Towards an Operation Model for Generated Web Applications

Mihály Jakob Holger Schwarz Fabian Kaiser Bernhard Mitschang

University of Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany

{mihaly.jakob, holger.schwarz, fabian.kaiser, bernhard.mitschang}@ipvs.uni-stuttgart.de

ABSTRACT

This paper describes a new approach for the development of data-intensive web applications that depend on non-trivial data manipulation. E-Commerce web sites, on-line auction systems and large enterprise web portals fall into this category, as they require comprehensive data access, data processing and data manipulation capabilities. However, existing methodologies mainly concentrate on modeling content, navigation and presentation aspects of read-only web sites. Approaches that consider modeling data operations incorporate them into existing models resulting in a less clear design. We argue that existing models are not sufficient to express complex operations that access or modify web application content. Therefore, we propose an additional *Operation Model* defining operations for data-intensive web applications. We also propose the utilization of a web application generator to create an *Operation Layer* based on this *Operation Model*.

Categories and Subject Descriptors

D.2.3 [**Software Engineering**]: Design Tools and Techniques – *Object-oriented design methods*

D.2.11 [**Software Engineering**]: Software Architectures – *Domain-specific architectures*

H.4 [**Information Systems Applications**]: General

H.5.4 [**Information Interfaces and Presentation**]: Hypertext / Hypermedia – *Architectures, Navigation, User issues*

General Terms

Design, Languages

Keywords

Web application design, Operation modeling, Data-intensive applications, Code generation, Object-orientation

1. INTRODUCTION

The development of data-intensive web sites has been the subject of many research approaches in the field of web application engineering. These web sites mainly focus on making large amounts of data available on the web and in some cases, they also provide simple data entry functionality. Most research approaches model web applications with different models for *content, navigation* and *presentation*. These models are useful as they provide an incremental methodology for building web applications in a systematic way. Furthermore, they can be used for automatic generation of web application code.

However, present-day web applications are not only based on large amounts of data. They also require powerful operations that determine the manner of content provision and allow data manipulation. E-commerce web applications, for example, often rely on advanced functionality like a shopping cart, powerful search options or personalized recommendations. To serve customers optimally, product listings have to be adequately filtered and sorted. On-line auction systems allow users to add new auction items to the system and to alter these items if necessary. These operations not only change simple content objects of the underlying web application but also add and modify relations between objects and extend the user interface by creating new pages that can be navigated to. Thus, present-day web applications have to provide at least operations for

- adding content objects or new relationships between content objects,
- altering existing content objects or existing relationships between content objects,
- deleting existing content objects or existing relationships between content objects,
- filtering and sorting content objects according to specified criteria.

These operations represent an important part of the web application's application logic that cannot be easily expressed with common models for content, navigation or presentation. Furthermore, the separation of content, application logic and presentation is an important paradigm that should be incorporated into the web application development process. For this reason, data operations should be modeled by means of a separate model instead of incorporating them into existing models.

In this paper, we introduce a new model, the *Operation Model* that defines operations of a web application. As a first step this model provides operations that allow complete read and write access to the web application's content. In the future we intend to use this model to facilitate user specific access to content and extend it to provide composite transactional operations. The Operation Model along with other well-established models build the basis for the automatic generation of complete web applications.

The rest of the paper is organized as follows: In Section 2 we briefly describe how data-intensive web sites are usually modeled. We also show by an example how WebML introduces operations

MDWE'06, July 11–14, 2006, Palo Alto, CA, United States of America.

into the application logic of web applications and point out some limitations of this approach. In Section 3 we emphasize important requirements for a solution that circumvents these shortcomings. We propose a new model, the *Operation Model*, and show how it can be used in conjunction with other well-established models to achieve maximum flexibility and a clear web application design. Section 4 depicts how the described models can be used for the generation of web applications. Finally, we conclude the paper and discuss open research issues in Section 5.

2. STATE-OF-THE-PRACTICE IN WEB APPLICATION MODELING

Large web applications require a thorough conceptual modeling of their content, application logic, navigation and presentation in order to keep the development process manageable and the resulting application maintainable. A formal language should be used to describe content objects, object relationships as well as their operational and presentational features. First, this ensures a uniform specification of the developed system. Secondly, it allows the generation of application source code and the generation of a user interface for the designated target system. An overview of well-established models for web application development is depicted in Figure 1.

The *Content Model* defines content objects and their relations. The *Composition & Navigation Model* describes the composition of web pages and the navigation structure of the web application. Finally, the *Presentation Model* defines the positioning and the visual appearance of web page components.

Over the last decade, numerous approaches for web application development have been proposed. Some of them like OOHDM [12], Araneus [11], AutoWeb [6], OO-HMethod [7], WebML [3][4][5], UWE [9][10] and W2000 [1][2] are complex design methodologies describing web applications with different models. Although the models in use are sometimes named differently, they usually possess modeling constructs for content, navigation, page composition and presentation aspects. Most of these methodologies concentrate on the modeling of data-intensive web sites that do not support operations. The two exceptions are WebML and W2000.

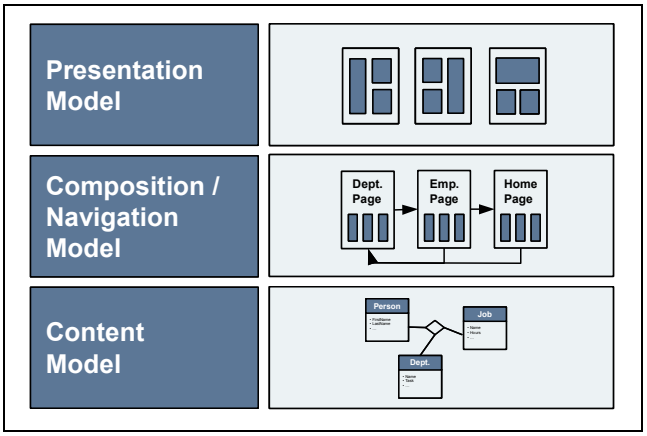

Figure 1. Established Web Application Development Models

The navigation model of WebML provides constructs for defining operations that create, delete and modify content objects and create or delete binary relations. Figure 2 depicts a simple WebML example. The *Content Model* comprises constructs modeling the entities *Employee* and *Department*. The *Composition & Navigation Model* shows the modeling of an *Employee Page*, an *Edit Employee* page and a *Department Page* as well as the operations *update employee* and *assign employee to department*. A simple navigation link points from the *Employee Page* to the *Edit Employee Page* and, after a successful modification (indicated by the OK-edge), back to the *Employee Page*. The *Department Page* contains basic information about the department and lists all employees of the system.

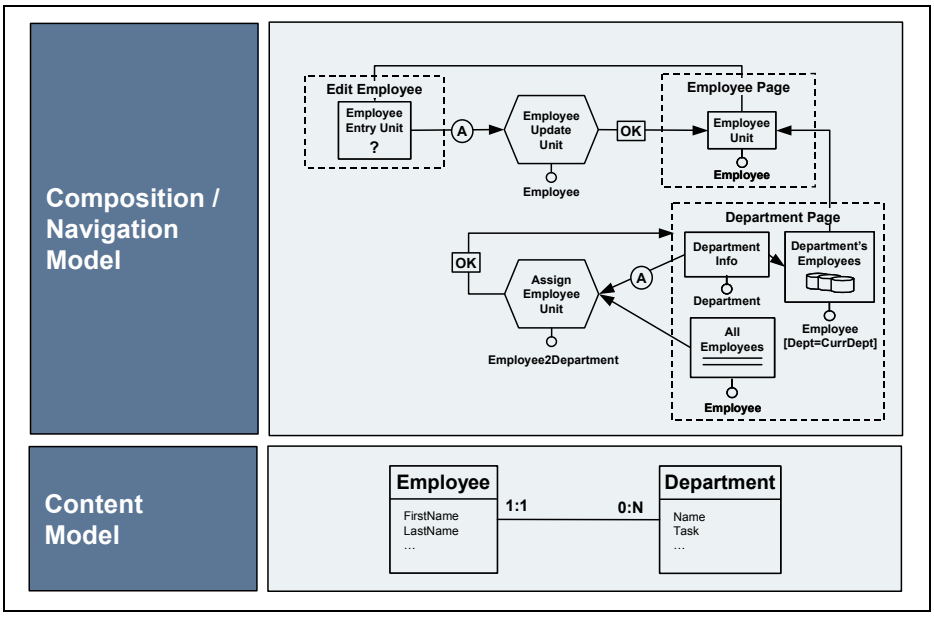

Figure 2. WebML Example

After selecting an employee from the *All Employees* index, the *assign employee* operation is used to assign it to the department and navigation returns to the same page following the OK-edge. Additionally, a multi-data unit shows all employees of the current department. Note, that WebML also provides so-called KO-edges, which are used to specify the navigation step after an operation failure. We omit these edges for the sake of simplicity. To keep the example small, we omit the *Presentation Model* as well.

The WebML approach for modeling operations has some limitations. First, the integration of operations into the navigation model makes this model less intuitive as some edges in the navigation graph represent navigational links whereas others represent data flow. For example, the edge labeled with the letter *A* between the *Employee Entry Unit* and the *Employee Update Unit* indicates the provision of parameter values from an entry form to the corresponding processing method. In contrast to that, the edge between the *Employee Unit* and the *Employee Entry Unit* just indicates a standard navigation step from one page to another. Secondly, connections between the two models are established only by naming and not in a visual manner. Thus, name-matching is required to determine where the data for the *All Employees* index or for the *Department's Employees* listing comes from.

Another approach to model functional requirements of web applications is taken by W2000. High-level UML diagrams are used for this purpose, e.g., use-case diagrams define users and their possible actions, whereas interaction diagrams define object interactions. Although this approach also defines operations for web applications, implementation and code generation aspects are not considered in the W2000 approach, which is an important difference compared to our approach.

3. OPERATION MODEL

As a conclusion drawn from the shortcomings presented in the previous section, we specify the following requirements for a proper solution:

- A separate model is required that specifies operations as part of the application logic of a web application to ensure the separation of operation logic from content and navigational concerns.
- Navigational and data-flow edges should not be combined in the same model. This avoids confusion and provides an intuitive way for the modeling of application logic.
- Modeling constructs of different models should be linked in a visual manner to provide an enhanced overview of the developed web application.

We propose an *Operation Model* that defines operations allowing complete read and write access to web application data. This model should serve as a mediator between the *Composition & Navigation Model* and the *Content Model* of the web application thereby ensuring the clear separation of content, operations and navigational concerns. Figure 3 shows an *Operation Model* example for an *Employee* entity.

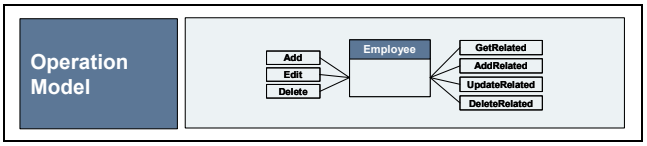

Figure 3. Operation Model

On the left side of the entity, we list basic operations that provide access to the data of this entity. These operations may only occur once and are self-explaining. On the right side of the entity, we list operation types that provide access to relationships between the *Employee* entity and other entities. Each operation type may occur several times as the entity can participate in arbitrary relationships. For example, if the *Employee* entity is participating in a relationship *works in* with the *Department* entity, the operations *GetDepartment*, *AddDepartment*, and so forth can be specified. These operations can be defined for all relationships of the *Employee* entity thereby resulting in multiple occurrences for each operation type. Figure 4 shows how the *Operation Model* interacts with both the *Composition & Navigation Model* and the *Content Model*.

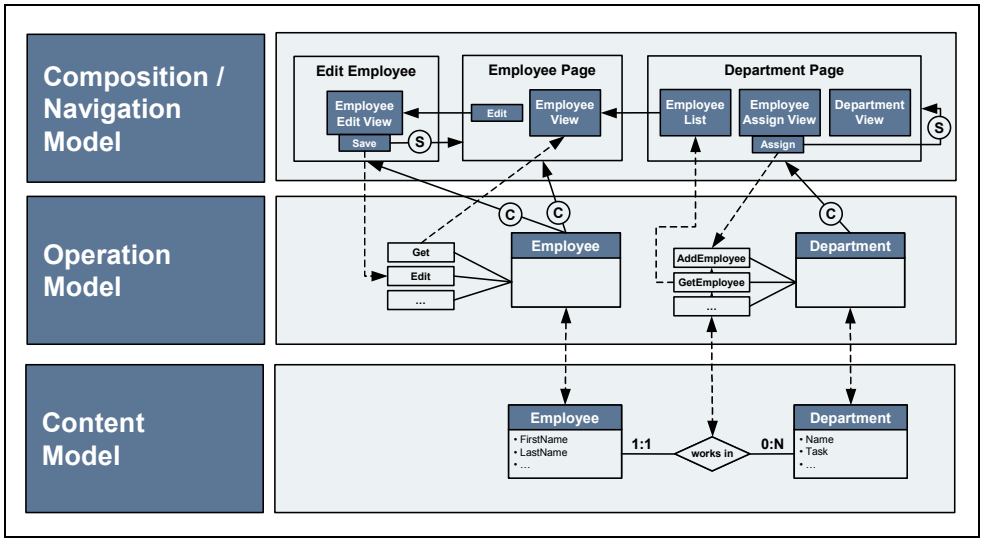

Figure 4. Operation Model Interactions

There are two types of connections between the *Composition & Navigation Model* and the *Operation Model*. The first type is a so-called *Context Edge* (labeled with the letter C) defining the context for each page. The context is always a certain entity of the

system that determines the scope of an operation. For example, Figure 4 shows two pages with the context *Employee* and one page with the context *Department*. The second type of connection between the *Composition & Navigation Model* and the *Operation Model* is a dashed *Data-Flow Edge* indicating an operation call. Note, that these edges are directed showing whether the data of the web application is accessed or modified.

Analogously to Figure 2 the *Composition & Navigation Model* of the example in Figure 4 defines an *Employee Page*, an *Edit Employee* page and a *Department Page.* An *Edit* button on the *Employee Page* allows the navigation to the *Edit Employee Page*. This page provides the *Employee Edit View* and a *Save* button. The effect of providing values on this page and pushing the *Save* button is twofold. First, it results in sending data to the *Edit* operation of the *Employee* entity that is modeled in the *Operation Model*. Secondly, it initiates the navigation back to the *Employee Page* following the *Success Edge* (labeled with the letter S). The *Department Page* provides three main elements. First, the *Department View* that covers basic information about the department. Secondly, an *Employee List* that shows the department's employees accessing *the GetEmployee* operation in the *Operation Model*. Thirdly, the *Employee Assign View*, which allows to assign an employee to a department. A list of all employees that is required for the latter view could come from the *GetEmployee* operation of an *EmployeeCatalog* entity. For the sake of simplicity, we omit this entity in Figure 4. When the user pushes the *Assign* button on the *Department Page,* the *AddEmployee* operation in the *Operation Model* is activated which in turn assigns an employee to the department. Note, that so-called *Failure Edges* that usually point to an *Error Page* are omitted as well.

Similar to the connections between the *Composition & Navigation Model* and the *Operation Model* there are also edges between the *Operation Model* and the *Content Model*. These edges show which entities or relationships are accessed by a certain operation. As basic operations are always associated with a single entity, their connection to the *Content Model* is indicated by the multi-directional edge between the entity constructs of both models. The *Employee* entity of the *Operation Model* is connected to the *Employee* entity of the *Content Model.* This shows what data is accessed by these operations. The situation is somewhat different when it comes to the modeling of operations that access relationships. Various operations of a certain entity may reference different relationships thus this connection is indicated by edges between operations of the *Operation Model* and relationships of the *Content Model*. For example, the *GetEmployee* operation is connected to the relationship *works in* thereby delivering data for the *Employee List* of the *Department Page*.

As a conclusion, we summarize the advantages of our approach. First, edges in the *Navigation Model* clearly represent the navigation structure of the developed web application. Secondly, the separate *Operation Model clearly defines operations* and associates them with the modeling constructs of the *Composition Model*. However, to achieve this enhanced overview a minor requirement has to be met. Each page has to be assigned to an entity that builds the page's context. As depicted in Figure 4, we indicate the context of a web page by a so-called *Context Edge* between the *Content & Navigation Model* and the *Operation Model*.

4. WEB APPLICATION GENERATION

The model-based development of web applications shows many benefits. One aspect is the clear structure of the developed web application that results from the usage of well-established models and standard modeling constructs. However, an equally important aspect is the ability to generate substantial parts of the web application based on existing formal definitions. Therefore, models and modeling constructs require equivalent definitions in a machine-readable formal language and translation between both formats should always be possible. Figure 5 depicts the generation process for web applications based on XML as a machine-readable format.

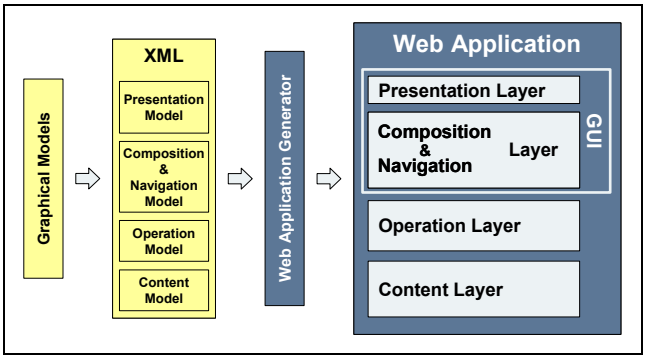

Figure 5. Web Application Generation

In [8] we describe a web application generator that uses XML definitions as input to generate ready-to-use web applications that support powerful operations for content manipulation. Based on different models the web application generator is able to generate corresponding layers for each of the used models. However, the direct usage of XML for model definition has one important disadvantage. The connection between different models has to be established by naming. Using the example from Figure 4, this means that the XML definition for the *Employee Assign View* in the *Composition & Navigation Model* has to reference the *AddEmployee* method's XML definition in the *Operation Model* by its name. Analogously the XML definition of the *AddEmployee* method has to refer to the *works in* relation in the *Content Model*. To abstract from such naming concerns we propse to use a graphical notation, like the one presented in this paper, as the first model in the web application development process. This allows to establish connections between different models by edges and to translate them into the appropriate namespace later on.

5. SUMMARY AND FUTURE WORK

In this paper we presented an approach for modeling and generating data-intensive web applications that rely on advanced operations. We introduced a new model, the *Operation Model*, which allows the definition of operations that build a bridge between the content and the user interface of a web application. An advantage of this approach is an enhanced overview of the web application modeling process supported by a further separation of concerns. We proposed that the *Operation Model* should be used for the generation of an *Operation Layer* of the resulting web application.

An important topic for our future research is the definition of a *User Model* that can be used in conjunction with existing models to define user specific navigation, operations and content

presentation. We believe that the modeling of users should be based on roles and access rights that can be associated with modeling constructs of existing models. Therefore, user modeling is an orthogonal aspect regarding content access, operations and navigation.

6. ACKNOWLEDGMENTS

This work was conducted in the context of the project *nova-net*, which is funded by the German Federal Ministry of Education and Research (BMBF).

7. REFERENCES

[1] L. Baresi, F. Garzotto and P. Paolini. Extending UML for Modeling Web Applications. *In Proc. of HICCS34*, Maui, Hawaii, USA, Januar 2001.

[2] L. Baresi, F. Garzotto and P. Paolini. From Web Sites to Web Applications: New Issues for Conceptual Modeling. *In Proc. of WCM2000*, Salt Lake City, USA, October 2000.

[3] A. Bongio, S. Ceri, P. Fraternali and A. Maurino. Modeling Data Entry and Operations in WebML. *In Proc. of WebDB2000,* Dallas, USA, May 2000.

[4] S. Ceri, P. Fraternali and A. Bongio. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. *In Computer Networks Vol. 33, No. 1-6, p. 137-157,* June 2000.

[5] S. Ceri, P. Fraternali and M. Matera. Conceptual Modeling of Data-intensive Web Applications. *In IEEE Internet Computing, Vol. 6, No. 4, p. 20-30,* July 2002

[6] P. Fraternali and P. Paolini. Model-driven Development of Web Applications: the AutoWeb System. *In ACM Transactions on Information Systems Vol. 18, No. 4, p. 323-382*, October 2000.

[7] J. Gomez, C. Cachero and O. Pastor. Extending a Conceptual Modelling Approach to Web Application Design. *In Proc. of CaiSE2000*, Stockholm, Sweden, June 2000.

[8] M. Jakob, H. Schwarz, F. Kaiser and B. Mitschang. Modeling and Generating Application Logic for Data-intensive Web Applications. To appear in Proc. of ICWE 2006, Palo Alto, USA, July 2006.

[9] N. Koch and A. Kraus. The Expressive Power of UML-based Web Engineering. *In Proc. of IWOOST02*, Cyted, 2002.

[10] A. Kraus and N. Koch. Generation of Web Applications from UML Models Using an XML Publishing Framework. *In Proc. of IDPT 2002*, Pasadena, USA, June 2002.

[11] P. Merialdo, P. Atzeni and G. Mecca. Design and Development of Data-Intensive Web Sites: The Araneus Approach. *In ACM Transactions on Internet Technology Vol. 3, No. 1, p. 49-92*, Februry 2003

[12] D. Schwabe, G. Rossi and S. Barbosa. Systematic Hypermedia Application Design with OOHDM. *The 7th ACM Conference on Hypertext*, Washington, USA, March 1996