Exploiting Conceptual Modeling for Web Application Quality Evaluation

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

This paper presents an approach and a toolset for exploiting the benefits of conceptual modeling in the quality evaluation tasks that take place both before the deployment and during the operational life of a Web application. The full version of the paper is available as a technical report at the address: http://www.elet.polimi.it/upload/fraterna/FLMM2004.pdf

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques– *evolutionary prototyping, Computer-aided software engineering (CASE)*. D.2.4 [**Software Engineering**]: Software/Program Verification – *Validation.*

General Terms

Design, Verification.

Keywords

Conceptual Modeling, Web Application Quality, Web Mining.

1. INTRODUCTION

Conceptual schemas, resulting from the adoption of model-driven methods for the design of Web applications, are rarely used to improve the effectiveness of the quality assessment process. In this paper, we present an evaluation framework that takes advantage of a conceptual modeling method for supporting the usability analysis of Web applications. The conceptual schemas deriving from the design phase are exploited in two complementary ways: *i) before* the application is deployed, for identifying errors and inconsistencies in application design that potentially reduce usability; *ii) after* the application is deployed, for validating usability with respect to the real user experience, by analyzing and mining Web logs enriched with semantic information deriving from the conceptual schema. Both techniques "speak the same language", because they exploit the application's conceptual schema produced by developers in the design phase; this facilitates the interpretation of the quality evaluation results, and the transformation of such results into corrective actions to be applied to the original application design, and then propagated to the implementation with the help of code generation tools.

The proposed evaluation framework has been defined in the context of a specific Web modeling method, WebML [1], and has been implemented by extending a commercial CASE tool [4].

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, e.g., those based on UML. In fact, our approach assumes only: 1) an XML specification of the application, e.g., like the one provided by the XMI encoding of UML diagrams: 2) a customizable logging facility, as granted by the most popular Web runtime frameworks.

WWW 2004, May 17-22, 2004, New York, NY USA.
ACM 1-58113-912-8/04/0005.

2. WEB MODELING LANGUAGE

WebML (Web Modeling Language) is a conceptual model for Web application design [1]. It is an ingredient of a broader Web development methodology, which is supported by a CASE tool, named *WebRatio* [4]. In WebML, the specification of a Web application consists of a *data schema* specified as an Entity-Relationship (E/R) model or as a UML class diagram, and of a *hypertext schema* describing the content of the site in terms of *site views*, *areas* and *pages*. A *site view* is a specific hypertext, designed for a particular class of users (Internet customers, administrators, stakeholders and so on) and it may exhibit a hierarchical organization, represented with the concept of *area*, defined as a recursive hypertext sub-module. Site views and areas are composed of *pages* that display elementary piece of contents. A *content unit* is a component for the publication of information inside a page and it corresponding to a parameterized query over the underlying E/R data model. WebML offers six predefined units (*data*, *index*, *multidata*, *scroller*, *multichoice index*, and *hierarchical index*), which show one ore more database instances. To compute its content, a unit may require the "cooperation" of other units, and the interaction of the user. Making two units interact requires connecting them with a *link*, represented as an oriented arc between a source and a destination unit. The aim of a link is twofold: permitting navigation (intra or inter-page), and enabling the parameter passing from the source to the destination unit. Besides having a visual representation, WebML primitives are also provided with an XML-based textual representation. This facilitates the automatic generation of final applications by means of CASE tools, as well as the automatic analysis of the application quality, as it will be explained in the following sections.

3. THE EVALUATION FRAMEWORK

The proposed evaluation framework supports three kinds of analysis.

- *Design Schema Analysis* (DSA) verifies at design time correctness and consistency of specifications, by finding violations of quality attributes that decrease the effectiveness of the conceptual schema, and thus of the resulting application.
- *Web Usage Analysis* (WUA) operates on *conceptual logs*, i.e., on semantically enriched log data collected at runtime, and produces reports on content access and on the navigation paths followed by users.
- *Web Usage Mining* (WUM) still operates on the conceptual logs, by applying data mining techniques for discovering

interesting user behaviors not foreseen by the application designers, which can prompt for the revision or extension of the application interfaces.

The main feature of the analysis performed over Web logs (both WUA and WUM) is the exploitation of the so-called *conceptual logs* [2]. These are XML-based "enriched" Web logs that integrate the conventional HTTP log data, collected by Web servers in ECLF (Extended Common Log Format) format, and meta-data related to the objects involved in the computation of pages. The latter data are collected by a logging library plugged in the runtime framework, and include information about: 1) the WebML content units composing the accessed pages; 2) the data objects populating the content units inside the accessed pages.

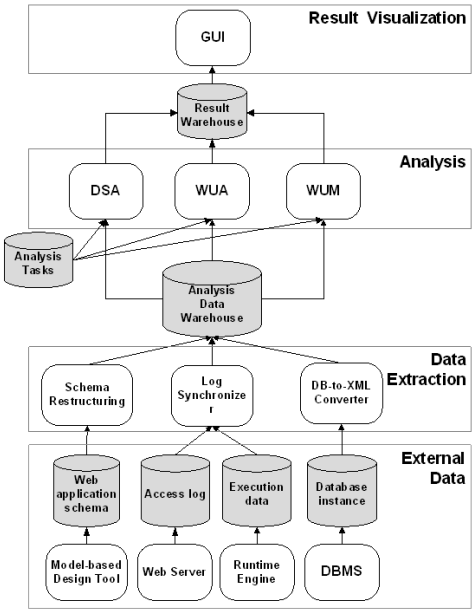

Figure 1. The evaluation framework architecture.

4. ARCHITECTURAL ISSUES

Figure 1 illustrates the architecture of the proposed quality evaluation framework, which is based on three layers: *Data Extraction*, *Analysis*, and *Result Visualization*. Two different warehouses are placed among these layers:

- The *Analysis Data Warehouse* stores data needed for quality analysis, represented in XML format. Such data are produced by the *Data Extraction* layer that elaborates and merges the application conceptual schema, the runtime logs and the application data sources.
- The *Result Warehouse* stores the results produced by the analysis in XML format. Such data are then used by the graphical user interface for visualization.

Moreover, the *Analysis Tasks* repository stores the data analysis procedures, encoded as queries over the mentioned data warehouses, both in XSL and in XQuery. Using XML and XSL/XQuery for expressing data analysis queries makes the quality evaluation framework extensible. Adding a new quality indicator requires adding the suitable warehouse queries for determining the value of the indicator.

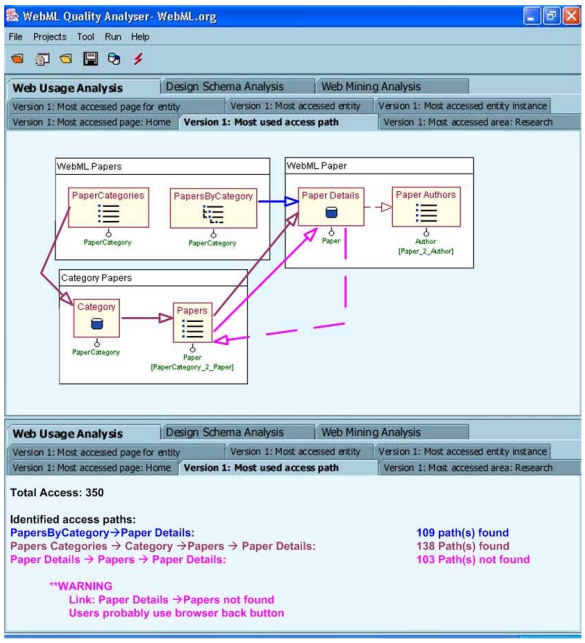

Figure 2 Example of results for navigation analysis.

5. THE QUALITY ASSESSMENT TOOL

The results of the quality analysis are displayed by the graphical tool shown in Figure 2, which highlights the spotted problems directly over the original conceptual model, so that the designer can immediately grasp which part of the application needs intervention and why. For example, Figure 2 shows the reconstruction of user navigation inferred from the conceptual logs, by superimposing the actual navigation paths over the hypertext schema. The analysis highlights an anomalous use of the browser's back button and suggests the addition of an explicit navigation link in the conceptual design.

The full version of the paper [2] illustrates the result of applying the quality evaluation method to a real-life Web application and comments on the different quality checks that the method supports and on the way in which corrective actions for improving the usability of the application can be inferred from the results of the three different classes of analysis described in Section 3.

6. REFERENCES

[1] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002.

[2] P. Fraternali, P. Lanzi, M. Matera and A. Maurino. Techniques and tools for exploiting conceptual modeling in the evaluation of web application quality. Technical Report, http://www.elet.polimi.it/upload/fraterna/FLMM2004.pdf

[3] P. Fraternali, M.Matera and A. Maurino. Conceptual-Level Log Analysis for the Evaluation of Web Application Quality. LA-WEB 2003: 46-57, IEEE Press, 2003.

[4] WebRatio. http://www.webratio.com