Model-Driven Development of Digital Libraries: Generating the User Interface*

Alessio Malizia
University "La Sapienza"
Dep. Computer Science
Rome, Italy
malizia@di.uniroma1.it

Esther Guerra
Universidad Carlos III
Dep. Computer Science
Madrid, Spain
eguerra@inf.uc3m.es

Juan de Lara
Universidad Autónoma
Dep. Computer Science
Madrid, Spain
jdelara@uam.es

ABSTRACT

Digital Libraries (DLs) are extremely complex information systems that integrate findings from disciplines such as hypertext, information retrieval, multimedia services, database management, and human-computer interaction. Designers of DLs are often multidisciplinary teams, which include library technical staff and computer scientists. Wasted effort and poor inter-operability can therefore ensue, raising the costs of DLs and risking the fluidity of information assets.

To alleviate these problems, we use a model-driven approach for the design and automatic generation of code for DLs. In particular, we use a Domain Specific Visual Language (DSVL) made of four diagram types (collection, structural, service and societal) which describe the different aspects of a DL. We have built a code generator able to produce XUL code from the design models for the DL user interface. This XUL code integrates predefined components for the different services, according to the model specification.

Categories and Subject Descriptors

Information Systems [**INFORMATION STORAGE AND RETRIEVAL**]: Digital Libraries—*System Issues*

General Terms

Design, Languages, Human Factors

1. INTRODUCTION

The proper concept of a Digital Library (DL) seems hard to be completely understood and evades definitional consensus [6]. For example, a Delphi study [4] of DLs coalesced a broad definition: organized collection of resources, mechanisms for browsing and searching, distributed networked environments, and sets of services objectified to meet users' needs. Underlying all these definitions, there is the agreement that DLs are fundamentally complex, due to their interdisciplinary nature. They are usually built from scratch using specialized architectures that do not benefit from past design experiences. In addition, formal models, which support research and development in most computer science subfields, are surprisingly unaccounted for within the DL literature. This lack of formality leads to branching efforts and has made interoperability one of the most crucial problems faced by the DL field.

In Model-Driven Development (MDD), models are the primary assets, from which code is generated for a particular platform. Moreover, many *User Interface Description Languages* (e.g. UsiXML, XAML and XUL) have been introduced so far that address different aspects of a *User Interface* (UI) [7]. In our framework, we follow an MDD approach for UI generation. With this purpose we introduce a new Domain Specific Visual Language (DSVL), called VisMODLE (Visual MOdeling of Digital Library Environments), for the description of the different aspects of a DL. It is made of a set of notations for describing collections, structures, services and societies – as a framework for providing formal and empirical unification of DL systems. We have created a modeling environment for VisMODLE, and a code generator that generate software tools for a given DL model. In particular, XUL code is generated for the UI in which different predefined components are invoked as specified by the models.

Our approach has the following advantages. The use of a language that is visual makes it easier for people to interpret the model, while being domain-specific leaves less room for misunderstandings. The language has been specified by metamodeling, what makes it possible to build tools providing automatic built-in syntactic and semantic checks that guide the DL designer in finding inconsistencies and gaps in the designs, and making sure that a change in a model is automatically reflected in other parts where it is relevant. In addition, although visual, the language is also formal, thus it is possible to simulate and analyze the models to demonstrate, understand and verify its behavior. Finally, deriving code from the models minimizes the number of errors in the final DL implementation.

The paper is organized as follows. Section 2 presents some related research, while in section 3 we sketch the overall

*Work supported in part by the EC's Human Potential Programme under contract HPRN-CT-2002-00275, SegraVis, and the Spanish Ministry of Science and Education, projects MD2 (TIC200303654) and MOSAIC (TSI2005-08225-C07-06).

proposed architecture. Section 4 presents the VisMODLE language, for which a modeling environment has been generated, as explained in section 5. In section 6 we show the generation of the XUL code for the DL's UI. Finally, section 7 ends with the conclusions and future work. Throughout the paper we use a toy example of a university library.

2. RELATED WORK

Formal models for DLs are rarely found. Wang [9] tried one first attempt to fill this gap. His so-called hybrid approach specifies a DL as a combination of a special-purpose database and a hypermedia-based UI, and uses this combination to formalize DLs with the Z language.

Lee et al. [5] have developed a canonical model for information systems, together with a compositional approach they applied to provide a partial solution for interoperability in DLs. Castelli et al. [1] presented works in the context of a multidimensional query language for DLs. They have described the concepts of documents developed on the notions of views and versions, metadata formats and specifications, and a first-order logic based language. Moreover, there are some declarative approaches, which are not supported by a strict underlying formal theory. These include the Digital Library Definition Language [8], and the DSpaces data model (which includes communities and bitstreams).

Examining the related bibliography we noted that there is a lack of tools or computer-aided systems, for designing and developing DL systems. Moreover, there is a need for modeling interactions among DL systems and users (as proposed in the HCI field) such as: scenario or activity-based approaches. The VisMODLE framework fills this gap providing a DSVL based approach for generating visual interaction oriented tools for DLs.

3. MODEL DRIVEN APPROACH TO DLS

Our work investigates and empirically evaluates a new MDD framework that allows DL designers to model ideas and mechanisms specific to interaction in the DL domain, and to transform them to the final (compilable) source code.

Figure 1 shows the framework. First, we have designed a DSVL, called VisMODLE, for the modeling of the different aspects of a DL. It has been defined through a metamodel (step "a" in the figure, see section 4). Next, we have generated a modeling environment for VisMODLE using the AToM^3 tool [2] (see section 5). This environment allows DL designers to visually specify DLs (step "b"). In addition, it incorporates a code generator ("LibGen") which generates code for the DL's UI and selects predefined components (from database "c") that implement the functionality of the services specified in the model (step "d").

To improve acceptability and interoperability, our framework makes flexible use of existing standard specification sub-languages for representing DL concepts. Therefore, most of the model primitives are defined as XML-based elements, which can enclose other sub-languages that help to define DL concepts. In more detail, MIME[1] types constitute the

[1]Multipurpose Internet Mail Extensions

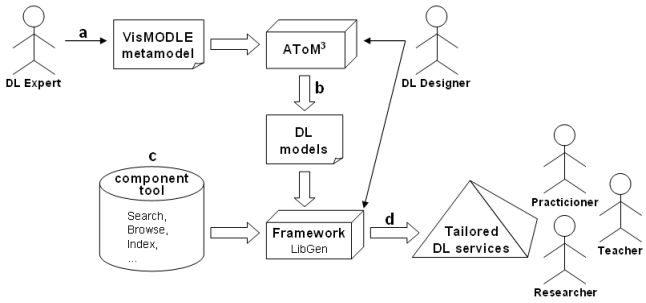

Figure 1: The Framework.

basis for encoding elements (e.g. documents) of a collection. The XML User Interface Language (XUL) is used to represent the DL visual interface. It provides a simple and portable definition of common widgets, thus drastically reducing the software development effort for visual interfaces. However, the framework is general and other UI description languages can be used.

4. A DSVL FOR DIGITAL LIBRARIES

In this section we present VisMODLE, a DSVL oriented to the design of DLs, as well as a running example, called *Library*, to show the overall process starting from the basic entities of the model. The example is kept small for presentation purposes, although our system has been able to deal with real-life DLs.

In VisMODLE the specification of a DL encompasses four complementary dimensions or viewpoints, including: multimedia information supported by the DL (Collection Model); how that information is structured and organized (Structural Model); the overall behavior of the DL (Service Model) and the different societies of actors and groups of services that act together to carry out the DL behavior (Societal Model). The complete metamodel is shown in Figure 2.

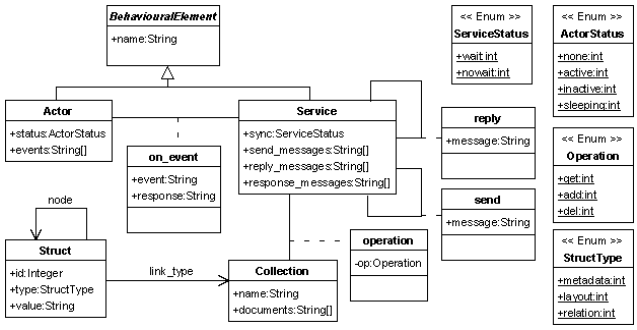

Figure 2: The VisMODLE Metamodel.

Collections are sets of elements. They can model both static (e.g. text) and dynamic content (e.g. presentation of a video). Our running example includes a collection model (not shown for space constraints) that defines a collection (called *Library*) of two documents: long1.pdf and long2.pdf.

The *Structure* specifies the way in which parts of a whole are arranged or organized. In DLs, structures can represent hypertexts, taxonomies, system connections, user relation-

ships, and containment. The window to the right in Figure 5 shows the structural model for the running example. Thus, the collection *Library* is made of documents structured with *Publication*, *Author* and *Title* metadata information (i.e. three *Struct* elements). Metadata entities are linked together with the *node* relation (organized as a tree) and linked to a collection by a metadata *link_type* relation.

Services as scenarios tell what happens to the elements in the collection and through the structures. Taken together the services describe scenarios, activities, tasks, and operations, and those ultimately specify the functionalities of a DL. Human information needs, and the processes of satisfying them in the context of DLs, are well suited to description with services, including these key types: fact-finding, learning, gathering, and exploring. In our example we make available two basic services: *Front Desk* and *Search*. The *Front Desk* is responsible for managing communications between actors and it is asynchronous (*sync* attribute is set to *nowait*), while the *Search* service executes queries on the DL and it is synchronous.

Finally, a *Society* is a set of entities and the relations between them. The entities include actors as well as hardware and software components, which either use or support services. A society is the highest-level component of a DL, which exists to serve the information needs of its entities and to describe the context of its use. DLs are used for collecting, preserving, and sharing information artifacts between society members. Figure 3 shows the societal model for our example. It involves two *Actors*: *Student* and *Librarian*. The scenario represents a *Student* trying to borrow a paper from the *Library*; he interacts with the *Front Desk* service requesting the paper and obtaining a response message about its availability. The *Front Desk* service forward the borrow request to the *Librarian* actor. Then, the *Librarian* sends a doc request message to the *Search* service, which queries the document collection (*get* operation) using metadata information provided by the borrow request, and waits the result to send back the response. The service returns an *is_available* boolean message which is propagated as a response to the *Librarian* and eventually to the *Student*.

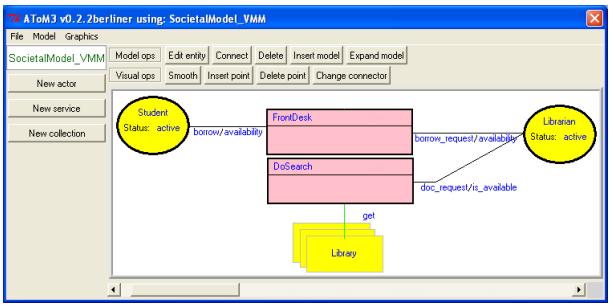

Figure 3: A *Societal* Model.

5. THE MODELING ENVIRONMENT

We have generated a modeling environment for VisMODLE using the metamodeling tool AToM^3 [2]. This is a tool for the description and automatic generation of modeling environments for DSVLs. It allows defining the DSVL syntax by means of metamodeling and model manipulation by means of graph transformation. Recently, AToM^3 has been provided with the possibility to describe multi-view DSVLs [3], such as the UML or VisMODLE. These are notations made of a set of different diagram types, each one describing a different aspect or viewpoint of the system.

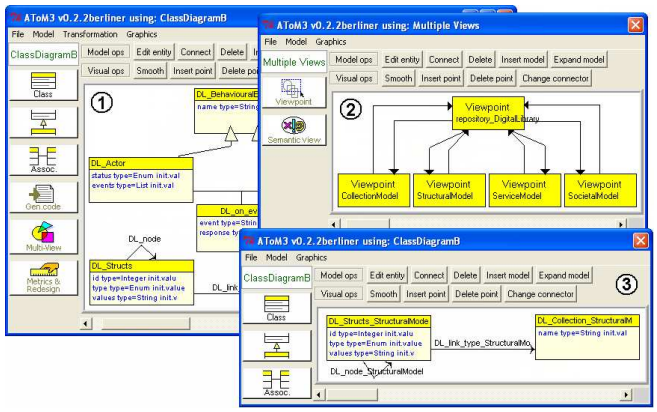

Figure 4: Defining the Environment for VisMODLE.

Figure 4 shows a moment in the definition of the modeling environment for VisMODLE. Window "1" contains the complete metamodel for the language. A tool to specify its different viewpoints is shown in window "2". We have defined four viewpoints: *Collection*, *Structural*, *Service* and *Societal*. The metamodel of each one of them is specified as a subset of the complete VisMODLE metamodel, as window "3" shows for the case of the Structural viewpoint. In addition, there is a special viewpoint which contains the full metamodel: the repository. It is used for ensuring consistency between different models. In the figure, relations between the viewpoints represent transformations, expressed by graph grammars, which are automatically generated from the metamodels for model consistency purposes [3].

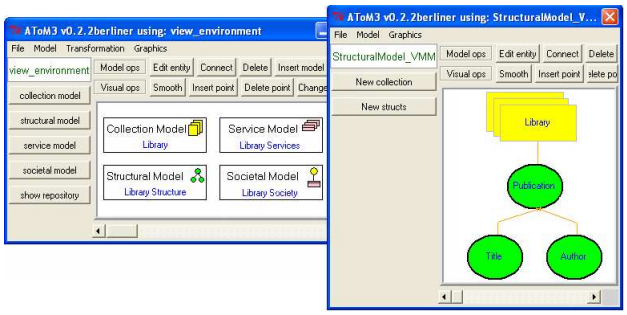

Figure 5: Modeling Environment for VisMODLE.

Figure 5 shows the generated modeling environment. The background window shows the four different views created by the DL designer for our *Library* example. The window on top shows one of them (the structural model). Note how, the generated environment allows the final user to create models, ensure their consistency, load and save projects. We have added additional functionality to the tool in order to generate XUL code for the UI and to invoke the necessary services. This is explained in the next section.

6. GENERATING THE USER INTERFACE

The model of the specific DL drives the UI generation. The UI layout is generated mainly from the information provided by the structural and collection models. In particular, code is automatically generated by "LibGen" when the DL designer modifies these types of models, thus immediately seeing the result of a change. It is also possible to produce code in a batch style. The UI events are managed by invoking the appropriate services according to the imported XUL templates, as shown in Figure 1.

The generated UI is built upon a set of XUL template files that are automatically specialized depending on the attributes and relationships designed in the modeling phase. The layout template for the UI is divided into two columns (see Figure 6, that shows the UI generated for the actor *Librarian* in our example). The left part is made of three boxes: *Collection*, *MetaData* and *MetaData Operations*. The right part manages visualization and multimedia information obtained from documents. The basic features provided with the UI templates are: document loading and visualization, and metadata organization and management.

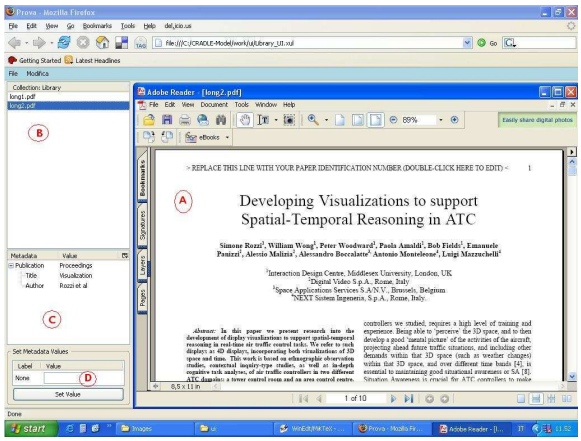

Figure 6: Generated UI for the Example.

The *Collection box* ("B" in the figure) manages the visualization of documents. The list of documents is obtained from the attribute *documents* of the collections specified in the DL design (i.e. long1.pdf and long2.pdf in the collection *Library* of our example). The visualization template works according to the (MIME) data type specified in the modeling phase. In fact, by selecting a document, the corresponding file is uploaded and visualized (starting the appropriate viewer, see "A" in the figure) within the generated UI, and in addition can be afterwards managed (print, save...).

The *MetaData box* ("C" in the figure) manages the tree structure used for representing the metadata information. The tree structure of the metadata is generated according to the metadata categorization modeled by the designer in the structural model (model to the right of Figure 5). In our example a tree has been generated with a root metadata node *Publication* that contains two children metadata nodes *Title* and *Author*. The XUL template contains all the basic layout and action features for managing a tree structure.

The *MetaData operations box* ("D" in the figure) is activated by clicking on a node of the metadata box. It manages metadata operations, such as insertion, deletion or editing.

These automatically generated interfaces have a standard template design that can be integrated by the users/designers simply adding code or additional style templates in the appropriate template placeholders. In VisMODLE, the societal model drives the linking among the visual interface templates (XUL code) and the services implementation (Java template code for browsing, indexing, searching, ...) in order to generate the full DL system (actors, services, collections and their interactions).

7. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a novel MDD approach for the generation of DLs. For this purpose, we have designed the VisMODLE DSVL, generated a customized modeling environment, and built a code generator able to produce the full application. In the present paper, we have focused on the automatic generation of the UI, using XUL.

Currently, we are working in extending VisMODLE with a behavioral diagram, and in adding analysis and simulation capabilities to the framework. Moreover, we intend to support XDoclet for the specification of the VisMODLE services. It allows automatic code generation, compliant with a standard definition which simplify coding for various technologies, such as: Java, Web Services and Web Portals.

8. REFERENCES

[1] D. Castelli, C. Meghini, and P. Pagano. Foundations of a multidimensional query language for digital libraries. In *ECDL*, pages 251–265, 2002.

[2] J. de Lara and H. Vangheluwe. AToM^3: A Tool for Multi-Formalism Modelling and Meta-Modelling. In *Proc. of FASE'2002*, volume 2306 of *LNCS*, pages 174–188. Springer, 2002.

[3] E. Guerra, P. Díaz, and J. de Lara. A formal approach to the generation of visual language environments supporting multiple views. In *VL/HCC*, pages 284–286. IEEE Computer Society, 2005.

[4] T. Kochtanek and K. Hein. Delphi study of digital libraries. *Inf. Proc. Manag.*, 35(3):245–254, 1999.

[5] S. Y. Lee, M.-L. Lee, T. W. Ling, and L. A. Kalinichenko. Designing good semi-structured databases and conceptual modeling. In *ER*, pages 131–145, 1999.

[6] J. C. R. Licklider. *Libraries of the Future*. MIT Press, Cambridge, Mass., 1965.

[7] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, and V. López-Jaquero. Usixml: A language supporting multi-path development of user interfaces. In R. Bastide, P. A. Palanque, and J. Roth, editors, *EHCI/DS-VIS*, volume 3425 of *LNCS*, pages 200–220. Springer, 2004.

[8] K. Maly, M. Zubair, H. Anan, D. Tan, and Y. Zhang. Scalable digital libraries based on ncstrl/dienst. In *ECDL*, pages 168–179, 2000.

[9] B. Wang. A hybrid system approach for supporting digital libraries. *JDL*, 2(2-3):91–110, 1999.