



Centro de
Investigación
Operativa



I-2007-17

Tool Support for Model-Driven Development of Web Applications

Jaime Gómez, Alejandro Bia, Antonio Párraga

June 2007

ISSN 1576-7264

Depósito legal A-646-2000

Centro de Investigación Operativa
Universidad Miguel Hernández de Elche
Avda. de la Universidad s/n
03202 Elche (Alicante)

TRABAJOS I+D

Tool Support for Model-Driven Development of Web Applications¹

Jaime Gomez (1), Alejandro Bia (2), Antonio Parraga (1)

(1) DLSI - Universidad de Alicante
e-mail: [jgomez, aparraga]@dlsi.ua.es

(2) CIO - Universidad Miguel Hernández
e-mail: abia@umh.es

Abstract: This paper describes the engineering foundations of VisualWADE, a CASE tool to automate the production of Web applications. VisualWADE follows a model-driven approach focusing on requirements analysis, high level design, and rapid prototyping. In this way, an application evolves smoothly from the first prototype to the final product, and its maintenance is a natural consequence of development. The paper also discusses the lessons learned in the development of the tool and its application to several case studies in the industrial context.

1 Introduction

The rapid evolution of Internet in general and of the WWW in particular has promoted in recent years intensive research in the field of conceptual modeling of Web applications. This fact has induced a new research trend within Software Engineering known as Web Engineering. In this context, different methods, languages, tools and design patterns for web modeling have been proposed. Some of the most relevant studied so far are HDM [4], WebML[2], OOHDM [7], UWE [10], ADM [1], and OO-H [6]. These methods are centered mainly in the definition of navigational and presentational aspects relative to the semantic of models to capture relevant properties of web environments. However, few are the proposals that have tried to apply their methods to solve complex real cases to verify the effectiveness of their modeling approach. Much lesser have been the attempts (successful or not) of building specific purpose tools for Web Engineering. We can mention WebRatio [9] developed at the Politecnico di Milano (Italy) under the technical direction of Prof. Piero Fraternali, or ArgoUWE [11] developed at the Ludwig Maximilians University of Munich (Germany) under the technical direction of Dr. Nora Koch.

¹ This article has been supported by the MEC through the METASIGN project, reference number: TIN2004-00779

This paper describes the engineering foundations of VisualWADE, a CASE tool for the design and automatic production of Web applications based on the OO-H method developed at the University of Alicante. The underlying idea behind VisualWADE consists of an appropriate combination of simple concepts (modeling elements), that allow the designer to model and automatically generate any type of web-based system, from a web-portal or company intranet to a secure web site for electronic commerce. VisualWADE exploits a group of very well-known concepts to capture the complexity of real web applications. The underlying method, OO-H (object oriented hypermedia), provides specific modeling elements to represent navigation maps based on a notation compatible with UML. The captured specification is compiled making use of model-based code generation techniques, and as a result, it produces a web application with a default user interface. This user interface can be refined within the environment to obtain the final appearance of the web application with the consequent increment of development productivity. The paper is organized as follows: section 2 provides a brief introduction to the OO-H method to familiarize the reader with the modeling notation. Section 3 describes the basic aspects of modeling organized according to three different perspectives through which a web application is modeled with VisualWADE (structure, navigation and presentation). A running example (web-based mail system) is used to describe the tool support. Section 4 describes the lessons learned in the application of VisualWADE in the development of real web engineering projects on various application domains (tax management, Internet banking, digital signatures). Finally, the paper ends with some conclusions, based on our experience, about the evolution and the future of web engineering methods, techniques and tools.

2 A brief introduction to the OO-H method

The OO-H method (object oriented hypermedia) is a generic method based on the object oriented paradigm that provides a specific notation and semantics for the development of web based applications. OO-H defines a set of diagrams, techniques and tools that altogether comprise a complete approach for the modeling of web application. The method includes: a design process, a navigational access diagram (NAD), an abstract presentation diagram (APD) and finally, a CASE tool that supports and automates the development process. With OO-H, a traditional business application can be “converted” to a web-compliant application by adding two new views (diagrams) that complement the structure (class diagram) and behavior views (interaction and state transition diagrams). The first of them, the NAD, is used to specify a navigation view. The second, the APD, captures concepts related to the final-interface presentation details. The NAD enriches a domain model with navigation and interaction characteristics. It also defines constraints about navigation and information which should be showed to the web user. For this purpose, OO-H uses the object constraint language OCL [8]). In this way, a precise navigation diagram can be obtained. On the other hand, the APD contains the definition of abstract pages (pages that are not attached to any specific web language) based on a set of XML templates that capture the relevant presentational properties of the web

interface under construction. OO-H is a well-recognized Web design method in the field of Web Engineering and several publications [5,6] provide detailed information about it. Due to the fact that the focus of this paper is to describe the OO-H tool (called VisualWADE), interested readers are redirected to the references commented above. In the next section, we describe the basic aspects to model a web application with VisualWADE. A running example focused on a web-based mail system is used to introduce the relevant concepts about the method.

3 VisualWADE: Basic Aspects

Domain modeling

The starting point to approach the design of a web application is the domain model represented with a class diagram (see Figure 1).

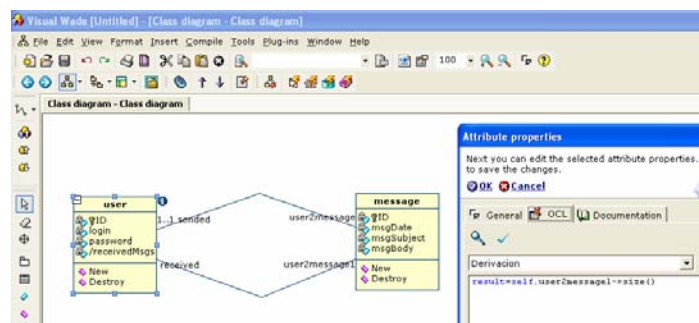


Figure 1: A class diagram in VisualWADE

The notation of this diagram is based on UML which makes it quite intuitive for a designer familiarized with object oriented analysis and design. The application helps the designer throughout the whole edition process by means of a simple, but yet powerful and intuitive graphical interface. The creation of classes, attributes and relationships is carried out with simple mouse actions. The availability within the environment of zoom buttons and global views help the organization and management of complex diagrams comprised of several classes. In Figure 1, a class diagram corresponding to the web-based mail system can be observed. In this case, the *user* class and the *message* class are specified and between them two associations to capture the information corresponding to the messages *sent* and *received* by the user. Derived attributes can also be specified within the class diagram. Attributes stereotyped as “*derived*” has an associated OCL formula that specifies how the value of the attribute is obtained. This is the case of the attribute *receivedMsgs* of the *user* class, whose value is obtained by navigating through the *user2message1* role and calculating the number of instances of the *message* class (*size()* function). Obviously, we have implemented within the environment an OCL compiler that allows the syntactic and semantic validation of the OCL formulas.

Navigation Modeling

A navigation view shows the way in which attributes and services provided by the classes defined in the domain model are accessed. In Figure 2 a partial view of the NAD diagram for a web user is shown.

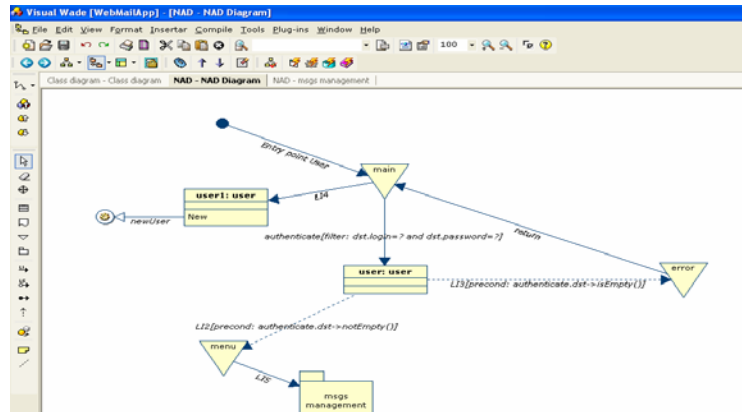


Figure 2: Navigation Diagram in VisualWADE

The entry point is the *main* abstract page, which represents the navigation starting node. From *main* the web user can navigate to the navigational class *user* through the *authenticate* link. Traversing the *authenticate* link requires the evaluation of filter (expressed as an OCL formula). In this case, the user must provide the *login* information (*dst.login*) and a *password* (*dst.password*) to check whether he/she is a valid user for the system. Depending on the success of this validation process the preconditions associated to links *LI2* and *LI3* will determine which of these links will be activated and therefore the destination abstract page (*menu* or *error*) to continue the navigation process. The *notEmpty()* and *isEmpty()* OCL functions provide the necessary information to know if a user is or not registered in the system. Continuing with the description of this navigation diagram, from the *main* abstract page and through the *LI4* link the *newUser* service of the navigational class *user* is accessed. This modeling situation allow a web user to activate the service *newUser* to register new users into the system.. Finally, from the *menu* abstract page the navigation continues to the navigational target *message management* while with the *error* abstract page it returns to the *main* starting page. In Figure 2, the toolbar on the left contains the modeling elements that can be inserted in a NAD. By mouse selections, the internal properties of each element can be edited. For example, a link connects an origin element with a destination element. Link properties allow to specify whether the information of the origin element must be shown in the same page that the information of the destination element or not. The visual environment also includes very useful functions such as copy/paste, do/undo, zoom-in/zoom-out, quick element search and finally rules to check the model's consistency (model checking) and the corresponding warning messages. This last feature will be discussed later on.

Presentation modeling

Once the designer has specified the navigation diagram (completed or partially completed), this diagram can be compiled. As a result a set of XML [3] pages that fulfills the navigation specifications are generated. The XML pages constitute the web application and contain a preliminary web user interface. This preliminary web user interface can be refined within the environment to render the final look and feel.

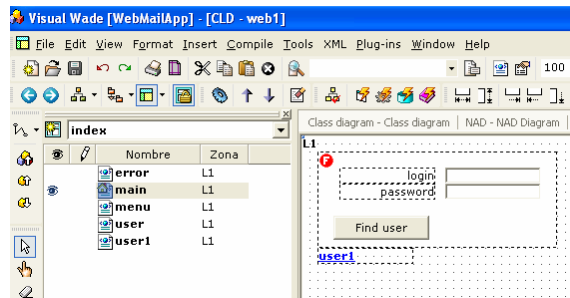



Figure 3: Presentation Diagram in VisualWADE

The aspects that can be refined are those corresponding to the properties of styles, location, colors, just as it could be done with any authoring tool like Frontpage or Dreamweaver.. Figure 3 shows the abstract presentation diagram, result of compiling the NAD of Figure 2. Several zones can be observed: on the left side the page viewer can be observed. It contains the abstract pages that have been generated, alphabetically ordered (*error*, *messages*, *menu*, *main*, *user*, *user1*). On the right side can be seen the editing area where the content of the abstract pages is visualized. The information of the *main* page is showed in this case. As a result of the compilation process from NAD to APD, a form has been generated with the fields *login* and *password* and the corresponding button to execute the action (*OK*). Also shown is the link *user1* which enables the navigation to register a *new user* into the system. This preliminary web user interface can be animated by means of the animation tool .

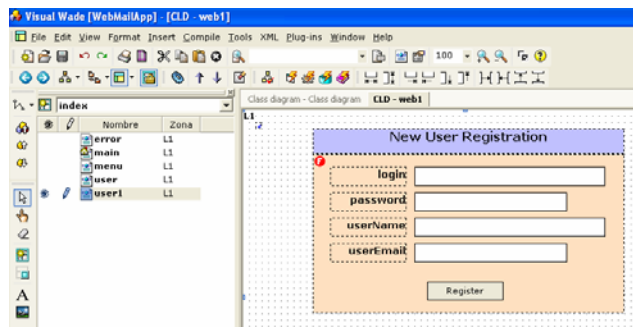


Figure 4: Refinement and animation in the APD

When the animation tool is active the links and buttons of the interface become sensitive to mouse selections allowing the corresponding navigation jumps. While a model is being animated it is still possible to select the edition tool to modify any property of the elements of the interface. Figure 4 is the result of having activated the animation tool, having selected with the mouse the link *user1* of Figure 3, and finally having used the edition tool to modify the look of the page. It can be observed that within these refinements a help text has been included and the position, size and color of the users' creation form have been changed. Once we have introduced the basic features of the VisualWADE environment, next we describe some of the relevant projects where we have applied the tool and interesting lessons learned.

4 VisualWADE: Lessons learned

VisualWADE has been used in such different environments as: the Autonomous Organization of Tax Administration of the Province of Alicante (SUMA) [12], the Mediterranean Savings Bank (CAM) [13], the Association of Industrial Technical Engineers of the Province of Alicante (COITI) [14], as well as to create VisualWADE's own website [15]. In this section we will present some of the characteristics of the systems that have been designed with VisualWADE within these companies. In SUMA Tax Administration, several Web applications have been developed, most of which are data-oriented. Among them, we highlight in chronological order: an inventory management system, SUMA's intranet and its Internet portal. The inventory management system is an application to manage the computer material inventory which is distributed along its 250 offices in the province of Alicante. Actually, inventory management can be seen as a traditional desktop application with a Web user interface. This was the first application that we completely designed using VisualWADE and also the one where more design errors were made. The reason was that we were strongly influenced by the type of user interfaces that the company was accustomed to use, and we were forced to carry out an "unnatural" design for a Web environment. Few Web visualization patterns were used so that the navigation was not intuitive and it was directed by the set of services offered from the interface (the same as in their traditional applications). Inventory management had more than 150 available services in the application. The experience of this development made both the SUMA team and the VisualWADE team to mature very much in a parallel way, and to realize that doing design like this prevented us from taking advantage of all the benefits of Web environments. The second project carried out at SUMA was its intranet. Basically, the intranet provides the necessary functionality to manage the group of news, events, tools and internal documents of the company through an intuitive and user-friendly Web interface. It is the first application that was designed for several user profiles, among them the administrator whose fundamental role is to supervise the information entered to the system from the different departments. Currently, the intranet of SUMA provides services to more than 400 users, and by the third month of operation it had increased its visits by more than 300% compared to the same period of the previous year. The last project carried out was the portal of SUMA. With a navigation design similar to that of the intranet,

this application offers new and interesting services to citizens and city councils. In its design, some of the advanced characteristics provided by VisualWADE have been used, like multi-language support and template-based interface generation. In fact, the graphic design was developed by a subcontracted external company. Later, this graphic design was adapted to the generated navigation model. Regarding the Mediterranean Savings Bank (CAM), the use of VisualWADE was more limited there, mainly for reasons of security which are described in the next section. Coordinated with the team of the CAM-directo service, whose responsibility is to maintain the group of applications to provide bank service through Internet, we used VisualWADE only to design the user interface navigation of the CAM-directo system itself. In this case, none of the capabilities of VisualWADE to generate predefined services was used, since the functionality already existed and had to be simply invoked from the navigation environment. To do so, we used the VisualWADE Web services integration modules (not described in this paper) that “encapsulated” the calls to the different necessary operations through a WSDL specification. The most difficult part was to adapt the return of parameters from the services to the navigation engine for later presentation on the Web interface. Regrettably, this task of integration with Web services still has to be made by hand. The experience was very positive, but we also realized that much of the potential use of VisualWADE is lost, since only a few of the characteristics of the environment like the interface generator were used. The need for privacy and security demanded by bank environments forced us to modify the Web design environments so that these security requirements are treated as first level elements. In this sense, we have created a workgroup with the CAM-directo team to identify and propose a set of modeling primitives that should be incorporated to a security model of Web applications. The PROVE (proyecto visados electronicos) project developed in the COITI (industrial engineer association of the province of Alicante) has been another system where VisualWADE has been applied successfully. The aim of this project is to provide a web-based application to manage electronic documents following the digital signature standards. PROVE makes an intensive use of Web services to offer digital signature services provided by the public key infrastructure (PKI) of the Generalitat Valenciana [16]. PROVE, is the first system developed with VisualWADE that connects three different information systems; the PKI Web server, the CAM-directo system and the COITI system. Conceptually, an industrial engineer can send through the PROVE system a digitally signed electronic document (pdf file that contains the description of a project), pay the corresponding administrative rates through the CAM-directo system, and submit it automatically to the department of COITI for registration. Just like the case of SUMA, the integration with Web services had to be carried out by hand. This is therefore a pending area for improvement in future versions of the tool. PROVE currently serves more than 1350 members, processing more than 15.000 projects per year. Finally, the conceptual design and generation of the VisualWADE website itself has been carried out using the tool. The VisualWADE portal is also based on templates and provides a set of very interesting navigation functionalities. All these functionalities have been fully generated using the properties of VisualWADE. For instance, the possibility to register at the website to access the free download of the tool, or to participate in discussion forums, or even to execute a project’s task-management application example. Contrary to the systems that have been presented

previously, in the VisualWADE portal there is no use of any functionality from third-party developers. In this sense, we could have used any free software for forum management available in Internet, but we preferred to build it from scratch to demonstrate the power and ease of use of the environment. The VisualWADE portal is open since November 2004. Currently it manages a transfer rate of more than 15 Gb por month, and more than 600 downloads of the tool from all over the world have already taken place. Although all these systems that have been briefly presented here constitute a real fact about the usability of advanced web development environments, our current purpose is to make VisualWADE available to the highest number of possible interested users with the objective of exploring new uses and development experiences. We are sure that this new phase will highlight interesting experiences yet to discover.

Next we present some of our experiences acquired during the development of the Web information systems described. We intend to provide some of the experiences of the VisualWADE team in the application of the tool for the resolution of real cases. As common denominator, all these cases required Web based solutions for their information systems. Most of these solutions (85 percent approximately) did not require new developments, but adaptations of existent information systems to the new Web environment, while only the remaining (15 percent approximately) could be considered as new developments, although in fact they were small extensions of functionalities not available until then. As mentioned before, VisualWADE provides, on one hand, predefined operations to support the design of date-intensive Web applications. This type of operations has demonstrated to be enough to provide basic business logic in applications like the inventory management or the intranet of SUMA. In fact, a great number of the Web applications demanded by companies are data-oriented. Therefore, it is particularly important to provide CRUD services for the great majority of Web environments that companies require, particularly companies that possess organizational information systems. On the other hand, the ability to specify and to use Web services in VisualWADE has facilitated the integration of functions that, in the form of legacy software, were needed in the new environment. For reasons dealing with security, privacy and complexity among others, these inherited functions could not be rewritten again but rather they had to be integrated into the new solutions. Consider that many of these companies have invested heavily in the past so that their information systems could reach the maturity and reliability desired. Therefore, what these companies need are not tools for the construction of Web software from scratch, but tools that facilitate the migration of their systems to the new environment, preserving the highly-reliable existent functionality. Regarding the primitives and navigation patterns, our experience shows the necessity of an intensive use of structures like index, show-all and guided-tours in the designed and generated applications. Also, the great volume of information managed by them often requires information to be presented by pieces. Therefore, it became indispensable to offer constructors to paginate the results. In VisualWADE such constructors are an implicit part of the navigation primitives, allowing the designer to enable or disable the pagination property as well as the number of objects per page. We have also identified some lacks related to navigation primitives of particular usefulness that unfortunately VisualWADE does not support at this time, among them, the nested

indexes and the multiple attribute selection. It has been detected that with these two navigation primitives some pending modeling situations, which were not possible to specify or had to be specified in another less efficient way, could have been covered in a satisfactory way. The animation capabilities provided within the presentation diagram proves to be highly productive in several senses. On one hand, it helps to reduce the learning curve of the tool by a 25% approximately, especially on navigation diagram concepts since from the animation environment the effect of a modeling specification on the end-user interface can be seen at a glance. On the other hand, the development speed is increased by a factor of 10 by providing specific model compilers that faithfully reproduce the appearance and the behavior of the animated interface. In all the companies where VisualWADE has been applied we had to carry out periods of computer personnel's training. Our experience in this sense shows that an important update is required by most of the personnel, especially regarding object oriented analysis and design techniques, which we found quite surprising. The computer personnel of these companies possess an adequate knowledge of data modeling, and therefore to understand or to specify a class diagram was not a problem for them. However, we had to struggle to explain the navigation diagram and particularly its most advanced concepts such as the use of OCL to specify filters and preconditions on links and services. In the case of the CAM project, this situation did not happen and the transfer of knowledge was much more agile due to the personnel's high training. Another thing that we have learned is that it is very important that the navigation models take into account temporal data about the navigated information. Unfortunately, the great majority of the existent Web design methods and tools, do not keep in mind this dimension when specifying a navigation space. When modeling the SUMA intranet project, we needed to support navigational requirements to maintain the expiration period of news and events that were generated within the intranet itself and we realized that we needed some means to deal with time in the navigation model of VisualWADE. In this case, we opted for a solution based on extending the OCL language of VisualWADE by adding time primitives. This allowed us to consider operations between dates, etc., within the environment. The experience of working with VisualWADE in the CAM-directo team has been very productive. Especially, to learn both restrictions of the environment and situations that would have been impossible to realize without this experience. For example, the CAM team did not like that VisualWADE's model compilers produce a fixed generation skeleton (architecture). They said that anyone who knew how the different artifacts generated communicate to each other, had a very valuable information to attempt against the security of the generated code. For this reason, our efforts to improve model compilers are directed to provide mechanisms based on MDD/MDA to produce different generation skeletons based on a set of architecture templates.

5 Conclusions

Web developers should improve the productivity and quality to satisfy market needs and reduce delivery times and costs. Regrettably, the methods and standard techniques that are used for Web development still present several deficiencies:

models and tools for analysis and design lack appropriate concepts to capture the development properties in this type of environments. As a consequence most of the application code is written by hand and difficult to reuse; documentation is scarce and of low quality, especially for user interfaces; costs and development time are difficult to predict and quickly get out of control during the maintenance and evolution phases of the application. VisualWADE, in its current state, solves some of these demands, and at present, is being used for the resolution of complex real cases in institutions like SUMA Tax Administration (Alicante), Mediterranean Savings Bank (CAM), and the Association of Industrial Technical Engineers of the Province of Alicante. VisualWADE and OO-H are under permanent development. Our next challenge is to integrate the new development paradigm based on MDD/MDA into the environment. We invite the interested reader to download the latest version of the tool and to experience the benefits of advanced web development environments.

References

1. P. Atzeni, G. Mecca, and P. Merialdo. Design and Maintenance of Data-Intensive Web Sites. In *Advances in Database Technology - EDBT'98*, pages 436–449, 03 1998.
2. S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. In *Position Paper, Web Engineering Workshop, WWW9*, 05 2000.
3. eXtensible Markup Language (XML). <http://www.w3.org/XML/>.
4. F. Garzotto and P. Paolini. HDM A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems (TOIS)*, 11(1):1–26, 01 1993.
5. J. Gómez, C. Cachero and O. Pastor. Extending a Conceptual Modelling Approach to Web Application Design. *CAiSE 2000*: 79-93. LNCS 1789. 2000.
6. J. Gómez, C. Cachero, and O. Pastor. Conceptual Modeling of Device-Independent Web Applications. *IEEE Multimedia* 8(2): 20-32. 2001.
7. D. Schwabe, G. Rossi, and D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proceedings of the seventh ACM conference on HYPERTEXT '96*, page 166, 1996.
8. Jos Warmer and Anneke Kleppe. *The Object Constraint Language. Precise Modeling with UML*. Addison-Wesley, 1998.
9. WebRatio Web Site <http://www.webratio.com>.
10. N. Koch, A. Kraus: The Expressive Power of UML-based Web Engineering, In *Proc. of the 2nd. IWWOST, CYTED, Málaga, Spain, June 2002*, 105-119.
11. A. Knapp, N. Koch, F. Moser, G. Zhang. ArgoUWE: A CASE Tool for Web Applications. *EMSISE03*, 14 pages, online publication at <http://www.pst.informatik.uni-muenchen.de/~kochn>
12. SUMA Gestion Tributaria. <http://www.suma.es>
13. Mediterranean Savings Bank. <http://www.cam.es>
14. PROVE. <http://www.copital.org>
15. VisualWADE. <http://www.visualwade.com>
16. PKI Generalitat Valenciana <http://pki.gva.es>