easily adapted to other model-driven Web design methods and tools. As it will be shown along this paper, the ingredients for its implementation are the availability of an XML representation of the application conceptual schema, and the adoption of logging mechanisms for enriching Web logs with information referring to the application conceptual schema.

In the rest of this section we will shortly introduce some basic concepts about the WebML model, which are useful for the comprehension of our evaluation approach. The reader already familiar with WebML is demanded to Sections 3-5, where the evaluation framework, some architectural issues, and excerpts of an evaluation session conducted over a real Web application are presented.

2.1 WebML models

WebML (Web Modeling Language) is a conceptual model that provides a set of visual primitives for specifying the design of the information content and the hypertexts of data-intensive Web applications [7]. It is also complemented with a development methodology that, in line with other model-based development methods [4, 20, 22, 27], consists of different phases, centered around the definition and/or the refinement of the application conceptual design. Thanks to the use of a CASE tool enabling the automatic code generation [8], the conceptual design can be automatically transformed into a running prototype. This greatly facilitates the evaluation activities since the early phases of development.

WebML consists of a *Data Model* and a *Hypertext Model*, for specifying respectively the content structure of a Web application and the organization and presentation of contents in one or more hypertexts.

The WebML *Data Model* allows designers to express the organization of data, through well-known notations (namely, the Entity-Relationship and UML class diagrams). For simplicity, in this paper, we will refer to the Entity-Relationship (E/R) model, which mainly consists of *entities*, defined as containers of data elements, and *relationships*, defined as semantic connections between entities.

The WebML *Hypertext Model* allows describing how contents, whose organization is specified in the data model, are published through elementary units, called *content units*, whose composition makes up *pages*. It also specifies how content units and pages are interconnected by links to constitute *site views*, i.e., the front-end hypertexts.

The WebML Hypertext Model therefore includes:

- The *composition model*, concerning the definition of pages and their internal organization in terms of *content units*. Content units offer alternative ways of arranging contents dynamically extracted from entities and relationships of the data schema. The binding between the hypertext and the data schema is represented by the *source entity* and the *selector* of the content units: the former specifies the type of objects published by a content unit, by referencing an entity of the E/R schema; the latter is a filter condition over the instances of the source entity, which determines the actual objects published by the unit. WebML offers six predefined units (*data*, *multidata*, *index*, *multichoice index*, *hierarchical index*, and *scroller*). A further unit (called *entry unit*) represents entry forms for inputting data. The visual notation of WebML units is reported in Appendix A, Table 1.
- The *navigation model*, describing links, between pages and content units, that supports information location and hypertext browsing. Links are represented as oriented arcs, and have the double role of enabling user navigation and transporting parameters needed for unit computation.
- The *operation model*, consisting of a set of operation units specifying the creation, updating and deletion of contents, and the interaction with external services. The visual notation of the basic WebML operation units supporting content management is reported in Appendix A, Table 2.

Figure 1 shows the WebML Hypertext specification, and a possible rendition, for two pages taken from the design of the WebML.org application (<http://www.webml.org>). As better described in Section 5, this application is the official Web site of the WebML team and research. It has been developed with the WebML method, and we have used it for experimenting our evaluation framework.

The WebML *People* page contains the WebML *People* index unit, that is defined over the entity *Person*, and shows the list of people belonging to the WebML team. The link departing from the index unit, represented as an arrow, allows users to select one person from the list and navigate to the page *WebML Person*. It transports the identifier of the selected person, which is used by the data unit *Person Details*, placed in the *WebML Person* page, for showing some attributes of the previously selected person. The *WebML Person* page also includes the index unit *Published Papers* that shows a list of papers published by the selected person. The definition of this unit is based over a *selector condition*, specified below the unit, that

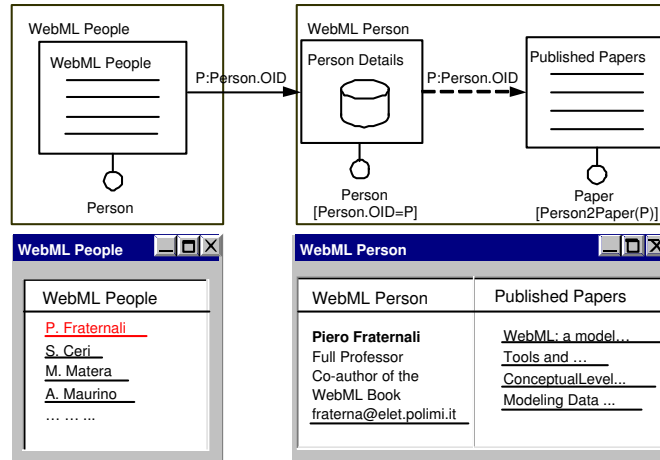


Fig. 1. Example of WebML specification (top) and a possible rendition (bottom).

```

01 <PAGE id="page11" name="WebML Person">
02 <CONTENTUNITS>
03   <DATAUNIT entity="Person" id="dau8" name="WebML Person">
04     <LINK name="ToPapers" newWindow="no" to="inu55" type="transport"/>
05     <DISPLAYATTRIBUTE attribute="Name"/>
06     <DISPLAYATTRIBUTE attribute="Bio"/>
07     <DISPLAYATTRIBUTE attribute="Email"/>
08   </DATAUNIT>
09   <INDEXUNIT id="inu55" entity="Paper" name="Published Papers">
10     <SORTATTRIBUTE attribute="Position" order="ascending"/>
11     <LINK name="See" newWindow="no" to="dau86"/>
12     <DISPLAYATTRIBUTE attribute="Title"/>
13     <SELECTOR>
14       <SELECTORCONDITION name="Person Papers"
15         attributes="OID"
16         predicate="in"
17         relationship="Person2Paper"/>
18     </SELECTOR>
19   </INDEXUNIT>
20 </CONTENTUNITS>
21 </PAGE>

```

Fig. 2. XML representation of page WebML Person, reported in Figure 1.

retrieves all the instance of the entity **Paper** associated to the selected person by means of the relationship **Person2Paper**, defined in the data schema between the two entities **Person** and **Paper**.

Besides having a visual representation, WebML primitives are also provided with an XML-based textual representation, used to specify additional detailed properties, not conveniently expressible in the graphic notation. WebML specifications can be therefore represented as visual diagrams, as well as XML documents. As an example, Figure 2 reports the (simplified) XML specification of the page **WebML person**, whose WebML visual schema is depicted in Figure 1. For further details on WebML, the reader is referred to [7].

2.2 Implementation and deployment of WebML applications

The XML representation of WebML schemas enables the automatic code generation by means of CASE tools. In particular, WebML is supported by the WebRatio CASE tool [8], which translates the XML specifications into concrete implementations.

WebRatio offers a visual environment for drawing the data and hypertext conceptual schemas, and an interface to the data layer that assists designers to automatically mapping the conceptual-level entities, attributes and relationships to physical data structures in the data sources, where the actual data will be stored. The core of WebRatio is a code generator, based on XML and XSL technologies, which is able to generate automatically the application code to be deployed on the J2EE platform. More specifically, the code generator produces the queries for data extraction from the application data sources, the code for managing the application business logic, and the page templates for the automatic generation of the application front-end.

The generated applications run in a framework implemented on top of an application server. The runtime framework has a flexible, service-based architecture that allows the customization of components. In particular, the logging service can be extended with user-defined modules, so as to log the desired data. This service has been used for gathering the conceptual data needed for the enrichment of the *conceptual logs*.

3 The evaluation framework

Our quality evaluation framework supports three types of analysis.

- *Design Schema Analysis* (DSA) verifies the correctness and consistency of design specifications [10, 18], to enhance the quality of the final application by looking for errors and irregularities in the application design. This phase focuses only on a static description of the application and does not take into account dynamic usage aspects. It is useful for inspecting the application specification, with the aim of discovering some structural problems that influence the usability of the final application. Typically, designers or expert evaluators perform this activity by hand, without using any kind of automatic tool. The automatic support can thus foster a more precise and systematic analysis.
- *Web Usage Analysis* (WUA) operates on log data dynamically collected at runtime, and produces reports on content access and navigation sequences. This analysis exploits the *conceptual logs*, that are “enriched” Web logs integrating the conventional data about HTTP page requests with information about the elementary page units and link paths accessed by the users, and the database objects used to populate pages. The aim of Web usage analysis is twofold:
 - On one hand they provide the designers with access statistics that can help identifying contents or hypertext pages and areas that are worth to be emphasized for responding either to users’s needs (as it happens for the most requested data or pages), or to the application communication goals (as it happens when some contents or pages achieve few users’access, while the application stakeholder consider them mission-critical).
 - On the other hand the analysis of users navigation paths can help discover some interaction problems due to usability lacks.
- *Web Usage Mining* (WUM) operates on the same enriched log data as WUA, and applies mining techniques for discovering interesting (sometimes unexpected) associations be-

tween accessed data. The aim is to identify possible amendments for accommodating newly discovered user needs.

The three analyses are centered on the existence of an application schema, expressed by means of a conceptual model, and exploit the schema knowledge in different manners. In particular, DSA and WUA use a top-down, goal-directed paradigm, in which evaluators have in mind precise analysis tasks, and use the application schema for formulating well targeted queries. WUM instead follows a bottom-up, inductive approach. It applies over conceptual logs for extracting “interesting” associations, which reflect user behaviors not foreseen by the application designers and that can be the symptom of design lacks, not necessarily errors. In order to specify mining queries, evaluators activate tasks for extracting associations among any logged element. Conceptual schema is then used for better interpreting the mined associations.

In the following, we will describe in details the three analysis techniques. DSA and the software module supporting the automatic analysis have been already described in [18, 10]. Therefore, their main features will be shortly recalled in Section 3.1. We will then concentrate on WUA and WUM, which are the original contributions proposed by this paper.

3.1 *Design Schema Analysis*

DSA is centered on the identification within WebML conceptual schemas of configurations, which we call *schema analysis patterns*, representing some potential sources of problems, with respect to some attributes of a model for the quality of conceptual schemas [10]. The analysis patterns consist of specific compositions of hypertext elements (pages, units, operation, links) serving a typical application purpose. Examples could be the arrangement of pages, units, and links for supporting the navigation between two relevant information objects, for accessing an information object via one or more access paths, or for creating a new information item through a content management operation.

Each quality attribute considered in our analysis is associated with:

- (i) A set of *analysis pattern descriptions*, specifying the hypertext configurations to be searched within the global application schema.
- (ii) An *analysis task* for verifying some properties of the retrieved evaluation pattern instances. It comprises:
 - A *metrics computation function*, generating aggregated numerical values that quantify the level of satisfaction of the considered quality attribute;
 - A *condition checking rule*, which highlights potential problems.

An example of metrics computation function could be the variance with which alternative patterns implementing the same function (e.g., the deletion of an information item) occur in the global application schema, whereas an example of condition checking rule may be a rule that warns the designer when s/he uses too many different hypertext patterns for solving the same problem, because this practice may contribute to increasing the users’ disorientation.

Design schema analysis is automatically performed over the XML representation of WebML schemas, by a software module called *Design Schema Analyzer*. For more details about this module and its application to the analysis of complex applications, the reader is referred to [18].

3.2 Web Usage Analysis

WUA takes an orthogonal approach to quality evaluation. It starts from run-time collected log data about the usage of the application and helps the designer evaluate if the application conceptual schema accommodates the actual way in which users browse the hypertext. Any discrepancy between the design schema and the actual usage experienced at runtime is a potential candidate for correction.

The distinguishing feature of our WUA technique is the exploitation of the application conceptual schema, which takes place in two ways:

- The raw log data are enriched with additional information, referring to the WebML content units included within pages accessed by the users, and to the data objects used for populating them. This enrichment generates the *conceptual logs*.
- The result of analyzing the user navigation behavior is displayed directly on top of the graphical representation of the hypertext schema, thus facilitating the understanding of quality assessment results.

WUA is supported by a software module, named *Web Usage Analyzer*, that elaborates the conceptual logs, by computing some access statistics coded as XSL rules. As for the Design Schema Analyser, the conceptual log computation is based on two sets of rules, one dictating the elements to be retrieved in the log, the other specifying analysis tasks to be executed over them.

The present version of the Log Analyzer supports two kinds of assessments, *access analysis* and *navigation analysis*, which will be described in the sequel.

3.2.1 Access Analysis

For the majority of log analyzers, access analysis consists of computing statistics on user accesses to pages. However, our model-based approach, which separates the structure schema and the hypertext schema, supports two types of analysis:

- *Data Access Analysis*, which computes statistics on the access to database entities and their instances, purposely ignoring the hypertext interface.
- *Hypertext Access Analysis*, which focuses on the usage of the hypertext pages, areas, and site views.

Data Access Analysis can for example respond to such questions as “*Which is the most/least accessed entity?*” or “*Which is the most/least accessed instance of Entity X?*”.

Hypertext Access Analysis extends the statistics normally offered by state-of-the-practice log analysis tools. First, it enables the analysis of accesses to hypertext components at different levels of granularity, covering not only page visits, but also accesses to site views, i.e., entire hypertexts defined as application front-ends, areas within site views, i.e., cohesive sets of pages publishing contents about some core objects of the application, and individual content units within pages. Second, thanks to the cross-links among the log entries, the elements of the data schema, and the elements of the hypertext schema provided in the conceptual logs, Hypertext Access Analysis may take into account the actual data objects used to fill-in the requested pages. Therefore, it can respond to such questions as “*Which is the most/least accessed page/area/site view displaying the content of Entity X?*” or “*Which*

is the most/least frequently used page for displaying a specific instance of Entity X?. These results greatly help designers evaluate the effectiveness of the hypertext in delivering the core content of the application.

An important aspect of the Web Usage Analyzer is that the evaluator can pose quality evaluation queries involving the accessed information objects, without being aware of the physical structure of the application's URLs and, in particular, of the parameters that identify the actual objects used at runtime to fill-in the page. Objects identifiers are managed by the runtime logs and abstracted as XML elements in the conceptual logs^a

3.2.2 Navigation Analysis

Besides access analysis, also navigation analysis plays a central role in hypertext quality evaluation. While access analysis is relevant for understanding if published data are accessed by users, and for identifying possible elements that need more emphasis in the application interface, Navigation Analysis concentrates on verifying if the hypertext topology supports content accessibility, or if conversely it presents some usability problems. In particular, it allows reconstructing navigation paths adopted by users for reaching some core information objects by means of access paths. This analysis is based on a WebML methodological assumption that distinguishes among different roles that information concepts can play within the Web application [9]. In particular, information concepts can be classified as:

- *Core concepts*, when they are the “central” objects, forming the main asset and expressing the mission of the Web application (e.g., the product sold in a B2C site, or the personal message published in a community center).
- *Access concepts*, when they support the location of core concepts.
- *Interconnection concepts*, when they interconnect core concepts; they are typically expressed by means of relationships, which the user can navigate to move the focus from one core concept to another related one.

In order to perform well focused navigation analyses, evaluators can therefore select in the hypertext schema units and pages publishing core information objects. Based on this input, Navigation Analysis then reconstructs *access paths*, defined over access entities categorizing the core entities, and *interconnection paths*, defined over relationships interconnecting the core entities. The analysis of the reconstructed paths, and their comparison with the conceptual schema then permit to highlight critical points, or also deviations with respect to paths provided by the designer (for example, when users make frequent use of the browser's back button or neglect some navigation paths provided by the designer, or when users incur in some loops).

3.3 Web Usage Mining

WUM operates on conceptual logs, and applies XML mining techniques for discovering interesting (sometimes unexpected) associations among visited hypertext elements and among accessed data.

^aIn real Web sites, identifying the accessed objects may be difficult due to URL scrambling for security reasons, or due to the heavy use of POST-based HTTP requests, as common, for example, in the Microsoft .NET platform.

The execution of mining statements over conceptual logs produces:

- *XML association rules* of the form $X \Rightarrow Y$, stating that when the log element X (called the *rule body*) is found, it is likely that the log element Y (called the *rule head*) will be also found. Depending on the adopted mining statement, the retrieved association can be related to database entities or instances, hypertext components (areas, pages, content units), or also hypertext components coupled with their populating data instances.
- *XML sequential patterns*, in which the rule body and head are also bounded to their position in the log sequence, so as to indicate the existence of a temporal relation.

Appendix A reports an example of sequential pattern mined from the `webml.org` conceptual logs. As can be observed, extracted patterns are enclosed within the root tag `<Sequences>`. Each single pattern is then enclosed by the tag `<SequenceRule>`, while the rule body and the rule head are respectively enclosed by the tags `<Antecedent>` and `<Consequent>`. As it will be better explained in Section 4, with the aim of facilitating their interpretation and supporting deeper analyses, extracted rules also include adjunctive properties and data that are not present in HTTP requests, but are retrieved from the application conceptual schema.

Based on the extraction of such rules, so far we have focused on three specific mining tasks:

- Finding areas or pages that are often visited together, through the mining of association rules between areas or pages in the same user session.
- Finding data that are often accessed together, considering as transaction a user request, implemented through the mining of association rules between data entities and instances accessed within the same user session. It is worth noting that such associations are not easily discovered in traditional logs that do not record data instances used to populate dynamic Web pages, and generally require several post-processing efforts.
- Analyzing user navigation sequences for accessing core contents, by mining sequential patterns related to sequences of pages and content units within the same user session. The WebML characterization of information concepts and content units allows filtering sequences, concentrating the analysis on relevant navigation paths leading to some selected core concepts.

4 The framework architecture

Figure 3 illustrates the overall architecture of the WebML quality evaluation framework. It is based on three layers: the *Data Extraction layer*, in charge of gathering and pre-processing input data, the *Analysis layer*, which computes log data gathered through the data extraction layer, and the *Result Visualization layer*, which visualizes the analysis results.

Among layers: the *Analysis Tasks* repository stores the analysis procedures, which can be expressed both in XSL and XQuery; the *Analysis Data Warehouse* stores data needed for analysis, represented in XML format; the *Result Warehouse* then stores the XML-based representation of the results produced by the analysis, which is then used by the graphical user interface for generating and visualizing the analysis reports.

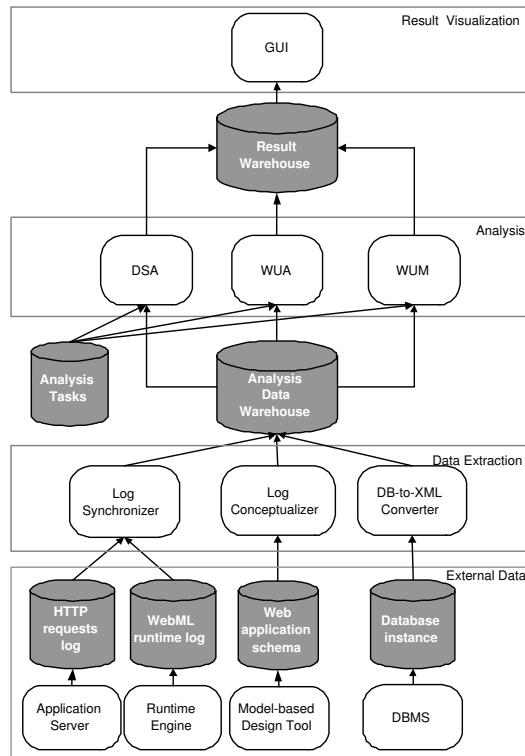


Fig. 3. The evaluation framework architecture.

It is worth noting that the ubiquitous use of XML technologies improves the number of strategies the evaluator can adopt in order to manipulate and query data. Also, the quality evaluation framework results to be very flexible and extensible: new analysis tasks can be easily specified and added to the framework. Therefore, each design team can define its own quality criteria, code their measures in XSL or XQuery, two extensively used W3C standards, and adding them within the the *Analysis Tasks* repository. Additionally, the use of warehouses between layers enables adding new software modules in a given layer, for example for performing new kinds of analysis, without affecting other components.

The rest of this section is devoted to describing the input data of the framework and the pre-processing applied over them for gathering conceptual logs, and the modules composing the three layers.

4.1 *Input data processing*

The input to the framework consists of:

- HTTP requests log generated by the application server in the ECLF format [30], also including user session IDs, for univocally identifying page requests belonging to a same user session. This feature greatly simplifies the pre-processing necessary for reconstructing users' sessions [13]; it also allows distinguishing among requests coming from

different user agents, even when proxies filter them or when users turn off cookie support on their browser. Figure 4 reports some lines, extracted from the application server log, that represent the request of a page from the webml.org application (the same page specified in Figure 2). For privacy reasons, the IP address of the requesting host has been replaced with a sequence of Xs. The logs include the `JSESSIONID` field, whose value (`acbTwnzgkSz6`) identifies the user session.

- The *WebML runtime log*, an XML log file storing events and data produced and consumed by the application runtime for serving page requests. Runtime log data are generated thanks to an extension of the WebML runtime environment that is based on Log4J, an open source package distributed by the Apache Software Foundation [2]. They include all the events generated by the application runtime when serving a requested page and populating its content units. An event can represent either the request of a page, or the computation of an individual unit within the page. Since each page request is managed by a specific thread, the events generated for a single page request are characterized by the same thread number. In the log file each event is delimited by the `<event>` tag that may also contain further sub-tags:
 - The `<message>` tag, comprising the event parameters. In case of content units population, it also includes the list of identifiers (OIDs) of the objects extracted from the data source.
 - The `<NDC>` tag, storing the identifier of the conceptual element (page or unit) to which the event refers.

Figure 5 shows an extract of the runtime log generated for the same page request illustrated in Figure 4. The recorded events are characterized by the same thread (`tcpConnection-80-17`). The first event (lines 1-7) denotes the page request. The other events (lines 8-14 and 15-21) denote the population of the units of the page. For example, the second event (line 8-14) refers to the population of a data unit (`dau84`). Its `<message>` tag (lines 10-12) includes the unit ID (`dau84`), the client IP address and SessionID, and a value (17) representing the OID of the single database instance extracted for populating the data unit. The third event (lines 15-21) refers to the population of an index unit (`inu9`). Its `<message>` tag includes a list of values (line 18) representing the OIDs of the three database objects extracted for populating the index unit.

- The XML representation of the application conceptual schema, automatically generated by the model-based design tool.
- The application data source, whose knowledge within the evaluation framework is needed for interpreting the identifiers of data instances logged in the WebML runtime log.

4.2 Data Extraction layer

The *Data Extraction Layer* is in charge of gathering the input data previously described, and importing them into the *Analysis Data Warehouse*, after performing three main actions:

```

01 XXX.XXX.XXX.XXX[29/Mar/2003:02:03:34 +0100]
02 "GET page11.do HTTP/1.1" 200 122018 http://webml.org
  "Mozilla/4.0"
03 (compatible;MSIE 6.0; Windows NT 5.0)" JSESSIONID=acbTwnzgkSz6}

```

Fig. 4. Extract from the application server log.

```

01 <log4j:event timestamp="Sat, 29 March 2003 - 02:03:34.959"
02     thread="tcpConnection-80-17">
03     <log4j:message>
04         Requested page service for id=page11
05     </log4j:message>
06     <log4j:NDC>page11</log4j:NDC>
07 </log4j:event>
08 <log4j:event timestamp="Sat, 29 March 2003 - 02:03:34.962"
09     thread="tcpConnection-80-17">
10     <log4j:message>
11         [dau8] [XXX.XXX.XXX.XXX,acbTwnzgkSz6]4
12     </log4j:message>
13     <log4j:NDC>dau8</log4j:NDC>
14 </log4j:event>
15 <log4j:event timestamp="Sat, 29 March 2003 16 - 02:03:34.969"
16     thread="tcpConnection-80-17">
17     <log4j:message>
18         [inu55] [XXX.XXX.XXX.XXX,acbTwnzgkSz6]15,24,10,11,16,9,14
19     </log4j:message>
20     <log4j:NDC>inu55</log4j:NDC>
21 </log4j:event>

```

Fig. 5. Extract from the runtime log.

- Synchronizing log data stored in the application server log and in the WebML runtime log, thus obtaining the *synch log*. This action is performed by a software module called *Log Synchronizer*.
- Enriching the synch log, by relating each logged request with the corresponding elements specified in the conceptual schema. The aim is to make log files more readable and easily analyzable, by means of the semantics deriving from the conceptual schema. This action is performed by a software module called *Log Conceptualizer*, and generates the *conceptual logs*.
- Generating an XML dump of the data source, through the use of a DB-to-XML converter [28]. The aim is to gather a uniform format, based on XML, for all the data needed for executing the evaluation tasks.

In the following, the Log Synchronizer and Log Conceptualizer modules are described in more details.

4.2.1 *Log Synchronizer*

This module synchronizes the application server log and the WebML runtime log, and generates a unified XML representation of all the available log data. Not all the requests stored in the application server log need to be considered for synchronization. Requests generated by software agents (spiders, Web crawlers, search engines, etc.), or requests of image files included within pages can be filtered out. Also, errors generated by incorrect user requests or server exceptions, which are characterized by status code 400 or 500, can be excluded from synchronization, but they are however stored in different files used for calculating statistics

```

01 <Request Request_Id="3178">
02   <LocalTime>
03     <DD>29</DD> <Month>Mar</Month> <YY>2003</YY>
04     <hh>02</hh><mm>03</mm><ss>34</ss>
05     <Timestamp>+0100</Timestamp>
06   </LocalTime>
07   <User>
08     <IPAddress>XXX.XXX.XXX.XXX</IPAddress>
09     <jSessionID>acbTwnzgkSz6</jSessionID>
10     <Browser>MSIE</Browser>
11     <Version>6.0</Version>
12     <Platform>compatible</Platform>
13     <OS>Windows NT 5.0</OS>
14     <CountryName/>
15   </User>
16   <Page SchemaRef="page11">
17     <PageContent>
18       <Unit>
19         <UnitSpecs SchemaRef="dau8"/>
20         <Data_Oid>4</Data_Oid>
21       </Unit>
22       <Unit>
23         <UnitSpecs SchemaRef="inu55"/>
24         <Data_Oid>15</Data_Oid>
25         <Data_Oid>24</Data_Oid>
26         <Data_Oid>10</Data_Oid>
27         <Data_Oid>11</Data_Oid>
28         <Data_Oid>16</Data_Oid>
29         <Data_Oid> 9</Data_Oid>
30         <Data_Oid>14</Data_Oid>
31       </Unit>
32     </PageContent>
33   </Page>
34 </Request>

```

Fig. 6. Extract from the synch log.

about application faults. As reported in Figure 6, each request in the application server log is extended with the corresponding events and data coded in the runtime log, namely:

- The identifiers of the units composing the page are delimited by the `<UnitSpecs>` tag.
- The OIDs of the database objects extracted for populating units are delimited by the `<Data_Oid>` tag.

Finally, requests are sorted by user session. The result is the *synch log* XML file. A fragment of such a file, deriving from the synchronization of the two logs shown in Figure 4 and Figure 5, is reported in Figure 6.

It is worth noting that the attribute `SchemaRef`, defined for pages and units nodes, represents values that univocally identify pages and units within the application conceptual schema. Therefore it provides a reference for retrieving additional properties, not traced by the logging mechanism, but represented in the conceptual schema, and to integrate them in the conceptual log.

4.2.2 Log Conceptualizer

The *Log Conceptualizer* is in charge of generating the final *conceptual log*, which merges the synch log with relevant structural properties specified in the XML-based conceptual schema of the application, such as names of pages, of areas enclosing pages, of page units and their source entities, etc. Figure 8 shows an extract of the conceptual log generated for the webml.org

application. The binding between log data and the conceptual schema elements is obtained through the `SchemaRef` attribute that characterizes pages and units in the synch log.

One relevant enrichment operated by the Log Conceptualizer is the *link reconstruction*, which identifies WebML links selected by users to move among pages. For each request, the HTTP server registers the referrer page only, that is the page from which the link is started, but it cannot identify which link, among the ones departing from the referrer page, is followed by the user. Such information is thus retrieved by means of a *link reconstruction algorithm* that, given the request for a page:

- Identifies the referrer page from which the request originates.
- Analyzes the schema of the referrer page and the parameters appended to the HTTP request.
- Retrieves the WebML link departing from the referrer page that transports the request parameters, and adds its identifier within the conceptual log, delimited by the tag `EntryLink` (see Figure 8 for an example).

Such a log enrichment enables the precise reconstruction of user navigation paths, including the identification of the use of the browser back button, an interesting navigation behavior that can reveal some usability problems [26].

Back identification is based on the comparison, for each requested page, between the referrer page and the previously requested page. The common situation found in log files is in fact that, given a request, its referrer page corresponds to the previously logged page request. When the user uses the back button, the HTTP page request of the target page does not appear in the log. This is because actually the page is not requested to the HTTP server, since it is maintained in the browser cache. Therefore, an indicator of the back use can be, for a given request, the mismatch between its referrer page and the previously requested page.

Figure 7(a) shows such a situation where a user, starting from Page A, visits Page B (arrow 1), then, by means of the back browser button, comes back again to Page A (arrow 2), and finally sees Page C (arrow 3). The ECLF log file (Figure 7(b)) will register the request of Page B from Page A, and of Page C from Page A, but it will not log the request of Page A from Page B, because it does not require an explicit HTTP request. We call this navigation pattern *shadow request*, because it is hidden to the server - and does not appear in the log. Once identified, *shadow requests* are registered into the synch log by adding, for the page requested after the back navigation, a `Referrer` element with `SchemaRef="Shadow"`, and an `EntryLink` element with `SchemaRef="Back"`.

The conceptual log reported in Figure 8 shows an example of shadow request identification. According to such a log, a user accesses the `WebML Paper` page (lines 01-13), after s/he has visited the `Category Papers` page (line 03). Navigation from `page41` to `page92` occurs by means of `link38` (line 04). Then, s/he uses the browser back button to come back to the `Category Papers` page (lines 14-27); after that, s/he navigates again `link38` and accesses the `WebML Paper` page (lines 28-41), to see another paper (`Data.Oid` equals 4 in line 36).

4.3 *Analysis layer*

The *Analysis* layer represents the core of our architecture: it computes evaluation results from data stored in the *Analysis Data Warehouse*. The layer is composed of three software

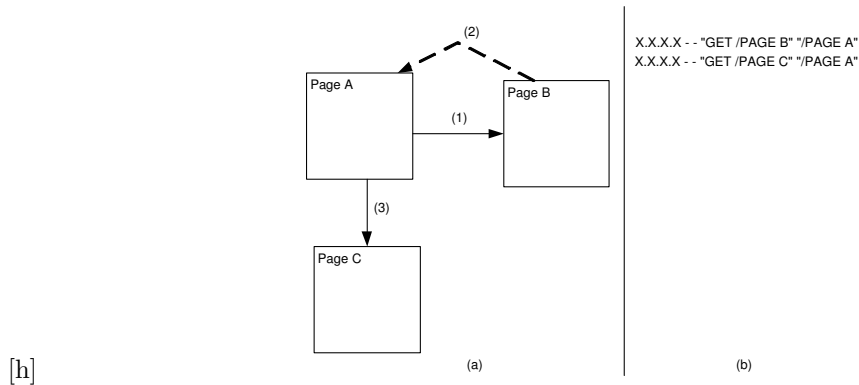


Fig. 7. Example of use of back browser button (a) and the corresponding (simplified) ECFL file.

components that implement the three types of analysis described in Section 3:

- The **DSA module** analyzes the XML representation of hypertext conceptual schemas, by means of two sets of XSL rules, stored in the *Analysis Tasks* repository:
 - *Analysis Pattern Description* rules, representing the XSL specification of *analysis patterns* to be retrieved and analysed within the schema.
 - *Analysis Task* rules, representing the XSL specification of the analysis procedures to be performed over analysis patterns retrieved with the previous rules.

For more details about the **DSA** component, the reader is referred to [18].

- The **WUA module** elaborates the conceptual logs using two kinds of rules stored in the *Analysis Tasks* repository:
 - *Usage Pattern Description* rules, representing the log elements to be retrieved and analyzed. A usage pattern can be a data element (an entity or an entity instance), a hypertext component (a page or an area) and the data instances used for their computation, or a core unit for which one wants to reconstruct the access and interconnection paths navigated by users.
 - *Statistics Computation* rules, representing procedures for computing analysis over the retrieved usage patterns.
- The **WUM module** also elaborates conceptual logs, by executing some mining tasks stored in the *Analysis Tasks* repository. It is implemented on top of **XMINE**, a tool developed for mining interesting relations from native XML documents [5]. The interoperability between this module and those generating the conceptual logs is guaranteed by the extensive use of XML technologies. The mining tasks that **WUM** is able to execute are expressed through a set of **XMINE** statements that resemble many similarities with XQuery statements. The execution of these **XMINE** statements over the conceptual logs produces rules, in form of *XML association rules* or *XML sequential patterns*. This output is then filtered by means of XQuery statements, to select most interesting results.

CategoryPapers

```

01 <Request Request_Id="162">
02   <Page SchemaRef="page92" Name="WebML paper" Area="Research">
03     <ReferrerPage SchemaRef="page41" Name="Category Papers" Area="Research"/>
04     <EntryLink SchemaRef="link38"/>
05     ...
06   <PageContent>
07     <Unit>
08       <UnitSpecs SchemaRef="dau39" Name="Paper" SourceEntity="Paper"/>
09       <Data_Oid>3</Data_Oid>
10     </Unit>
11   </PageContent>
12 </Page>
13 </Request>
14 <Request RequestId="163">
15   <Page SchemaRef="page41" Name="Papers Category" Area="Research">
16     <ReferrerPage SchemaRef="Shadow">
17       <EntryLink SchemaRef="Back"/>
18     ...
19   <PageContent>
20     </Unit>
21     <UnitSpecs SchemaRef="dau47" Name="Category" SourceEntity="PaperCategory"/>
22     <Data_Oid>15</Data_Oid>
23   </Unit>
24 </PageContent>
25 </Page>
26 ...
27 </Request>
28 <Request RequestId="164">
29   <Page SchemaRef="page92" Name="WebML paper" Area="Research">
30     <ReferrerPage SchemaRef="page41" Name="Category Papers" Area="Research"/>
31     <EntryLink SchemaRef="link38"/>
32     ...
33   <PageContent>
34     <Unit>
35       <UnitSpecs SchemaRef="dau39" Name="Paper" SourceEntity="Paper"/>
36       <Data_Oid>4</Data_Oid>
37     </Unit>
38   </PageContent>
39 </Page>
40 ...
41 </Request>

```

Fig. 8. Fragment of conceptual log.

4.4 Result Visualization layer

The last layer of our software architecture allows evaluators to invoke the analysis tasks stored in the *Analysis Tasks* repository, and shows the results, through a graphical user interface developed in JAVA. Some visualization examples will be shown in the following section.

5 Case study

In order to prove the effectiveness of our framework, we have applied it to the *webml.org* application (<http://webml.org>), the official reference site about Web modeling and the WebML language. It publishes a rich set of resources, including excerpts of the WebML book, organized by chapters, downloadable materials, papers written by WebML researchers and by other research groups, tutorials and exercises useful for WebML teachers and students. The site has been designed with WebML and deployed with WebRatio on a J2EE platform. It consists of two site views:

- A *content management site view*, for gathering, storing and updating contents dynamically published in the public Web site. It is organized along eight different areas, whose

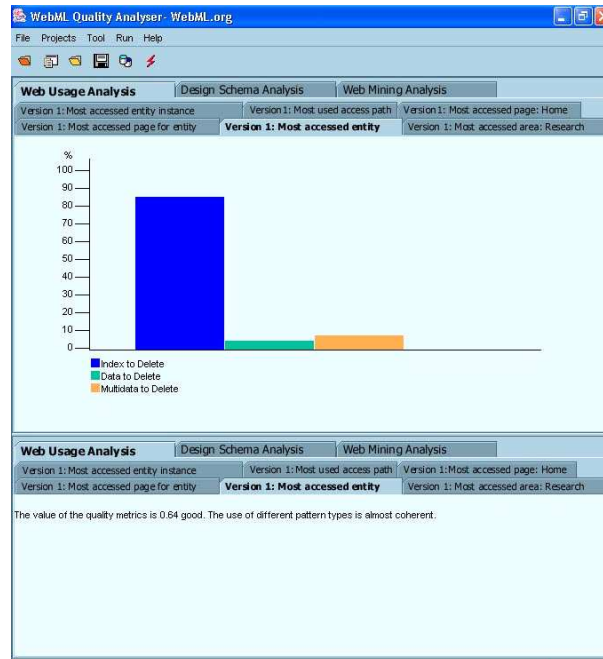


Fig. 9. Consistency analysis computation in the Design Schema Analyzer.

pages allow invoking content management operations, such as creating, modifying and deleting database objects. The access to such a site view is restricted to registered users only.

- A *public site view*, accessible from any unregistered user interested in retrieving contents about Web Modeling, the WebML language, and its related initiatives. The public site view consists of seven different areas (*Overview, Book, News, People, Teaching, Research, Industry, and Community*).

The data schema of *webml.org* is centered on a few core entities (**Book, Paper, Material, Exercise**), which are interconnected among them and associated with a few additional access entities (e.g., **MaterialType, PaperCategory**, and so on) serving the purpose of categorizing the site's content.

The hypertext schema is quite complex and permits the users to reach the same piece of content (e.g., an exercise on hypertext modeling) in different ways. The application has been published online on 28/03/2003 and has an average of 60 unique visitors per day.

5.1 Design Schema Analysis

Figure 9 reports the result visualization for the consistency analysis of delete operations applied over the conceptual schema of *webml.org*. Delete operations are used for specifying the deletion of some information objects. The analysis procedure identifies all the **Delete** operations specified within the schema, and verifies if they are designed coherently. The bar chart shows that:

- In the 87% of cases the deletion is activated by selecting the object to delete from a list of candidate objects, described by few key attributes (*Index To Delete* configuration).
- In the 8% of times the deletion is activated by selecting the object from a list where each object is fully described by means of all its attributes (*MultiData To Delete* configuration).
- Only in the 5% of cases the deletion is performed by first displaying the list of candidate objects, then the details of one single object selected from the list, and then activating the operation (*Data To Delete* configuration).

The distribution of occurrences of the last two configurations highlights a lack of consistency. A corrective action could therefore consist of adopting the same design solution every time a delete operation occurs in the hypertext. It is worth noting that in some cases inconsistencies are caused by conscious design choices, needed for responding to specific application constraints. However, in the situation above described, as also confirmed by the application designers, the inconsistency was not an explicit choice, but a mistake that needed to be fixed.

5.2 Access Analysis

Figure 10 shows a data access analysis graph, generated by the Web Usage Analyzer from log data relative to a period of 15 days in March 2003. The graph shows that the most accessed entity is the one that represents the WebML book chapters, which gets 23% of the user's requests. By clicking on an entity, it is then possible to drill down to the distribution of accesses across the entity instances. In the specific case, Chapter TOC (Table Of Content) results the most accessed one.

Access Analysis permits to obtain further results. For example, Figure 11 ranks all the hypertext pages that have been used to access the TOC chapter, sorted by the number of users' accesses. **Book Home** page, contained in the **Book** section of the site, is the most used, featuring 42% of the total accesses to TOC. The second most used page, with 15% of accesses, is **Selected Page**; inside the **Book** section, it displays sample pages, in PDF format, extracted by a given chapter.

Based on the previous results, designers have been suggested to accommodate the user preferences, facilitating the access to the most requested data. They might enrich the key pages of the site, for example the Home Page, with navigation shortcuts to the most requested pages publishing the book TOC (**Book Home** or **Selected Page**), for letting users access such contents in one click [24].

5.3 Navigation Analysis

We have applied Navigation Analysis for analyzing access paths followed by users to reach the content unit **PaperDetails**. The aim was to verify the accessibility level by users for the core entity **Paper**. In fact, the unit **PaperDetails** publishes data about a WebML paper within the **Research** section of the public Web site.

Figure 12 shows two access paths, as defined by designers in the application conceptual schema, both leading to the content unit **PaperDetails**:

1. A path exploiting the access entity **PaperCategory**. It comprises the index unit named **PaperCategories** (in page **WebML papers**), presenting the list of the available categories, the data unit **Category** (in page **CategoryPapers**), representing data about one

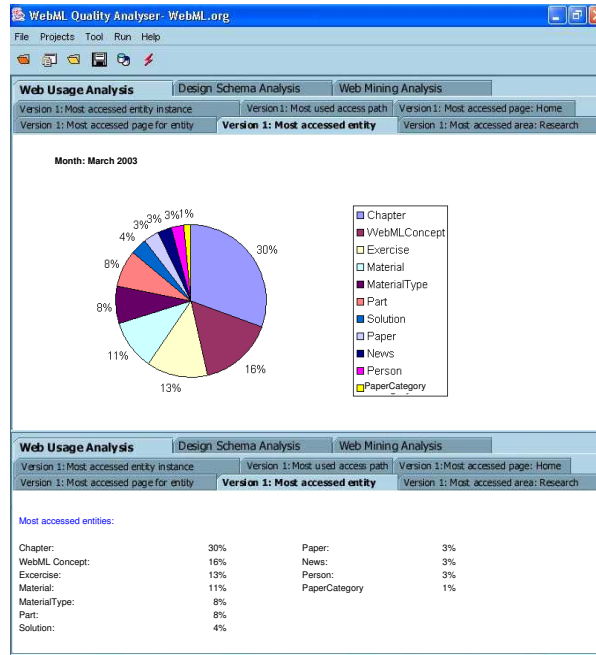


Fig. 10. A report for the “Most accessed entity” analysis task.

selected category, and the index unit **Papers**, which presents the list of papers in the selected category.

2. A second path, still exploiting the access entity **PaperCategory**. It is composed of a hierarchical index unit (**PapersByCategory**) placed in page **WebML Papers**, which publishes the list of papers hierarchically organized with respect to paper categories, and allows users to select one paper and navigate to the paper data directly, bypassing page **CategoryPapers**.

Figure 13 illustrates the reconstruction of access paths, as resulting from the *webml.org* conceptual logs. As can be noted, the user navigation is represented directly over the hypertext conceptual schema, thus facilitating the comparison. The analysis highlights that in the majority of cases, users access the **PaperDetails** unit by following one of the two defined access paths (39% of times for path 1 and 32% of times for path 2). However, navigation analysis has also highlighted that in the 29% of times, after reaching the **PaperDetails** data unit, users resort to the back button of the browser to go back to the **Category Papers** page, select a new paper from the **Papers** index unit, and access a new paper instance. The use of back button is highlighted in the schema by means of dotted lines. It is discovered through the algorithm already described in Section 4.2.2.

The results of this analysis highlights that:

- Several users, after accessing a given paper, are interested in accessing other papers in the same category.

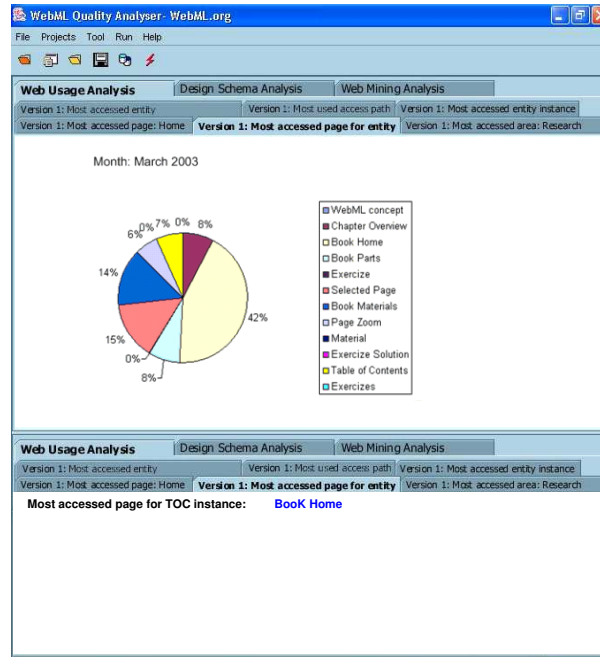


Fig. 11. Accesses distribution of the most accessed instance along the different pages publishing its data.

- in order to select a new paper, users are asked to perform some navigation steps that are not supported by an explicit link, and for this reason adopt the back button.

Therefore, the duplication of the index of papers in the WebML Paper WebML Paper page can provide users with a mechanism for moving among papers of a given category; it also reduces the number of navigation [24].

5.4 *Web Usage Mining*

One of the analysis tasks applied over WebML conceptual logs has consisted of mining navigation sequences leading to core contents.

The sequential pattern reported in Appendix A represents an interesting result. It indicates that there exists a temporal relation between (i) the access to the page Overview (lines 4-26), introducing general concepts about WebML, and (ii) the access to page WebML Material (lines 29-44), publishing a tutorial on WebML^b. The support of the pattern is 0.04 (line 2); it indicates that in the 4% of the navigation sequences, users who access the Overview page also access the WebML tutorial in one of the following navigation steps. Most important, the pattern confidence is 0.12 (line 2), showing that, if users access the Overview page, there

^bThe identification of the content for page WebML Material is possible thanks to the log enrichment through OIDs of the data objects populating the accessed pages. The value 11 for the element DataInstance (line 35) of the unit Material Details (line 34), having Material as source entity (line 34), in the XML dump of the data source corresponds to a PDF file of the WebML tutorial.

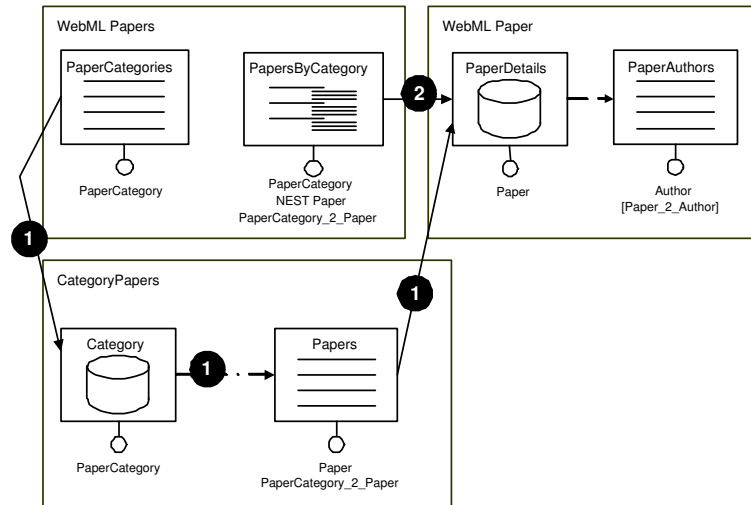


Fig. 12. Portion of the webml.org schema, representing the two access paths for reaching papers.

is an estimated probability of 12% that an access to the tutorial page will be also found in the next page requests.

From the conceptual schema it results that between the two pages *Overview* and *WebML Material* there is no direct navigation path; therefore, the retrieved relation between the two pages highlights an unexpected user behavior, not supported by the hypertext interface, and therefore difficult to identify through the simple analysis of navigation paths, as performed by the *WUA* module. This finding suggests two possible ways of reconstructing page *Overview* for accommodating this discovered user need: (i) the page could include a link to the instance of *WebML Material* page publishing the *WebML tutorial*; (ii) from the page, it should be possible to directly download the *WebML tutorial*.

6 Related work

Web application quality has been so far pursued mostly by proposing best practices and design principles [24], or by means of structured model-based development methods [4, 20, 27].

Some tools have been developed for the automatic analysis of Web page design [21]. However, they mostly operate over the HTML coding of pages, with the aim of discovering presentation problems, while they neglect structural and navigation problems. The evaluation method illustrated in [23] addresses such issues, by prescribing a design inspection technique. However, it does not offer automatic support. Our *DSA* module can be therefore considered one of the first initiatives for the automatic analysis of conceptual schemas.

Several methods and tools have been proposed for the automatic analysis of Web logs. They have two emerging goals: (i) calculating statistics about site activities, and (ii) mining data about user profiles to support personalization. The majority of the public and shareware tools (see for example [1, 3, 6]) are traffic analyzers. As also described in [15], their functionality is limited to producing:

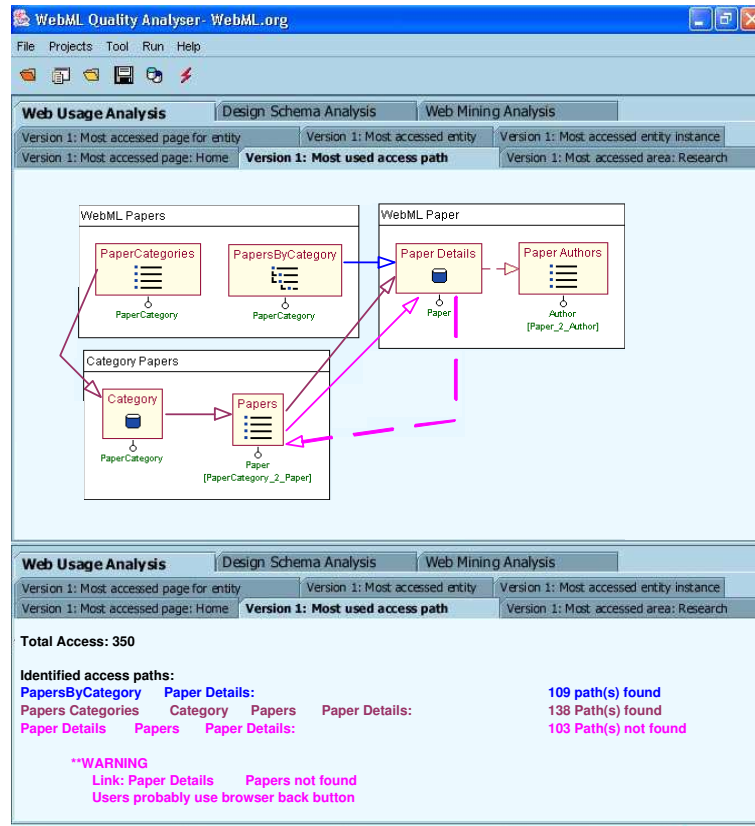


Fig. 13. Visualization of results for the analysis of access paths over the core unit *PaperDetails*.

- *Reports about site traffic*, such as total number of visits, average number of hits, average view time, etc.
- *Diagnostic statistics*, such as server errors and pages not found.
- *Referrer statistics*, such as search engines accessing the application.
- *User statistics*, such as top geographical regions.
- *Client statistics*, such as user's Web browsers and operating systems.

Only few of them (for example [3]) also track user sessions and present specific statistics about individual users' accesses.

In addition to calculating traffic statistics, our WUA module is able to compute advanced statistics, related to database entities and instances, and to hypertext components of any granularity. In the majority of other approaches such kinds of analysis are not supported and, if provided (see for example [17]), they require extra-efforts for the inclusion of some scripts buried in the page code that are able to generate *ad hoc* log data. In our approach, the production of conceptual logs does not require any additional activity during the application development. Thanks to the model-based approach, at runtime the application engine instantiates elements of the conceptual schema. Therefore, it is also able to "naturally" log

execution data that reflect the defined conceptual schema. Also, the logged data have minimal size and their calculation does not impact sensibly the overall runtime performance.

Thanks to the intensive use of the application conceptual schema, our framework also introduces a number of advantages with respect to Web usage mining. Several data mining projects have demonstrated the usefulness of a representation of the structure and content organization of a Web application [14, 15, 25]. As affirmed in [12], “the description of the application structure is considered a critical input to the pre-processing algorithms that can be used as filter before and after pattern discovery algorithms, and can provide information about expected user behaviors”. However, Web usage mining approaches often require additional, sometimes complex, computations for reconstructing the application schema [12, 16].

As shown by this paper, the conceptual logs, which characterizes our approach, can be easily tailored on specific mining algorithms used in Web Usage Mining, with the great advantage of eliminating the typical Web Usage Mining preprocessing phase completely. In fact, according to our approach: (i) *Data cleaning* is mainly encapsulated by the Log Synchronizer module; (ii) *the identification of user sessions* is done by the WebML runtime; (iii) *post-mining retrieval* of content and structure information is unnecessary since this information is available from the WebML conceptual schema, and is also integrated into the final conceptual logs.

7 Conclusions

The ever-increasing spread of the Web asks for new methods for improving the quality of Web applications. Most current Web applications are centered on large sets of data and require very complex hypertext structures that very often are difficult to understand by users and do not meet their requirements. Quality enforcement has been so far pursued mostly by addressing the application analysis and design with the help of structured development methods, possibly based on conceptual models of Web applications. However, although conceptual modelling does improve the final quality of the application by fostering regularity and the definition and reuse of effective design patterns, a gap exists between the model-driven analysis and design phases and the application maintenance and evolution phases, where most of the quality evaluation activities and of the corrective actions are performed thanks to the implicit or explicit feedback of “real” users. In particular, the conceptual schemas developed in the “upper” part of the application life cycle are not used in the post-delivery phases.

This paper has proposed a framework for quality evaluation based on the integration of the design-time conceptual schemas of the application and the usage data collected at runtime. The original features of the proposed technique and tool can be summarized as follows:

- Web usage data are expressed in the same vocabulary as the conceptual models (e.g., they refer to entities, relationships, content units, pages, areas, and so on). This eliminates the impendence mismatch between log data and design documents, which occurs when conventional, low-level log analysis tools are used, and even permits to display Web usage statistics directly on the conceptual design diagrams.
- Thanks to the ubiquitous use of XML and XSL, the quality evaluation framework is very flexible and extensible: new evaluation queries and measures can be easily specified by means of XSL rules and added to the rule repositories. In this way, each design team can define its own quality criteria, and code their measures within WQA by simply

writing XSL code, an extensively used W3C standard.

- The proposed framework is easily implementable and adapts well to any Web conceptual model. The pre-requisites for its implementation is the capability of logging a few data on the objects and object types involved in the computation of page contents. The required data have minimal size and their calculation does not impact sensibly the overall runtime performance.

Our future work will concentrate on applying the proposed evaluation framework to larger Web applications, so as to further validate our method and tools, and on the incremental enrichment of the quality metrics and statistics based on Web usage data. In its current version, the framework concentrates on the evaluation of two quality factors, namely design correctness and usability of the final application. However, thanks to its flexibility, the framework architecture and the supported analyses can be easily extended, to cover the evaluation of other quality factors, such as performance, availability and security.

We are also working on the definition of a user interface for allowing designer to define new quality evaluation queries and reports, without the need of manual XSL programming.

Bibliography

1. Analog. Analog 5.32: Introduction., 2003. <http://www.analog.cx/docs/Readme.html>.
2. Apache. HTTP Server Project, 2003. <http://httpd.apache.org/docs>.
3. AWS.D. WebLog, 2003. <http://awsd.com/scripts/wevlog/index.shtml>.
4. L. Baresi, F. Garzotto, and P. Paolini. Extending UML for Modeling Web Applications. In *Proc. of HICSS'01, Maui (USA), January 2001*. IEEE Press, January 2001.
5. D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. Lanzi. A Tool for Extracting XML Association Rules. In *Proceedings of ICTAI'02, 4-6 November, Crystal City, USA*. IEEE Computer Society, 2002.
6. CapeCom. WebLogs, 2003. <http://www.cape.com>.
7. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002.
8. S. Ceri, P. Fraternali, A. Bongio, S. Butti, R. Acerbis, M. Tagliasacchi, G. Toffetti, C. Conserva, R. Elli, F. Ciapessoni, and C. Greppi. Architectural Issues and Solutions in the Development of Data-Intensive Web Applications. In *Proceedings of CIDR 2003, January 2003, Asilomar, CA, USA, 2003*.
9. S. Ceri, P. Fraternali, and M. Matera. Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing*, 6th(4):20–30, July-August 2002.
10. S. Comai, M. Matera, and A. Maurino. A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In *Proceeding of the ER'02-IWCMQ'02 Workshop, Tampere, Finland, October 2002*, volume 2784 of *LNCS*, pages 339–350. Springer Verlag, 2002.
11. J. Conallen. *Building Web Applications with UML*. Object Technology Series. Addison Wesley, 2002.
12. R. Cooley. The Use of Web Structures and Content to Identify Subjectively Interesting Web Usage Patterns. *ACM TOIT*, 3(2), May 2003.
13. R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1):5–32, January 1999.
14. R. Cooley, P. Tan, and J. Srivastava. *Discovery of Interesting Usage Patterns from Web Data*. LNCS/Lecture notes in Artificial Intelligence. Springer Verlag, 2000.
15. M. Eirinaki and M. Vazirgiannis. Web Mining for Web Personalization. *ACM TOIT*, 3(1), February 2003.

16. F. Facca and P. Lanzi. Recent Developments in Web Usage Mining Research. In *Proceedings Of DaWaK 2003, Prague, Czech Republic, September 2003*, LNCS. Springer Verlag, 2003.
17. FireClick, 2003. <http://www.fireclick.com>.
18. P. Fraternali, M. Matera, and A. Maurino. WQA: an XSL Framework for Analyzing the Quality of Web Applications. In *Proceedings of IWOST'02 - ECOOP'02 Workshop, Malaga, Spain, June 2002*, 2002.
19. M. Genero, J. Nelson, and G. Poels, editors. *Proceedings of IWCMQ'02 - ER'02 International Workshop on Conceptual Modeling Quality, Tampere, Finland, October 2002*, LNCS. Springer Verlag, 2002.
20. J. Gomez, C. Cachero, and O. Pastor. Conceptual Modeling of Device-Independent Web Applications. *IEEE MultiMedia*, 8(2), March-April 2001.
21. M. Ivory and M. Hearst. The State of the Art in Automating Usability Evaluation. *ACM Computing Surveys*, 33(4):470–516, December 2001.
22. P. Kruchten. *Rational Unified Process. An introduction*. Addison Wesley, 2nd edition, 2000.
23. M. Matera, M. Costabile, F. Garzotto, and P. Paolini. SUE Inspection: an Effective Method for Systematic Usability Evaluation of Hypermedia. *IEEE Trans. on SMC*, 32(1), January 2002.
24. J. Nielsen. *Web Usability*. New Riders, 2000.
25. P. Pirolli, J. Pitkow, and R. Rao. Silk from a Sow's Ear: Extracting Usable Structures from the Web. In *Proceedings of Confernece on Human Factors in Computing Systems CHI'96, April 1996*. ACM Press, April 1996.
26. J. Pitkow. In Search of Reliable Usage Data on the WWW. In *Proc. of 6th International Conference on World Wide Web*, pages 451–463, May 1997.
27. G. Rossi, D. Schwabe, L. Esmeraldo, and F. Lyardet. Engineering Web Applications for Reuse. *IEEE Multimedia*, 8(1):20–31, January 2001.
28. V. Turau. DB2XML, 2003. <http://www.informatik.fh-wiesbaden.de/turau/DB2XML/>.
29. WebRatio. Site Development Studio, 2003. <http://www.webratio.com>.
30. W3C. Extended Common Log File format, 2003. <http://www.w3.org/TR/WD-logfile.html>.

Appendix A. WebML content and operation units

Table 1. Basic WebML content units.

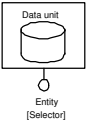
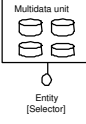
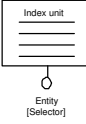
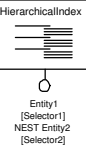
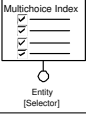
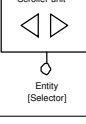

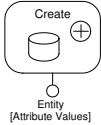
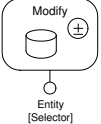
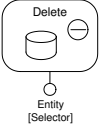
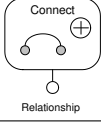
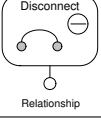
Unit name	Visual Notation	Description
<i>Data unit</i>		It displays a set of attributes for a single entity instance.
<i>Multidata unit</i>		It displays a set of instances for a given entity.
<i>Index unit</i>		It displays a list of properties, also called <i>descriptive keys</i> , of a given set of entity instances, and enables the selection of one single instance.
<i>Hierarchical Index unit</i>		A variant of the index unit, which displays list of properties of instances selected from multiple entities, nested in a multi-level tree.
<i>Multichoice Index unit</i>		A variant of the index unit, in which each element of the list is associated with a checkbox, allowing the user to select multiple instances.
<i>Scroller unit</i>		It represents a scrolling mechanism, based on a block factor, for the elements in a set of instances.
<i>Entry unit</i>		It displays a form for collecting input values into fields.

Table 2. Basic WebML operation units.

Unit name	Visual Notation	Description
<i>Create unit</i>		It creates a new entity instance.
<i>Modify unit</i>		It modifies attributes of an entity instance.
<i>Delete unit</i>		It deletes one or more entity instances.
<i>Connect unit</i>		It creates a relationship instance.
<i>Disconnect unit</i>		It drops a relationship instance.

Appendix B. Example of Sequential Patterns

```

01 <Sequences>
02 <SequenceRule support="0.037096774193548385" confidence="0.11734693877551021"/>
03 <AntecedentSequence>
04 <ItemSet>
05 <Item>
06 <Page SchemaRef="page03" Name="Overview" Area="Overview">
07 <PageContent>
08 <Unit>
09 <UnitSpecs SchemaRef="dau23" Name="TheWebML Model" SourceEntity="WebMLConcept"/>
10 <Data_OID>1</Data_OID>
11 </Unit>
12 <Unit>
13 <UnitSpecs SchemaRef="dau25" Name="The Idea" SourceEntity="TextChunk"/>
14 <Data_OID>15</Data_OID>
15 </Unit>
16 <Unit>
17 <UnitSpecs SchemaRef="inu26" Name="WebML Models" SourceEntity="WebMLConcept"/>
18 <Data_OID>2</Data_OID>
19 <Data_OID>4</Data_OID>
20 <Data_OID>5</Data_OID>
21 <Data_OID>8</Data_OID>
22 </Unit>
23 </PageContent>
24 </Page>
25 </Item>
26 </ItemSet>
27 </AntecedentSequence>
28 <ConsequentSequence>
29 <ItemSet>
30 <Item>
31 <Page SchemaRef="page18" Name="WebML Material" Area="Teaching">
32 <PageContent>
33 <Unit>

```

```
34         <UnitSpecs SchemaRef="dau38" Name="Material Details" SourceEntity="Material"/>
35         <Data_OID>11</Data_OID>
36     </Unit>
37     <Unit>
38         <UnitSpecs SchemaRef="dau40" Name="Material Type" SourceEntity="MaterialType"/>
39         <Data_OID>5</Data_OID>
40     </Unit>
41 </PageContent>
42 </Page>
43 </Item>
44 </ItemSet>
45 </ConsequentSequence>
46 </SequenceRule>
47 ... ..
48 </Sequences>
```