Tool Support for Model Checking of Web application designs*

Marco Brambilla[1], Jordi Cabot[2] and Nathalie Moreno[3]

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. Da Vinci, 32. I20133 Milano, Italy
mbrambil@elet.polimi.it
[2] Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya
Rbla. Poblenou 156. E08018 Barcelona, Spain
jcabot@uoc.edu
[3] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Málaga
Complejo Tecnológicio, Campus de Teatinos. E29071 Málaga, Spain
vergara@lcc.uma.es

Abstract: In this work we report our experience in applying model checking techniques to the analysis of static and dynamic properties of Web application models. We propose a mix of tools that facilitate model driven design of Web applications, automatic code generation, and automatic property verification. As recommended by current tendencies in the academic field, we bridge the gap between the tools by devising a set of MDA transformations between the different models. We show that such approach is feasible although we also highlight how current state-of-the-art industrial tools are still partially inadequate for providing seamless support to MDA approaches for industrial Web applications.

1. Introduction

The design of a web application involves the definition of a data model (to specify the data used by the application), a hypertext model (to describe the organization of the front-end interface) and a presentation model (to personalize its graphical aspect).

In industrial web applications, data and hypertext models easily become huge and very complex. Consequently, their definition is an inherently error prone process yet their correctness is especially important when web designs are used to automatically derive the implementation of the web application (as in [4]).

Unfortunately, support for web designs verification is rather limited. Common tools and methods for web application development present poor verification facilities, mainly purely syntax consistency analysis, as the reachability of all pages

* This work has been partial supported by the Italian grant FAR N. 4412/ICT, the Spanish-Italian integrated action HI2006-0208, the grant BE 00062 (Catalan Government) and the Spanish Research Projects TIN2005-09405-02-01 and TIN2005-06053.

from the home page or the existence in the data model of all entity types and attributes referred in the hypertext model.

A complete verification of web designs requires more formal model checking techniques. Currently, there are very few formal verifiers for web applications. Furthermore, using them properly requires deep knowledge of temporal logic (model checking techniques formalize the design to be verified as a finite state machine while the properties to check are expressed in temporal logic). The syntax and semantics of these concepts are completely alien to web designers, who are used to think in terms of pages, links, forms, and so on. This impairs the applicability of model checking tools and prevents their wide adoption in industrial web development projects.

We believe that current state-of-the-art technologies and tools makes now feasible to align Web application modelling and formal model checking so that we can get the best of both worlds: the usability and expressivity of graphical web modelling languages together with the verification capabilities of model checking tools. Clearly, such alignment could boost the adoption of model driven web development methods in the software industry, cutting down the cost of web application development and, at the same time, improving its quality.

As a first attempt to fully integrate formal verifiers with web modelling tools we have devised an MDA framework for the transformation of web models into the formal models expected by model checking tools. In this sense, the aim of this paper is to report our experience and, based on its results, to analyze the main issues that still remain to be solved in the tools for design and verification of Web applications.

Our framework comprises a set of tools that support several standards defined by the OMG. These standards include UML (Unified Modeling Language), MOF (Meta-Object Facility), XMI (XML Metadata Interchange), and MOF/QVT (Query/View/Transformations), among others. The choice of this transformation framework brings a set of benefits with respect to other more direct transformations (as XSLT or Java implementations): (1) it overcomes the limitations on the complexity of the XSLT transformations; (2) it provides the flexibility (reusability) to easily include other types of hypertext models in the framework and (3) it brings the possibility of formally verifying the transformation process (since in our approach both the source and target languages as well as the own transformation are specified by means of formally specified metamodels)...

The rest of the paper is structured as follows. Sections 2 and 3 present Wave and WebML, respectively. Section 4 shows the set of tools we combine to support our approach for web development. Finally section 5 draws some conclusions and outlines future work.

2. Wave: A Web Application Verifier

Wave [6] is a tool in Java for verifying temporal properties of data-driven Web applications. Wave takes as inputs the specification of a Web application (written in a declarative, rule-based specification textual language) and a property (expressed in first-order temporal logic), and outputs whether the property is true or false for that

Web application together with useful verification information (such as a counter example in case of a false property).

A property is true when all possible runs (i.e. run-time executions) of the web application satisfy the property. In this sense, Wave goes beyond purely syntactic analysis and permits to verify all common temporal verification properties (e.g. see [7]). As an example, with Wave we can check that before shipping an order the user has paid that same order in some previous web page. Instead, with a syntactic analysis we can just check that the shipping page is not reached before the payment page; we cannot ensure that the shipped order is the order paid in a previous payment page.

3. WebML

WebML [12][4] is a high-level notation for data-, service-, and process- centric Web applications. It allows specifying the data model of a Web application and one or more hypertext models (e.g., for different types of users or for different publishing devices) used to publish and manipulate the underlying data.

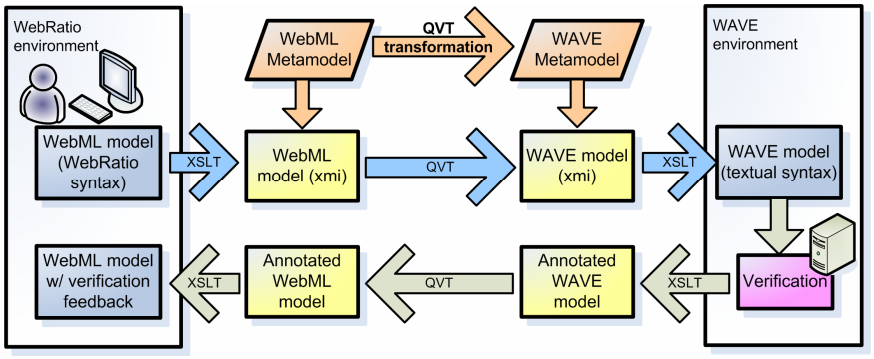
Each hypertext is called site view and is defined as a graph of pages, consisting of connected units, representing at a conceptual level the primitives for publishing contents into pages: the content displayed by a unit is extracted from an entity type, possibly filtered. Units are connected by links carrying data from a unit to another, to allow computation of the hypertext and definition of navigational paths for users. Hypertext models also include content-management units to specify modification operations (insert/update/delete) on the data underlying the site.

WebML is supported by the CASE tool called WebRatio [13], which provides facilities for designing WebML models and for automatically generating the running code of the application. However, WebRatio provides no support for the verification of generic user-defined properties for the hypertext. The only checking that is performed is related to the correctness of the Web application with respect to the WebML syntax and semantics.

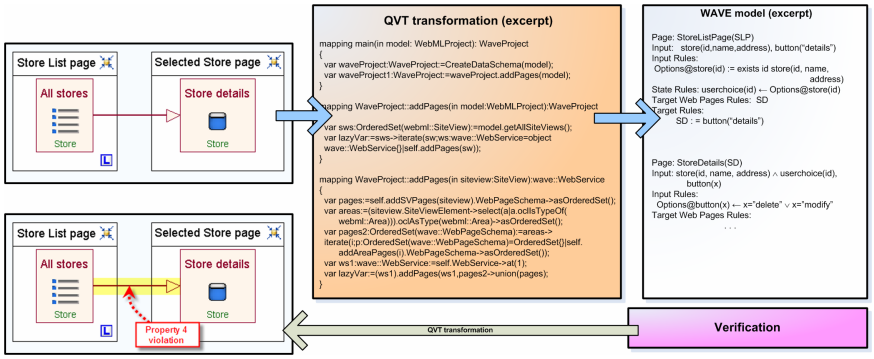
4. Tool Framework

Our formal verification of web designs is implemented through collaboration among several tools. In this section we comment the different steps of the web development process when using our framework and describe the tools employed in each step. An overview of the whole process is shown in Fig. 1.The upper part (Fig. 1a), shows the interactions among the tools and their input and output models. The bottom part (Fig. 1b) sketches the application of the framework over a very simple WebML model.

Step 1. *Specification of the web application models with the WebRatio CASE tool* [13]. In our approach, the design of a web application starts with the specification of the data and hypertext models in WebML [4] using WebRatio. WebRatio stores all specified models in a single XML file. Bottom left part of Fig. 1 shows a simple



(a)



(b)

Fig. 1. Overall view of the proposed framework and its application over an example model

hypertext model comprising a page with an index (*All stores*) and a page with the details of the selected object (*Store details*), connected by a link.

Step 2. *Transformation of the WebRatio syntax into an XMI representation*. The XML file exported by WebRatio must be transformed into an XMI representation [9] to fit into the MDA transformation framework. Such transformation is implemented through XSLT rules. The XML Schema of the XMI representation follows the structure of the WebML metamodel (see the next step).

Step 3. *QVT-Transformation of the WebML models into Wave models*. This step bridges the gap between WebRatio and the Wave model checker. The transformation is formally defined as model-to-model transformation [5], with rules specified using the standard QVT-Operational language [10]. The execution of the transformation is performed with the help of the Borland Together Architect tool [3]. Together supports the QVT-Operational language in the rule specification. Given a transformation definition and an input model, Together produces the output model according to the specified transformation rules. Before applying the transformations in Together, we need to previously perform the following tasks:

- Definition of new metamodels for the WebML and Wave languages. Model-to-model transformations assume the existence of a metamodel (in particular a MOF metamodel [11]) for the initial and target languages of the transformation. Based on the concrete syntax of both languages we have developed and integrated in Together new WebML and Wave metamodels. The WebML metamodel we use is a slightly revised version of the one presented in [8].
- Definition of the transformation rules to map WebML concepts to Wave concepts. Each rule transforms a WebML element into its corresponding Wave element.

In Fig. 1b we show a small transformation excerpt and the Wave model resulting from applying it over the initial WebML model.

Step 4. *Transformation of the Wave XMI representation into the Wave textual syntax*. As a result of the previous transformation, Together stores the generated Wave model in an XMI file. The structure of this XMI file is compliant with the (Together) Wave metamodel. In order to load the model into the Wave tool we must first transform this XMI representation into the (textual) syntax expected by Wave.

Step 5. *Model checking of the resulting Wave model.* Once the model is loaded in Wave, the designer can start checking that the model verifies the properties he/she is interested in. Such properties must be specified in LTL (Linear Temporal Logic). To facilitate their definition we developed a tool [2] that provides a visual notation to express LTL formulas. Then, these formulas are translated into LTL textual syntax.

Step 6. *Improvement of the WebML models according to the verification feedback*. If the verification determines that some properties are invalid, the designer can refine the WebML model to solve the issues. Obviously, the designer must be able to understand the feedback. To this purpose, the feedback should be expressed in terms of the elements of the original WebML models and not in terms of the generated formal Wave models. This can be achieved through reverse model transformations that highlight and annotate the original WebML models, to make the issues evident into the WebRatio tool. Therefore, the designer will see annotations on the WebML elements corresponding to the Wave elements violating the verified properties. In Fig. 1b, *Property 4* appears to be violated by the link connecting the two WebML units.

5. Conclusions and further work

This work presented a framework for the design and verification of Web applications, based on the WebML visual notation and on the Wave formal language and verifier. We described the transformation between them and we discussed how the framework could exploit the benefits of both languages. Indeed, the usability and readability advantages of WebML are coupled with the formal specification and verifiability properties provided by Wave. Thanks to Wave, general properties specified by the user can be checked, included rules on the execution and on the navigation of the user. Refer to [1] for additional material (as the full transformation rules, the metamodels description, etc.) regarding this tool framework.

As mentioned in the introduction, our approach is based on a set of tools and standards defined by the OMG. However, several issues were encountered in using these OMG standards: poor support for QVT transformations in current CASE tools; compatibility problems between the different XMI representations expected by each tool (i.e., UML tool vendors fail to generate fully XMI-compliant specifications of the models); a transformation architecture more complex than we expected at the beginning (a lot of steps are necessary to go from the xml generated by WebRatio to the textual input of Wave), partially due to incompatibilities between the tools.

From our experience with this framework, we draw two main conclusions: (1) it is feasible to align web modelling languages and formal verifiers; and (2) much effort is still necessary to simplify the whole process. Otherwise, it will be difficult to foster the use of our framework (or similar ones) among industry partners, despite its clear benefits to improve the web development process. Fortunately, tool support for the OMG standards is continuously growing, so we can expect than some of the drawbacks encountered will be lessened in a near future.

Further work goes in the direction of improving the usability of the framework, by means of a seamlessly integration with the WebRatio tool in a way that allows the designer to be unaware of the intermediate transformations, thanks to proper visual interfaces for specifying the properties to be checked and for presenting the obtained results directly within the CASE tool. Moreover, predefined properties should be provided to the designer in the form of patterns that when applied over a concrete hypertext model, generate automatically the appropriate LTL formulae for that model. Another approach for simplifying the property definition is to allow visual specification of properties too, like in the solution presented in [2].

References

[1] M. Brambilla, J. Cabot, N. Moreno. Tool Support for Model Checking of Web Application Designs. http://www.elet.polimi.it/upload/mbrambil/webmlwave, 2007.

[2] M. Brambilla, A. Deutsch, L. Sui, V. Vianu. The Role of Visual Tools in a Web Application Design and Verification Framework: A Visual Notation for LTL Formulae, ICWE'05, LNCS 3579, pp. 557-568, 2005.

[3] Borland Together Architect, http://www.borland.com/us/products/together/, 2007.

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

[5] K. Czarnecki, S. Helsen. Feature-based survey of model transformation approaches, IBM Systems Journal, vol. 5, n. 3, 2006.

[6] A. Deutsch , M. Marcus , L. Sui , V. Vianu , D. Zhou. A Verifier for Interactive, Data-driven Web Applications, SIGMOD'05, 2005.

[7] G. Holzmann. The Spin Model Checker Primer and Ref. Manual. Addison-Wesley, 2003

[8] N. Moreno, P. Fraternali, A. Vallecillo. A UML 2.0 Profile for WebML Modeling, Model-Driven Web Engineering, ICWE'06 Workshops, 2006.

[9] OMG. XML Metadata Interchange (XMI) Specification v.2.0. (formal/03-05-02), 2003.

[10] OMG. MOF QVT. Final Adopted Specification (ptc/05-11-01), 2005.

[11] OMG. Meta Object Facility (MOF) Core Specification, (formal/06-01-01), 2006.

[12] WebML.org. http://www.webml.org , 2007.

[13] WebRatio 4.3. http://www.webratio.com/, 2007.