A Review to Model-Based User Interface Development Technology

Pedro J. Molina
CARE Technologies S.A.
Pda. Madrigueres, 44.
03700 Denia, Alicante, Spain
+34 966 345555
pjmolina@care-t.com

ABSTRACT

This position paper discusses the idea of the suitability of the Model Based User Interface Development (MB-UID) to develop commercial applications in industrial environments. Main problems, advantages, author's experiences, and current trends are presented.

Keywords
Model-based user interface development, conceptual modeling, device independent user interface, code generation.

INTRODUCTION

Model-based User Interface Development has had a significant research during more than twenty years. Under these conditions, technology should be mature enough to be applied to the professional development software community.

However, reality reveals the some pitfalls. Industry is still leaded by RAD tools (Rapid Application Development), IDE (Integrated Development Environments) and authoring tools (like Macromedia Dreamweaver or Macromedia Flash). These kinds of tools are characterized for being very WYSIWYG (What You See Is What You Get) oriented, useful to create prototypes and real UIs, at a low level detail, exploiting all the design aspects the tool is conceived for. Empowering, in this way, the creativity of the artist or designer, enabling the possibility of innovation of new interaction styles and designs but also opening the door to create bad designs by inexperienced designers.

In this game, the quality of traditional UIs depends in a strong factor on the experience of the designers and their skills in the platform and development tools.

On the other hand, the market is moving to web interfaces, ubiquitous systems, and wireless. More user interfaces for every-day applications are needed in several devices (multiple user interfaces), also at the same time.

Developing UIs of this kind of systems has extra constrains and constitutes a challenge per se. Different kinds of homogeneity among different platforms should be preserved in the User Interface. Otherwise, the user's learning curve will force him to learn a new UI for each device he uses to access the system.

In the quoted systems, the user interface can be specified once and then refined for each target devices until reaching the implementation. Such approach can be supported by MB-UID methods and tools. Moreover, using an approach capable of generating the UI code, a high percentage of time and resources can be saved.

NOVAK'S RULE

In my humble opinion, the main problem of MB-UID approaches can be summarized using the Novak's rule:

> *"Automatic Programming is defined as the synthesis of a program from a specification. If automatic programming is to be useful, the specification must be smaller and easier to write than the program would be if written in a conventional programming language."* [6]

Methods should be properly supported by tools, and should provide visible assets, evident to the practitioners like code generation, model validation, animation, verification, documentation, etc.

UNSOLVED PROBLEMS

Depicting the Novak's rule for MD-UID methods and tools we can list the following ones:

1. **Maintainability.** Specifications and artifacts produced by the tools should have easy maintenance: allowing to keep the changes in the specification or, alternatively, allowing a sort of reverse engineering from the code to the model.
2. **Scalability.** Practitioners and designers need to apply these methods and tools to real-life problems. If the methods and tools have been poorly tested outside academic environments, is difficult to fine-tune the method and tools for

scalability to guarantee the success in industrial scenarios.

3. **Round Trip Problems.** [1] Code generated from a model has more constrains and less choices that the allowed in the final target device. Frequently, some sort of tweaking or beautification must be applied to modify the final UI to fit the usage scenario. These out-of-the-model changes should be reapplied (ideally automatically) whenever the model changes and the UI needs to be regenerated accordingly.
4. **Integration with artists' designs.** Especially important for web sites and multimedia contents, artists have a relevant role in the developing of such systems. MB-UID artifacts must be integrated with this other source of elements to configure the final UI. To support this kind of development, special tools or conventions must be followed in both sides of the development.
5. **Lack of standards** like UML for software design or architecture. There is no standard for UI development. At least in software design there is a common agreement in the notation: UML. However, for user interface, we are still far away of having such standards.
6. **Lack of robust code generators.** More code generators are needed to produce UIs for different kind of applications. Few tools are available on the market and are limited to specific context of usage.
7. **Lack of integration with business logic.** UIs per se are not enough to build running systems. Good integration mechanisms with application's functionality (business logic) are needed. Much better if the business logic can also be specified and generated.
8. **Lack of commercial tools supporting the methods.** Few tools are available on the market. Practitioners interested in the field need to know them in first place, use and evaluate them to check if a tool fits their needs. There are also methods without any tool support at all: making much harder for practitioners to use such methods.

PROS

Despite the quoted problems, I strongly believe that producing commercial applications using MB-UID technology has mayor advantages:

1. **The abstraction level is higher** that working with development environments. The specification is less dependent from the underlying technology.
2. **Better productivity.** A percent of the final UI can be directly generated and used in the final system without any further changes.
3. **Better quality.** Generated code has always the same elements in the same places. Repeating tasks exactly in the same way every time is easier for a machine than for humans, obtaining, in this way homogeneity. Conformity with standards is also easy to obtain by generating the code according to such standards.
4. **Less errors.** Generated code contains zero generation errors (if the generator is robust and it has been thoroughly tested). Semantic errors can still appear but as a consequence of misunderstanding requirements or modeling misprints.
5. **Provides a precise Engineering process.** Development can be repeated, measured and tracked as a production line. Future projects can be estimated and scheduled with more precision based on previous data.
6. **Multiple device support:** using generators for multiples devices, platforms and architectures.
7. **Less Time to Market.** UIs can be built in less time allowing to put the product early in the market.

ENVISIONING NEW TOOLS AND CHALLENGES

From my point of view, new generation tools should address the previously quoted problems to overpass the Novak's rule.

Modeling tools should be visual and support sketching as done in DENIM [4]. Easy of use is crucial to make work perceived as a non time-consuming task.

An standard in XML representation is needed urgently as a base for tools interchange. Later on, notations and semantics should be standardized also.

Going form initial requirements gathering to final

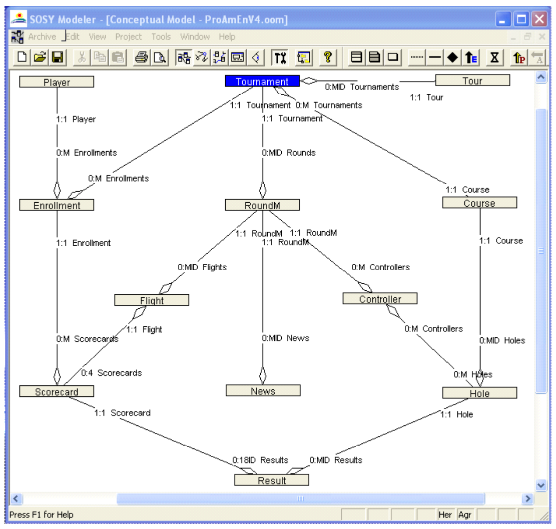
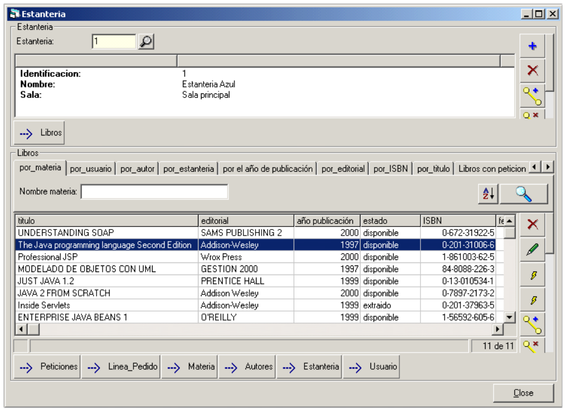


Figure 1. Modeling tool for creating specifications.



Estanteria: 1

Identificacion:	1
Nombre:	Estanteria Azul
Sala:	Sala principal

Libros

por_materia | por_usuario | por_autor | por_estanteria | por el año de publicación | por_editorial | por_ISBN | por_titulo | Libros con peticion

Nombre materia:

titulo	editorial	año publicación	estado	ISBN	fe
UNDERSTANDING SOAP	SAMS PUBLISHING 2	2000	disponible	0-672-31922-5	
The Java programming language Second Edition	Addison-Wesley	1997	disponible	0-201-31006-6	
Professional JSP	Wrox Press	2000	disponible	1-861003-62-5	
MODELADO DE OBJETOS CON UML	GESTION 2000	1997	disponible	84-8088-226-3	
JUST JAVA 1.2	PRENTICE HALL	1999	disponible	0-13-010534-1	
JAVA 2 FROM SCRATCH	Addison Wesley	2000	disponible	0-7897-2173-2	
Inside Servlets	Addison-Wesley	1999	extraido	0-201-37963-5	
ENTERPRISE JAVA BEANS 1	O'REILLY	1999	disponible	1-56592-605-6	

11 de 11

Figure 2. Generated UI for Windows environment.

implementation is a long path that needs to be depicted in subphases, for example: analysis, logical design, physical design, implementation. An MB-UID should cross all this stages in a consistent seamless way. At such, Model Driven Architecture [7] based on Platform Independent Models (PIMs) and Platform Specific Models (PSM) should be taken into account.

Finally, I think that patterns [3, 8] can play a relevant role in the development of UIs. Design patterns, Usage patterns, or Conceptual patterns are examples of how patterns can help to build User Interfaces.

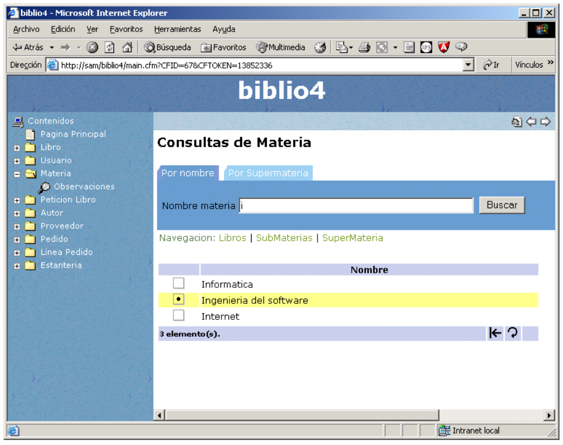
If developers perceive the methods and tools as something really useful (saving works-hours in any way) they will be using them.

AUTHOR'S BACKGROUND

I have a PhD in Computer Science obtained in the Universidad Politécnica de Valenica, Spain. During the development of the PhD thesis I combined the academic point of view from university and a practical work implementing the ideas in a software engineering firm: CARE Technologies, where I currently works as a researcher and a software engineer.

During the last years I have been involved in the development of a user interface specification model based on conceptual patterns [5], a tool for supporting it, and code generators to produce UI code to several desktop and web platforms. All of these products are part or the commercial tool suite OlivaNova Model Execution System.

Figure 1 shows a screenshot of the OlivaNova Modeler [2]. Figure 2 shows an example of generated UI for VB in Windows. Finally, Figure 3 shows a UI generated for web environment using Cold Fusion MX.



http://sam/biblio4/main.cfm?CFID=67&CFTOKEN=13852336

Consultas de Materia

Nombre materia i

Navegacion: Libros | SubMaterias | SuperMateria

	Nombre
☐	Informatica
●	Ingenieria del software
☐	Internet

3 elemento(s).

Figure 3. Generated UI for a Web environment.

More info at: http://www.dsic.upv.es/~pjmolina/EN/.

REFERENCES

1. Bergman. L. et al. "Combining Handcrafting and Automatic Generation of User-Interfaces for Pervasive Devices". In Ch. Kolski y J. Vanderdonckt (editors), *"Computer-Aided Design of User Interfaces III"*, pp. 155–166. Kluwer Academics Publisher, Dordrecht, Valenciennes, France, May 2002.
2. CARE Technologies S.A. *OlivaNova Model Execution System*. http://www.care-t.com
3. Greene, S. et al. *CHI'2003 HCI Pattern Workshop*. http://nitro.watson.ibm.com/chi2003Workshop.
4. Landay J., Myers B. *"Sketching Interfaces: Toward More Human Interface Design."* IEEE Computer, vol. 34, no. 3, March 2001, pp. 56-64.
5. Molina, P.J. et al. *"JUST-UI: A User Interface Specification Model"* Proceedings of Computer Aided Design of User Interfaces, CADUI'2002, Les Valenciens (2002) France, pp. 323-334.
6. Novak G.S. *Novak's rule*: http://www.cs.utexas.edu/users/novak/index.html.
7. Object Modeling Group. Model Driven Architecture. 2001. Available at: http://www.omg.org/cgi-bin/apps/doc?ormsc/01-07-01.pdf
8. van Welie M., Trætteberg H. *"Interaction patterns in user interfaces"*. In 7th. Pattern Languages of Programs Conference, Allerton Park Monticello, Illinois, USA, August 2000.