# A High-Level Visual Language for Generating Web Structures

Mark Minas
Computer Science Dep., University of Erlangen,
Martensstr. 3, 91058 Erlangen, Germany
minas@informatik.uni-erlangen.de

Leon Shklar
Computer Science Dep., Rutgers University,
New Brunswick, NJ 08902, USA
shklar@cs.rutgers.edu

## Abstract

*In earlier work, we discussed using data encapsulation to provide integrated access to large amounts of heterogeneous information through WWW browsers. Here, we describe a visual data modeling language which supports the generation of metadata entities to encapsulate raw data. A prototype version of an interpreter for our interactive Visual Repository Definition Language (VRDL) is now operational and may be used for building and maintaining information repositories.*

## 1. Introduction

Efforts to support search and retrieval of heterogeneous information have historically concentrated on different application areas, including software reuse, geospatial data, digital libraries [3], etc. Software reusability now extends beyond code to include other software assets such as specifications, designs, documentation, etc. [1]. In this context, we have described *InfoHarness* [6] — a system that provides rapid access to large amounts of new and existing heterogeneous information through Web browsers without any relocation or restructuring of data. *InfoHarness* imposes logical structure on raw data by performing an analysis and generating metadata to encapsulate the original information.

A textual information modeling language was originally introduced to support the generation of *InfoHarness* metadata entities [6]. Because of the success of the visual programming approach [7] with casual and inexperienced programmers, we have built a visual version of this language, the Visual Repository Definition Language (VRDL). We used the Nassi-Shneiderman diagram representation [5], which is not (as far as we know) incorporated in prominent software products, but which successfully serves as a visualization aid in teaching novice programmers. We have applied *DiaGen*, a generator for diagram editors [4], to building a syntax-directed editor for VRDL.

In the following, we briefly summarize the *InfoHarness* object model and present an example of using VRDL for building information repositories.

## 2. *InfoHarness* metadata entities

The most basic concept in *InfoHarness* is that of *encapsulation units*, which are defined as metadata entities that encapsulate portions of the original information of interest to end-users. An encapsulation unit may be associated with a file, a portion of a file, a set of files, or an operation (e.g., a database query).

An InfoHarness Object (IHO) is either a *simple object*, composed of a single encapsulation unit, or a *collection object*, composed of a set of references to other IHOs, or a *composite object*, combining a simple object and a set of references to other objects. Each object may contain an arbitrary number of additional attributes. Collection objects may contain references to independent (and even existing heterogeneous) indices that reference portions of data encapsulated by child objects.

## 3. The Visual Repository Definition Language

As a sample application, we present the judicial opinions from the U.S. Supreme Court that are available at `ftp.cwru.edu`. Here, information related to a single case may be distributed between multiple files.

Given the location of the original information, the desired run-time presentation of individual cases, and the desired indexing technology, the following steps are required to generate the repository of the Supreme Court cases: (1) Create simple objects that encapsulate individual judicial opinions (one per file). (2) Group together objects created in (1) that belong to the same case and provide for the advanced Web presentation of the case objects. (3) Use the Latent Semantic Indexing (LSI) technology [2] to create a full-text index from the textual content of objects created in (2).

These steps are implemented by the VRDL program in Fig. 1. It assumes that the LSI indexing technology is supported and that the encapsulation and presentation methods for the types 'Court' and 'Case' are available in the *InfoHarness* type library.

Each VRDL program consists of a declaration block (Pos. 0) and a statement sequence (Stmt 1–6) like a Nassi-Shneiderman diagram. Each variable is represented by an

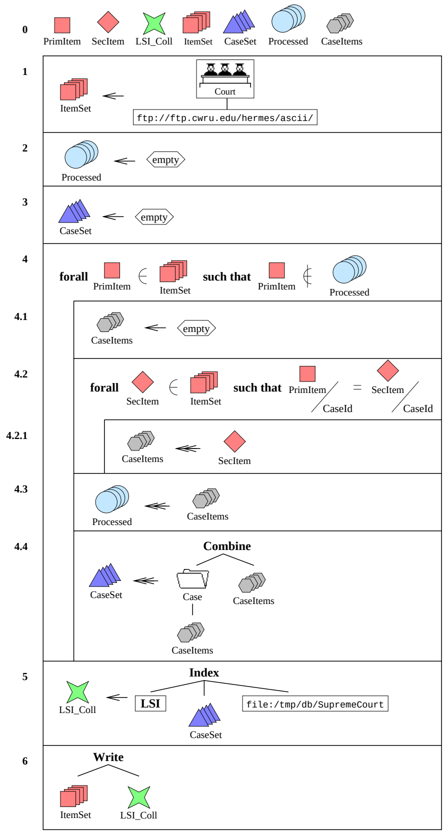

**Figure 1. VRDL program for representing the U.S. Supreme Court cases at `ftp.cwru.edu`.**

icon along with its name. Shape and color provide easy distinctions between different variables. Set objects are indicated by stacks of icons. The following statements implement step (1) in stmt 1, step (2) in stmts 2–4, and step (3) in stmt 5. Stmt 6 writes out the results, which build the information repository properly formatted for Web use.

Stmt 1 serves to encapsulate individual opinions located at `ftp://ftp.cwru.edu/hermes/ascii/` and assigns the generated set of simple objects to set variable 'ItemSet'. In general, *encapsulation* expressions create simple IHOs by analyzing raw data.

Stmts 2 and 3 initialize the set variables 'Processed' and 'CaseSet'. 'Processed' is used to accumulate objects from 'ItemSet' which are already processed. 'CaseSet' is used to group together objects that encapsulate individual cases.

Stmt 4 is an iteration statement providing selective access to set members. Here, the forall-body (stmts 4.1–4.4) is executed for every element of 'ItemSet' that has not been yet considered and is not contained in 'Processed'. For each iteration, an element of 'ItemSet' is assigned to 'PrimItem'.

Elements of 'ItemSet' that belong to the same case as 'PrimItem' are selected in stmt 4.2 using the 'CaseId' attribute, which is set by the encapsulation method for the type 'Court'. Consider that stmt 4.2.1 *adds* 'SecItem' to 'CaseItems'. Stmt 4.3 updates set variable 'Processed' in order to exclude these IHOs from further consideration. Stmt 4.4 then creates composite objects that both encapsulate all case-related information and contain references to simple objects encapsulating individual opinions. The presentation method for type 'Case' is responsible for generating the internal hyperlinks to individual opinions. The *combine* expression is used for building composite objects by combining a simple object with a set of references to other objects.

Stmt 5, when all composite case objects are grouped together, creates an indexed collection for objects in 'CaseSet'. Latent Semantic Indexing is used as an indexing scheme. In general, an *index* expression creates collection objects that contain a set of references to other IHOs and to a searchable index.

## 4. Conclusions

We have briefly introduced VRDL, a visual language describing how to build Web information repositories from large amounts of heterogeneous data. VRDL is the result of our efforts to design a simple, yet powerful, language that supports modeling heterogeneous information based on the underlying notions of sets and types, and to make using the language simple enough for unsophisticated programmers. Whereas the concept of a simple, declarative, textual language to support modeling is not new[1], the visual language is a unique feature increasing user-friendliness.

## References

[1] V. Basili. Support for comprehensive reuse. *Software Engineering Journal*, pages 303–316, 1991.

[2] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Hashman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6), 1990.

[3] C. Fisher, J. Frew, M. Larsgaard, T. Smith, and Q. Zheng. Alexandria digital library: Rapid prototype and metadata schema. In *Advances in Digital Libraries*, pages 173–194. Springer-Verlag, New York, 1995.

[4] M. Minas and G. Viehstaedt. DiaGen: A generator for diagram editors providing direct manipulation and execution of diagrams. *Proc. VL '95, Darmstadt, Germany*, pages 203–210. Sept. 1995.

[5] I. Nassi and B. Shneiderman. Flowchart techniques for structured programming. *ACM SIGPLAN Notices*, 8(8):12–26, Aug. 1973.

[6] L. Shklar, K. Shah, and C. Basu. Putting legacy data on the Web: A repository definition language. *Computer Networks and ISDN Systems*, 27(6):939–952, Apr. 1995.

[7] N. Shu. *Visual Programming*. Van Nostrand Reinhold, New York, 1988.

[1]For a detailed discussion of related languages see [6].