Techniques and Tools for Exploiting Conceptual Modeling in the Evaluation of Web Application Quality

P. Fraternali, P.L. Lanzi, M. Matera, A. Maurino
DEI - Politecnico di Milano
P.zza L. da Vinci, 32 – 20133 – Milano, Italy
{fraterna, lanzi, matera, maurino}@elet.polimi.it

ABSTRACT
Conceptual modeling has proved effective in the rationalization of the upper part of the software lifecycle, thanks to the platform-independent specification of requirements and of design schemes and to the support of automatic code generation. This paper presents an approach and a toolset for exploiting the benefits of conceptual modeling also in the quality evaluation tasks that take place both prior to the deployment and during the operational life of a Web application.

Categories and Subject Descriptors

General Terms
Design, Verification.

Keywords

1. INTRODUCTION
Modern software development methodologies advocate an agile approach to development, 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. This process demands pervasive quality evaluation, for continuously identifying the modifications necessary to cope with changed or new requirements. Since the boundary between the development phase and the operational life of the application 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.

Fast prototyping 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 [5, 18]. However, the conceptual schemas resulting from the design activities are rarely used to support quality assessment [19], due to the lack of model-driven methods and tools.

In this paper we present a comprehensive evaluation framework that supports quality analysis of Web applications both at design time and after the application is deployed. The framework exploits the conceptual schemas produced in the design phase in two complementary ways: i) conceptual schemas are used at design time for evaluating quality a priori, in absence of real usage data and based on quality rules aimed at identifying sources of inconsistency that potentially reduce the application quality; ii) after the application is deployed, rich Web usage logs, also including semantic information concerning the application conceptual schema, are used for evaluating quality a posteriori, verifying if the users follow the navigation paths embodied in the designed hypertext interfaces, or else exploit alternative (sometimes unexpected) access mechanisms, not foreseen by the hypertext designers. Both techniques “speak the same language” because they are based on the application’s conceptual schema, which facilitates the interpretation of the evaluation results and the transformation of such results into corrective actions applied to the conceptual schema of the application, and then to the implementation.

The proposed evaluation framework is presented in the context of a specific Web modeling language [6], and has been implemented by extending a commercial CASE tool [5, 24]. However, we claim that the approach is of general validity, and can be easily adapted to other model-driven Web design methods and CASE 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 logs with information referring to the application conceptual schema.

To field-test the evaluation framework, we have applied it to the webml.org application (http://webml.org), the reference site about the WebML language, which publishes a rich set of resources, including courseware and Web modeling extracts of the WebML book, organized by chapters, downloadable materials, papers, tutorials, and exercises. The site consists of two Web applications:

- A public web site, accessible from unregistered user interested in Web Modeling, the WebML language, and its related activities. This site consists of different sections (Overview, Book, News, Resources, Research, and Industry).
- A content management application, whose access is restricted to administrators only, for gathering,
The paper is organized as follows. Section 2 illustrates some background concepts referring to the WebML method. Section 3 describes the evaluation framework, by illustrating the main features of the a priori and a posteriori analyses. For each introduced analysis, Section 3 also illustrates some excerpts from the webml.org evaluation. Section 4 then illustrates the software architecture of the proposed framework, and the tools we have developed for supporting the automatic analysis of Web applications. Finally, Sections 5 reports on some related work, while Section 6 draws our conclusion.

2. BACKGROUND

This section shortly introduces the basic concepts of WebML, the conceptual design language adopted in this paper to exemplify the quality evaluation framework.

2.1 WebML Models

WebML (Web Modeling Language) is a conceptual model for Web application design [6]. It is an ingredient of a broader Web development methodology, which is supported by a CASE tool, named WebRatio [5, 24].

In WebML the specification of a Web application has two main perspectives:

- The information assets managed by the application, their properties and logical correlations.
- The hypertexts of the application, consisting of pages for publishing content and of links for navigating and performing actions.

For expressing the organization of data, WebML exploits existing 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 consists of entities, defined as containers of data elements, and relationships, defined as semantic connections between entities. Entities have named properties, called attributes, with an associated type. Entities can be organized in generalization hierarchies, and relationships can be restricted by means of cardinality constraints.

Beside the data model, WebML allows developers to describe one or more hypertexts for publishing and managing content. The general structure of a hypertext is described in terms of site views, areas and pages. A site view is a specific hypertext, designed for a particular class of users. For example, in an e-commerce application a public site view may serve the Internet customers, while a number of private site views may be targeted to different stakeholders, like product managers, marketing managers, administrators, and so on.

Site views may exhibit an internal organization, represented with the concept of area, defined as a hypertext module, which in turn can be organized into sub-areas. Site views and areas are then composed of pages that are containers of elementary piece of contents. The content of each page and the navigational commands for moving among pages are specified using the notions of content unit and link. A content unit is a component for the publication of information inside a page. 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 data schema and the hypertext is represented by the source entity and the selector of the content 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 six predefined units (data, index, multidata, scroller, multichoice index, and hierarchical index).

To compute its content, a unit may require the “cooperation” of other units, and the interaction of the user. Therefore, each WebML unit exposes an input interface, specifying the parameters that it can accept, and an output interface, defining the parameters that the unit can provide to other units. 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.

WebML also provides a unit (called entry unit) representing entry forms for inputting data. An entry unit contains a set of fields whereby the user can input different types of data (texts, numbers, images, files, etc.).

Finally, WebML also models the execution of arbitrary business actions, by means of operation units. An operation unit is placed outside the pages, and 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.

Figure 1 shows a WebML schema and a possible rendition. The Modify Paper Data page contains the Papers Index unit that shows the list of all the available papers. The link departing from the index unit allows users to select one paper from the list. The link transports the identifier of the selected paper, which is used by the entry unit Insert new data for retrieving and showing the paper attributes within the form fields. User can then insert new data and follow the link departing from the entry unit (rendered as a submit button) for activating the Modify Paper operation.
As represented by the KO link originating from the operation, if for any reason the operation fails, the application shows again the Modify Paper Data page; otherwise, as represented by the OK link, the user is provided with the Paper Details page, where the Paper data unit shows the newly modified paper data.

Besides having a visual representation, WebML primitives are also provided with an XML-based textual representation, which allows specifying additional detailed properties, not conveniently expressible in the graphic notation. For further details on WebML and its formal definition, the reader is referred to [6].

2.2 Systematic Design through WebML Sub-Schemas

Data publishing and content management on the Web typically exhibit some regularity, which can be exploited in the design of Web applications. The WebML design method therefore proposes a sequence of steps for assembling the data schema and the hypertext schema, which assume that Web applications can be abstracted as complex arrangements of elementary sub-schemas [6, 7], i.e., pairs of data diagrams (describing portions of the data schema), and hypertext diagrams (describing composition of pages and navigational patterns).

The essence of the proposed method is the classification of the role that information concepts play within the Web application:

- **Core Concepts** form the main asset and express the mission of the Web application.
- **Access Concepts** support the categorization of core concepts, for facilitating access to their instances.
- **Interconnection Concepts** interconnect core concepts, and are typically expressed by means of associations between information objects in the E/R diagram.

Such a classification helps building the hypertext schema, which can be assembled using design modules that represent “canonical” configurations of units and links built on top of the previously identified data sub-schemas. In this way, designers build new applications by reusing common design experiences [23], which normally results into more consistent schemas.

Recognizing sub-schemas also helps the evaluation phase, because it provides a systematic way for characterizing the targets of evaluation, which are the core entities of the data schema, the access paths leading to them, and the interconnection paths for moving among them. These targets indeed form the application backbone, and therefore their quality is worth to be investigated.

2.3 Implementation of WebML Applications

WebML is supported by the WebRatio CASE tool [24], which offers a visual environment for drawing the WebML conceptual schemas. The core component of WebRatio is a code generator, based on XML and XSL, which automatically generates the application code from the XML specification. Specifically, it 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 application front-end.

The generated applications run in a standard runtime framework on top of ASP.NET and Java 2 application servers, and have a flexible, service-based architecture allowing components customisation. In particular, the logging service can be extended with user-defined modules to log the desired data.

2.3.1 WebML Development of the Running Case

The data schema of webml.org is centred 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., a book chapter) in different ways.

The entities and relationships of the conceptual data schema have been mapped on top of Postgres database. Pages of the hypertext schema have been translated into 182 JSP 1.1 templates. The application has been deployed on a Java 2 platform.

3. THE EVALUATION FRAMEWORK

The proposed evaluation framework supports three kinds of analysis.

- **Design Schema Analysis** (DSA) verifies at design time the internal coherence of specifications [8, 15]. The aim is to enhance the final application quality through the improvement of schema quality.
- **Web Usage Analysis** (WUA) operates on semantically enriched log data collected at runtime, and produces reports on content access and navigation paths followed by users.
Web Usage Mining (WUM) operates on the same enriched log data as WUA, and applies mining techniques for discovering interesting (sometimes unexpected) associations between accessed data.

In the next subsections we illustrate these different analysis approaches, by showing some of the results of applying each of them to the running case. In Section 4 we then present the architecture and tools that we have implemented to automatically support the analysis.

### 3.1 Design Schema Analysis

DSA looks for design inconsistencies and lacks, by discovering and analyzing some hypertext portions, called *evaluation patterns*, which are potential source of problems with respect to the attributes of a quality model we have defined for hypertext schemas. More details on the quality model underlying DSA can be found in [8].

#### 3.1.1 Excerpts of DSA in the Running Case

**Analysis Task.** We have applied DSA to the webml.org content management application, for verifying the consistency of content management operations.

**Results.** Figure 2 shows the visualization of results for the analysis of consistency applied over delete operations. Such operations are used in the hypertext schema for specifying the deletion of some information objects. The bar chart shows that:

- **Web Usage Mining (WUM)** operates on the same enriched log data as WUA, and applies mining techniques for discovering interesting (sometimes unexpected) associations between accessed data.

In the next subsections we illustrate these different analysis approaches, by showing some of the results of applying each of them to the running case. In Section 4 we then present the architecture and tools that we have implemented to automatically support the analysis.

### 3.2 Web Usage Analysis

WUA operates on log data collected at runtime, and produces quality reports on user content access and navigation paths. The main objective is verifying if the hypertext design and the way in which users browse contents are mutually consistent.

The peculiarity of our approach is the exploitation of so-called *conceptual logs*, defined as XML-based "enriched" Web logs that integrate the conventional HTTP log data, logged by Web servers in ECLF (Extended Common Log Format) format [26], and meta-data related
The latter data are recorded by a logging plug-in of the application runtime engine, and include: 1) information about the WebML content units composing the accessed pages; 2) information about the entity instances used for populating the content units inside the accessed pages.

Figure 3 reports an extract of conceptual logs, related to the request of a page (page33) in the webml.org Web application. The main peculiarity is that for each requested page, the <PageUnits> tag includes data about the computation of content units composing the page, as defined in the hypertext conceptual schema. In particular, for each unit:

- the <Unit_ID> tag delimit the unit identifiers, which is the same value that univocally identifies the unit within the hypertext conceptual schema;
- the <DataInstance> tags delimit the primary keys of the database instances used at runtime to populate content units.

Note that the attribute SchemaRef for pages and content units represents references to the definition of such elements within the conceptual schema. During analysis, such references permit to retrieve additional properties, not traced by the logging mechanism, but represented in the conceptual schema, e.g., the source entity of each content unit, or also the areas and site views that contain a given page.

Based on conceptual logs, WUA comes in two flavors: Access Analysis, which focuses on the frequency of access to elements of the data schema and of the hypertext schema, and Navigation Analysis, which focuses on the frequency of use of access and interconnection paths for browsing core concepts.

3.2.1 Access Analysis
Access Analysis computes traffic statistics, with the aim of verifying if the communication goals for which the Web application has been designed are efficiently supported by the hypertext interface. The model-based approach, which distinguishes between data modelling and hypertext modeling, allows performing:

- Data Access Analysis: it computes access statistics related to entities and their instances, for responding 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, which focuses on the usage of the interface elements (pages, areas, and site views), for responding to such questions as "Which is the most/least frequently used page for displaying the content of Entity X?" or "Which is the most/least frequently used page for displaying a specific instance of Entity X?".

It therefore results that our access analysis extends the statistics normally offered by state-of-the-practice traffic analyzers, which mostly address page visits and do not log database instances used for populating dynamic pages. In order to achieve the same results, such tools would require the extension of page code through scripts for tracking contents displayed by each page (see for example [14]). Conversely, our model-based approach does not require any additional effort during the implementation phase.

3.2.1.1 Excerpt of Access Analysis in the Running Case

ANALYSIS TASK. We have applied Access Analysis over webml.org conceptual logs, for computing statistics about the most accessed entity, the distribution of accesses along the entity instances, and the distribution of accesses along the different pages publishing the most accessed instance.

RESULTS. Figure 4 shows a graph generated by analyzing conceptual logs collected in March 2003. It shows that the most accessed entity is the one that represents the WebML book chapters (called Chapter), which gets 30% of the user's requests. By clicking on an entity, it is then possible to drill down, and see the distribution of accesses among the various entity instances. In the specific case, chapter TOC (Table Of Content) is the most accessed one.

Hypertext Access Analysis permits the further deepening of the analysis. For example, Figure 5 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 a page of a given chapter in PDF format.

**Corrective Action.** Based on the previous results, designers might accommodate the user preferences, facilitating the access to the most requested data. For example 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.

### 3.2.2 Navigation Analysis

Navigation Analysis verifies if the hypertext topology supports content accessibility, by reconstructing the navigation paths adopted by the users for reaching the core application content [20].

The evaluators choose the units publishing data about core entities they are interested in. Based on this input, Navigation Analysis reconstructs *access paths*, defined over access entities categorizing the core entities, and *interconnection paths*, defined over relationships interconnecting the core entities. The comparison with the conceptual schema then permits to highlight deviations between navigation paths mostly accessed by the users and the paths provided by the designer.

**Results.** Figure 7 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:

- 39% of times (138 out of 350) users have chosen the sequence of navigation steps defined on paper categories (path 1).
- 32% of times (109 out of 350) users have chosen the access path through the hierarchical index unit, which directly leads to the PaperDetails data unit (path 2).
Figure 7. Example of results for navigation analysis.

- 29% of times (103 out of 350), after reaching the PaperDetails data unit, users resort to the back button of the browser [20] 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.

CORRECTIVE ACTION. The previous statistics about the adopted access paths suggest the duplication in the WebML Paper page of the index of papers, as a solution for reducing the use of the back button and the number of navigation steps required to access a paper.

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.

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 weml.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>.

With the aim of facilitating their interpretation, extracted rules are also enriched with adjunctive properties, e.g., the name and the source entity for each unit, retrieved in the application conceptual schema by means of the SchemaRef attributes, originally present in the conceptual logs.

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.

- The analysis of the user navigation sequences for accessing core contents, implemented 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 core concepts.

3.3.1 Web Usage Mining in the Running Case

ANALYSIS TASK. We have applied the above-mentioned mining tasks over the weml.org conceptual logs. In particular, one of the tasks has consisted of mining navigation sequences leading to core contents.

RESULTS. 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 4-28), introducing general concepts about WebML, and (ii) the access to page WebML Material (lines 29-46), publishing a tutorial on WebML, as indicated by the value 11 for the element DataInstance (line 36) for the unit Material
Details (line 34), having Material as source entity\(^1\) (line 35). 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 will also access the WebML tutorial in one of the following navigation steps. Most important, the pattern confidence is 0.12 (line 3), showing that, if users access the Overview page, there is an estimated probability of 12% that an access to the tutorial page will be also found in the next page requests.

**Corrective Action.** 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. For this reason, this finding suggests two possible ways of reconstructing page Overview: (i) the page should 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.

### 3.4 Methodological Remarks

The three kinds of analysis form an evaluation framework that is centered on the existence of an application schema, expressed by means of a conceptual model, and exploits the schema knowledge in different manners.

Starting from a quality model, such as the one reported in [8], DSA and WUA aim at identifying violations of quality attributes that decrease the usability of the final application. The two analyses use a top-down, goal-directed paradigm, in which the designers have in mind precise analysis tasks, and use the application schema for formulating well targeted queries. In particular, the DSA directly elaborates the conceptual schema for assessing some properties. The WUA applies over conceptual logs, and still uses the schema knowledge because the advanced logging mechanism, provided by the application runtime engine, extends Web logs with meta-data related to the application conceptual schema. This last feature allows computing statistics along different dimensions - those suggested by the adopted modeling method - not only limited to page accesses.

WUM is used for discovering user behaviors not foreseen by the application designers, which can be the symptom of design lacks, not necessarily errors. The aim is to identify possible amendments for accommodating user needs. For this reason, WUM follows a bottom-up inductive approach. It applies over conceptual logs for extracting associations, such as those between accessed pages or data. In order to specify mining queries, evaluators do not need to know the application schema; they just activate tasks for extracting associations among any logged element. Conceptual schema is then used for better interpreting the mined associations.

![Figure 8. The evaluation framework architecture.](image)

### 4. Architectural Issues

Figure 8 illustrates the architecture of the WebML quality evaluation framework. It is based on three layers: Data Extraction, Analysis, and Result Visualization. Two different warehouses are placed among layers:

- The Analysis Data Warehouse stores data needed for analysis, represented in XML format.
- The Result Warehouse stores the results produced by the analysis in XML format. Such data are then used by the graphical user interface for result visualization.

Moreover, the Analysis Tasks repository stores the analysis procedures, which can be expressed both in XSL and XQuery.

It is worth noting that the choice of XML for data representation improves the number of strategies the designer can adopt in order to manipulate and query data. Also, the use of warehouses between layers improves the framework extensibility, since it is possible to add new software modules in a given layer, without affecting other components.

---

\(^1\) In the application data source, the instance of the entity Material with primary key = 11 corresponds to the WebML tutorial.
4.1 Data Extraction Layer

The input to the framework consists of:

- The XML representation of the application conceptual schema, automatically generated by the model-based design tool.
- Web Server access logs in the ECLF format [26].
- The execution data, logged by the application runtime engine.
- The application data source, whose knowledge within the evaluation framework is needed for interpreting the identifiers of data instances, logged by the runtime engine.

The Data Extraction Layer is in charge of gathering data from the previous sources, and importing them into the Analysis Data Warehouse, after performing three main actions:

- Restructuring WebML schemas, i.e., eliminating non necessary information, such as comments and specific implementation details.
- Synchronizing data contained in the Web Server log and in the runtime engine log, thus obtaining the conceptual log. More details on the generation of conceptual logs can be found in [16].

4.2 Analysis Layer

The Analysis layer represents the core of our architecture: it computes quality evaluation results from data stored in the Analysis Data Warehouse. The layer is composed of three software components that implement the three types of analysis described in Section 3.

4.2.1 Design Schema Analysis

The DSA module implements the automatic analysis over the XML representation of hypertext conceptual schemas. It exploits two sets of XSL rules, stored in the Analysis Tasks repository:

- Evaluation Pattern Description rules, representing the XSL specification of hypertext portions to be retrieved and analysed within the schema;
- Evaluation Method rules, representing the XSL specification of the analysis procedures to be performed over evaluation patterns retrieved with the previous rules.

For more details about the DSA component, the reader is referred to [15].

4.2.2 Web Usage Analysis

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 or interconnection paths navigated by users.

- Statistics Computation rules, representing procedures for computing statistics over the retrieved usage patterns.

4.2.3 Web Usage Mining

The WUM is implemented on top of XMINE, a tool developed for mining interesting relations from native XML documents [4]. 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, stored in the Analysis Tasks repository, 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. Finally, in order to improve their comprehension, the extracted rules are enriched with additional schema properties, not logged by the application engine, by means of transformation that replaces the SchemaRef attributes, originally present in the conceptual logs, with selected properties defined in the application conceptual schema.

4.3 Result Visualization Layer

The last layer of our software architecture is devoted to show graphically the analysis result, through a Java graphical user interface (GUI), developed on top of the analysis modules. Some visualization examples have already been shown in Section 3.

The GUI allows evaluators to select the kind of analysis they want to perform (DSA, WUA, WUM). Then, they select the specific analysis task (a design schema analysis, a log statistics, a mining query), among those stored within the Analysis Tasks repository. The obtained results are then displayed in a format that depends on the chosen analysis task. As shown by the examples provided in the previous sections, DSA and WUA results are shown through graphs, while the filtered mined rules are visualized in XML format.

5. RELATED WORK

Web application quality has been so far pursued mostly by proposing best practices and design principles [21], or by means of structured model-based development methods [3, 9, 18, 22, 23]. Some of these methods also prescribe design schema analysis as a mean for improving the final application quality [17, 19]. However, none of them offer automatic support.
During last years, several methods and tools have been proposed for the analysis of Web logs. The majority of the public and shareware tools (see for example [1, 2, 14, 25]) are however traffic analyzers. As also described in [10, 13], their functionality is limited to producing statistics about traffic, diagnostic, referrers, and clients.

Our Web Usage Analysis allows calculating such traffic statistics, also reconstructing user sessions\(^2\). However, the distinguishing feature of our approach is the computation of advanced statistics, related to database entities and instances, and to hypertext components of any granularity. In other approaches (see for example [1, 2, 25]), such kinds of analysis are not supported at all and, when provided (see for example [14]), they require the inclusion of some scripts buried in the page code, able to generate ad hoc log data.

Several data mining projects have demonstrated the usefulness of a representation of the structure and content organization of a Web application [12]. As reported in [10], “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 [10, 13].

Our mining approach eliminates the typical Web Usage Mining pre-processing phase completely. In fact, activities such as data cleaning, user session identification, and the retrieval of content and structure information are unnecessary, because they are encapsulated within the generation of conceptual logs or because meta-data about the conceptual schema are integrated in conceptual logs.

6. CONCLUSIONS

The ever-increasing spread of the Web asks for new methods for improving the quality of Web applications. Conceptual modeling does improve the quality of the final application by fostering regularity and the definition and reuse of effective design patterns. A gap however exists between the model-driven design phases and the application maintenance and evolution phases, where most of the quality evaluation activities and of the corrective actions are performed. This paper has proposed an approach to quality evaluation of Web applications that tries to fill in the gap, by exploiting conceptual schemas in pre- and post-delivery phases.

A relevant feature of our approach is that, thanks to the adoption of a modular architecture, and to the ubiquitous use of XML technologies, the quality evaluation framework is 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, and code their measures by simply writing XSL or XQuery code, two extensively used W3C standards.

Another peculiarity, which distinguishes our approach, is that the production of conceptual logs does not require extra effort during the application implementation. 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.

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 statistics and mining tasks for analyzing Web usage data. We are also working on the improvement of the graphical user interface; in particular, we are developing an interface module, for allowing designers to define new analysis tasks though a visual paradigm, without the need of manual XSL and XQuery programming. We are finally defining a solution for reconstructing user sessions from log data even when server-side caching strategies are applied to increase the performance of Web applications.

7. REFERENCES


\(^2\) This feature, addressed only by few methods, is possible without any extra effort: the WebML-based application engines use Session IDs for retaining the state of user interaction, and the logging mechanism records them in the Conceptual Log.


8. APPENDIX A. Sequential Pattern Example