

# Tools for Modeling and Generating Safe Interface Interactions in Web Applications

Marco Brambilla<sup>1</sup>, Jordi Cabot<sup>2</sup>, and Michael Grossniklaus<sup>1</sup>

<sup>1</sup> Politecnico di Milano, Dipartimento di Elettronica e Informazione  
P.za L. Da Vinci, 32. I-20133 Milano - Italy  
{mbrambill|grossniklaus}@polimi.it  
<sup>2</sup> INRIA - École des Mines de Nantes  
Rue Alfred Kastler, 4 B.P. 20722 - F-44307 NANTES Cedex 3 - France  
jordi.cabot@inria.fr

**Abstract.** Current Web applications embed sophisticated user interfaces and business logic. The original interaction paradigm of the Web with static content pages that are browsed by hyperlinks is, therefore, not valid anymore. We advocate a paradigm shift for browsers and Web applications, considering new kinds of events, side effects, and asynchronous interactions. *Pages* are replaced by *States*, and *Back/Forward* navigation along the browsing history is replaced by a full-fledged interactive application paradigm, featuring *Undo/Redo* capabilities, with support for exception management policies and transactional properties. This new paradigm offers a safer and more precise interaction model protecting the user from unexpected behaviours.

## 1 Introduction

The Web has evolved from a platform for navigating hypertext documents to a platform for implementing complex business applications, where user interaction relies on richer interaction paradigms (RIAs, AJAX), that include partial page updates, asynchronous interactions, and several kinds of events, generated by both users and systems. In this context, the original interaction paradigm of the Web, based on a simple navigation approach of moving from one page to another is too simplistic. Browsers themselves, that still provide the traditional features of *Back* and *Forward* page navigation along the browsing history, are inadequate for dealing with the complexity of current applications [1]. For instance, the Web application behaviour is indeterministic with respect to the actions allowed by current browsers. Depending on the browser and on the application, different behaviours can occur when: the user clicks on the Back and Forward buttons in a page where some fields were filled in (sometimes the data is preserved, sometimes it is lost); the user clicks on the Back button when browsing an AJAX application (often he is sent back to the initial state of the whole application); or the user navigates back on a link where a side effect had been triggered (e.g., updating some data, see the *Amazon bug*). The behaviour after exceptions and errors (e.g., session timeout) is also indeterministic.

These issues complicate the modelling of complex Web applications and hamper the user experience. We propose to evolve the interaction paradigm by moving the Web (and the supporting browsers) from the browsing paradigm based on *Pages*, with related *Back* and *Forward* actions, to a full-fledged interactive application paradigm, based on the concept of *State*, that features controlled *Undo* and *Redo* capabilities, exception management, and transactional properties. This results in a safer interaction environment where users can freely navigate through the web application and undo/redo his/her actions without running into unexpected behaviours.

This paper presents a toolset consisting in: 1) a model editor to specify Web application interfaces and 2) a code generator that automatically produces prototypical safe applications from the models by, among other services, automatically computing the next/previous state of the application based on the current state, the user event and the application state machine model, according to the method presented in [2]

## 2 Modelling Safe Interfaces for Web Applications

The first step of the development process is the specification of the interface and behaviour of the Web application. In our proposal Web applications are represented as state machines consisting of *states* (i.e., possible situations the application can be in) and *transitions* (i.e., changes from a state to another, triggered by an event). A single Web page can comprise several states (in that case, pages are modelled as submachines internally composed of substates), depending on the granularity chosen by the designer. Additional modeling primitives allow the definition of exception events and exception states (that help designers model the application response to unexpected situations) and the definition of *transaction* regions, i.e., a set of states that must be successfully accomplished altogether, with all-or-nothing semantics. We also offer a set of predefined kinds of transitions between states (e.g., click button, list selection,...) to facilitate the definition of the state machine.

As an example, figure 1 shows our model editor at work, depicting a model for a two-page web email application: *inbox* page shows an index of all available messages and allows to delete them and to choose one to see its details in *msgView* page. This *msgView* page ... \*\*complete the explanation

The online editor has been implemented as a Rich Internet Application, exploiting the OpenJacob Draw2D and Yahoo! User Interface libraries. The editor allows to save, load, edit, and validate models, and provides a link to the code generation feature that produces running prototypes from the models (see next Section). A beta version of the editor is available online at:

<http://home.dei.polimi.it/mbrambil/safeinterfaces.html>

The modeling language used to describe the state machines is defined by the internal metamodel shown in Figure 2 (white-coloured classes). Our metamodel is based on the the state machines sublanguage of the UML which we

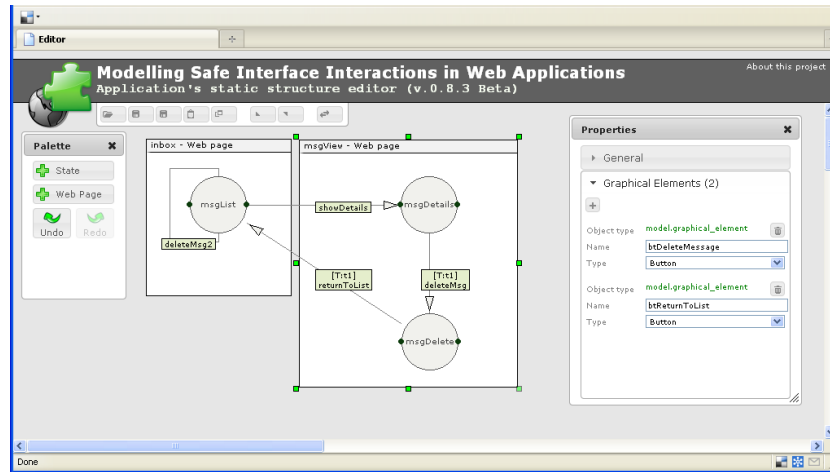


Fig. 1. Example of web interface model drawn with our online model editor.

have adapted to the Web applications domain by adding the concepts of Page, GraphicalElement, Transaction and so forth mentioned above.

### 3 Implementing Safe Web applications: Run-time Support

Our tool support also helps to implement the modeled web application by: 1 - automatically parsing the model information and passed it on to a predefined server component that acts as a controller for the application (MVC architecture); 2 - providing a run-time API that programmers can use to interact with the controller and easily manage all events involving state changes in the application and implement correct state behaviours (including undo and redo features) with little effort.

The data structures internally used by the controller to manage run-time dynamic information (current state the user is in, input parameters, user events) are shown in Figure 2 (grey-coloured classes). For instance, every move of the application user to a state is recorded as a new *Visit*. Obviously, the same user can perform several visits to the same state. The visits trace is permanently stored to allow undo/redo computation.

Both the static and run-time information of the application can be accessed/updated using our predefined API. For space reasons, we only present the main API functions. For instance, methods `getNext` and `getPrevious` can be used by the application developer to query the next or previous visit in the history, respectively. Instead, the `do` and `undo` method are then used to actually perform a move to the next or previous visit and, thus, they manipulate the history records during the process. Note that the `do` method uses the parsed information from the state machine to know the state to go to according to the

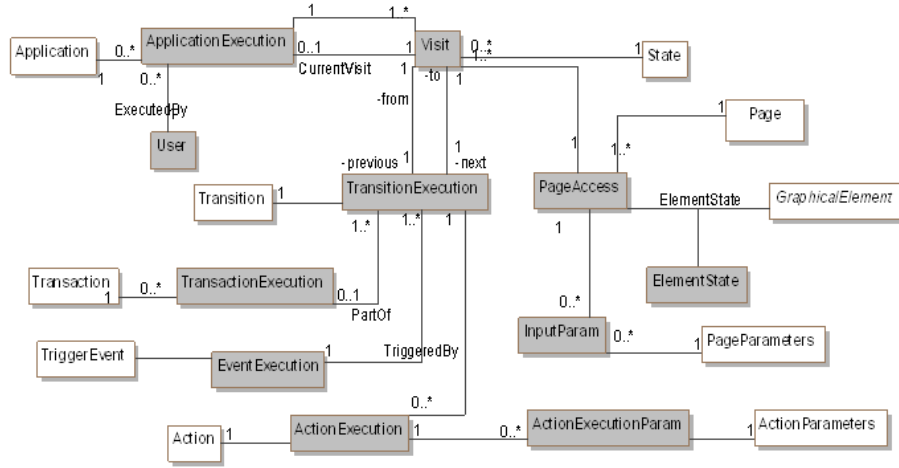


Fig. 2. Interaction metamodel

current user state (information associated to the user’s visit information) and the event triggered by the user. The method *redo* re-visits a state that has already been visited.

## 4 Related Work

Most traditional methods for user interface design are based on state machines. State-based definitions have been used for general user interfaces [3], for hypermedia navigation [4], and for complexity of dynamic Web sites [5]. StateWebCharts [6] extend statecharts with concepts targeted towards the modelling of Web applications. Finally, Draheim and Weber [7] propose the use of bipartite state machines to model both the pages and the server actions.

In the domain of model-driven Web engineering, the specification of RIAs was addressed in [8], in extensions to WebML [9] and to RUX-Model [10], in ADV-Charts [11], and in orchestration models for widgets [12]. Nevertheless, most existing Web design methodologies such as WebML [13], Hera [14], OOHDM [15], etc. do not provide concepts to precisely specify complex application states. While these approaches support the specification of the desired interface behaviour, they do not consider the problems caused by the user-interaction with the Web browser nor do they provide any kind of transactional support. Currently, designers have to identify the potential interaction problems using verification/validation techniques [16] and, then, correct these issues.

Various approaches study the different problems that can occur in the context of back navigation [1, 17, 18]. To address the problems identified by these authors with a generic and MDD method is precisely the goal of our proposal.

**Table 1.** API Methods (Excerpt)

Method	Remarks
<code>State::getNextState(Event e): State</code>	informs about the next state to go based on the current one and the given event
<code>Visit::getNext(): Visit</code>	queries next visit
<code>Visit::getPrevious(): Visit</code>	queries previous visit
<code>ApplicationExecution::do(EventExec e, Parameter[] p): Visit</code>	moves to next visit
<code>ApplicationExecution::redo(): Visit</code>	moves to the (previously visited) next visit
<code>ApplicationExecution::undo(): Visit</code>	moves to previous visit reversing all executed actions
<code>Visit::clone(): Visit</code>	creates a clone of the visit
<code>ActionExecution::do(ActionExecutionParam[] params)</code>	executes the action
<code>ActionExecution::undo(ActionExecutionParam[] params)</code>	undoes the effect of the action
<code>TransitionExecution::undo()</code>	undoes all actions associated to the transition
<code>TransactionExecution::rollback()</code>	rollbacks the transaction

## 5 Conclusion

We have sketched an approach for modeling Web application interfaces using extended state machines. A run-time API supports the implementation of applications and ensures safe and deterministic application behaviour even in the case of exceptions. As future work, we plan to validate the solution in industrial case studies and provide full coverage of transactionality of side effects.

**Acknowledgments.** This work has been supported in part by the Spanish-Italian Integrated Action HI2006-0208, the Catalan Government Grant 2007 BP-A 00128 and the Swiss National Science Foundation Grant PBEZ2-121230.

## References

1. Milic-Frayling, N., Jones, R., Rodden, K., Smyth, G., Blackwell, A., Sommerer, R.: Smartback: Supporting Users in Back Navigation. In: Proc. WWW'04. (2004) 63–71
2. Brambilla, M., Cabot, J., Grossniklaus, M.: Modelling safe interface interactions in web applications. In: Conceptual Modeling - ER 2009. Volume 5829., Springer (2009) 387–400
3. Jacob, R.J.K.: A Specification Language for Direct-Manipulation User Interfaces. ACM Trans. Graph. **5**(4) (1986) 283–317
4. de Oliveira, M.C.F., Turine, M.A.S., Masiero, P.C.: A Statechart-based Model for Hypermedia Applications. ACM Trans. Inf. Syst. **19**(1) (2001) 28–52
5. Leung, K.R., Hui, L.C., Hui, S., Tang, R.W.: Modeling Navigation by Statechart. In: Proc. COMPSAC'00. (2000) 41–47

6. Winckler, M., Palanque, P.: StateWebCharts: A Formal Description Technique Dedicated to Navigation Modelling of Web Applications. In: Proc. Intl. Workshop on Design, Specification and Verification of Interactive Systems. (2003) 279–288
7. Draheim, D., Weber, G.: Modelling Form-based Interfaces with Bipartite State Machines. *Interacting with Computers* **17**(2) (2005) 207–228
8. Preciado, J.C., Linaje, M., Sánchez, F., Comai, S.: Necessity of Methodologies to Model Rich Internet Applications. In: Proceedings of International Symposium on Web Site Evolution, September 26, 2005, Budapest, Hungary. (2005) 7–13
9. Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G.: Conceptual Modeling and Code Generation for Rich Internet Applications. In: Proceedings of International Conference on Web Engineering, July 10-14, 2006, Menlo Park, CA, USA. (2006) 353–360
10. Linaje, M., Preciado, J.C., Sánchez-Figueroa, F.: A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. In: Proceedings of International Conference on Web Engineering, July 16-20, 2007, Como, Italy. (2007) 226–241
11. Urbietta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the Interface of Rich Internet Applications. In: Proc. LA-WEB’07. (2007) 144–153
12. Pérez, S., Díaz, O., Meliá, S., Gómez, J.: Facing Interaction-Rich RIAs: The Orchestration Model. In: Proc. ICWE’08. (2008) 24–37
13. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc. (2002)
14. Vdovják, R., Fräsincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering* **1**(1-2) (2003) 3–26
15. Schwabe, D., Rossi, G., Barbosa, S.D.J.: Systematic Hypermedia Application Design with OOADM. In: Proc. Hypertext’96. (1996) 116–128
16. Alalfi, M.H., Cordy, J.R., Dean, T.R.: A Survey of Analysis Models and Methods in Website Verification and Testing. In: Proc. ICWE’07. (2007) 306–311
17. Biel, B., Book, M., Gruhn, V., Peters, D., Schäfer, C.: Handling Backtracking in Web Applications. In: Proc. EUROMICRO’04. (2004) 388–395
18. Baresi, L., Denaro, G., Mainetti, L., Paolini, P.: Assertions to Better Specify the Amazon Bug. In: Proc. SEKE’02. (2002) 585–592