Tower Patterns for Web Hyperdocument Framework Design

R. V. Snytsar, V. B. Filatov
Interna Ltd.,
Moscow, Russia.
E-mail: [roy,vf]@rrg.msk.su

Abstract

Development of an advanced Web-based information system often demands introduction of several new document types. To introduce a new document type, system designer needs to develop a couple of applications responsible for different aspects of document manipulation, namely, storage, authoring, retrieval and browsing, and therefore do a lot of programming. The objective of this paper is to provide recipes for reduction of programming efforts in hypermedia system design. This paper suggests that a modification of a Booch-like diagramming technique that takes into account distribution of document functionality between several applications enables us to develop patterns for hyperdocument design. We will refer to these patterns as 'tower patterns' to distinguish them from lower-level design patterns. We provide a collection of tower patterns useful for object-oriented hypermedia management system design. Some patterns cover scenarios for processing statically stored documents while some for dynamically created ones.

1. Introduction

When designing new document type for Web presentation, one needs to make the following decisions.

- Determine what document formats and what databases may be presented in the publication.
- Decide what applications for document and database manipulation should be written or reused.
- Choose what plugin applications for client side presentation of your documents should you write.
- Connect to the Net and set up your Web server.
- Create CGI scripts that will access databases or convert documents into HTML.
- Publish your information.

Usually it is convenient for system designer to consider his efforts to be distributed over four levels of WWW architecture [Dre87].

- The first level is a description of the ***document storage formats***.
- The second level is formed by applications for ***document authoring and manipulation***.
- ***Agents level*** it the third one. This level consists of Web server software and CGI scripts.
- The fourth level is a set of tools for visual object presentation on user machine or ***browser level***.

Web is a disintegrated environment for data publishing. One document may be represented by different applications on different system levels and on different machines in the Net. On the other hand, different document types may be processed by applications with similar behavior. Thus, problem of software design reuse becomes crucial.

For easy implementation of this structure first of all one needs a set of classes responsible for document storage, presentation and manipulation. But to obtain a framework suitable for document type design one also have to specify modeling constructs that describe how individual classes, instances and applications interact on different levels of hypermedia system.

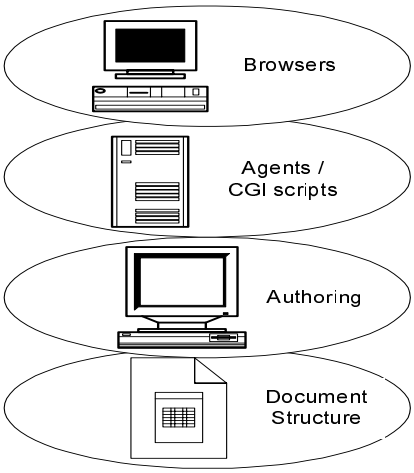

Figure 1.WWW architecture levels.

The usual technique for software components reuse is the introduction of abstract classes, while frameworks are a technique for reuse of entire application and subsystems. The popular method for description of application frameworks is pattern language[Joh92]. We wish to use patterns at a larger scale of hypermedia system. These patterns should describe experience in reuse of applications ensemble that process a document on different levels of Web architecture.

Several approaches to the development of pattern languages for hypermedia system design have already been considered by different researchers.

Schwabe et al [SR94] presented an object-oriented method for designing hypermedia applications. They pointed out four steps of hypermedia system design and developed patterns to be used during all design steps.

Buhr [Buhr96] noticed that description of distributed system in terms of patterns requires paradigm shift from the object-oriented design patterns in the “gang-of-four” style [GHJV93] and proposed the use case maps as behavioral patterns for client-server system.

De Bra et al. [Bra92] proposed a data model that is intended to serve as the basis for analysis of hyperdocument structure as well as viewing and processing tools (the tower model).

We suppose that the notion of the tower as an application ensemble that covers all aspects of hyperdocument functionality is an excellent basis for higher-level analysis and provides a suitable paradigm for the development of patterns collection for hypermedia system design.

The rest of the paper is organized as follows. Chapter 2 introduces a catalog of tower patterns. The next chapter provides a sketch of a data model that may be useful for analysis of desired Web application architecture. Then we conclude with a brief discussion of our patterns collection.

2. Tower patterns collection

Now we should discuss the most useful tower patterns and their reusability issues. Each pattern imposes using of a specific scenario of hyperdocument processing during its life cycle, namely, storage, authoring, retrieval and browsing.

To describe the pattern we use Tower Pattern Template that is close to template format proposed in [MD96]

Intent

What does the tower pattern do? What is its rationale and intent? What particular design issue or problem does it address?

Problem

The specific problem to be solved.

Motivation

A scenario in which the pattern is applicable.

Forces

Considerations that must be taken into account when choosing a solution to the problem.

Solution

The proposed solution to the problem.

Participants

Describe the classes and/or objects participating in the tower pattern and their responsibilities.

Diagram

A graphical representation of the pattern using a notation based on Booch notation. We denote an instance of a document-specific subclass by cloud, instance of reusable class by dark cloud, usage relation by simple arrow and aggregation relation by arrow with diamond. Clouds are hosted on different floors of a tower to emphasize their distribution on the different layers of Web architecture. Every object is placed on the lowest floor it is used on. Reference from one tower floor to another means reuse of the class already implemented on lower level of system design. Each floor designates an application that is created using some existing class architecture.

Consequences

What are trade-offs and results of using a pattern? What benefits do you get when follow the pattern?

Implementation issues

How do existent frameworks fit this pattern? How can the introduced classes be quickly implemented on the basis of the existent framework? Since the presented patterns were discovered while we were attempting to reuse MFC-based Document and View classes, we consider Microsoft Foundation Classes (MFC) architecture as the underlying level for our design.

2.1 Pattern: HTML Specialty

Intent

HTML Specialty document type encapsulates HTML document structure and provides alternative view for the document for performing higher-level operations on it.

Problem

You want a specific HTML document type.

Motivation

HTML Specialty pattern defines framework for additional view of an HTML document.

Well-known example of the document of HTML Specialty type is a Netscape Bookmark file. Although it is usual HTML file that contains unordered list tags, Netscape provides special tree-view window for easy manipulation and organization of the bookmarks.

Let us consider more typical situation that stimulates introduction of this pattern. Suppose you supervise development of a database where documents are bibliography references presented in the HTML format. You wish that these documents follow some template (for example, name of periodical should be in bold font). So you can either review and reformat all documents created by different people or create specialized HTML editor to be used for your documents creation instead of the common editor. This editor presents a document as a form with editable fields. When writing document to the disk, the editor uses proper formatting tags and style sheet.

Another example is a helper application that drives author to use the HTML pattern language [Ore95] by propagating pattern approach to document structure description.

Forces

- It will be easier to provide homogeneous formatting for similar documents using such a specific document type.
- The information is already well structured.

Solution

Create a specific editor program to create and manipulate these documents. The output of the program should be in HTML format.

Participants

- **HTML document** encapsulates document structure.
- **application** provides specific view class responsible for user interface and manipulation with document object.

Diagram

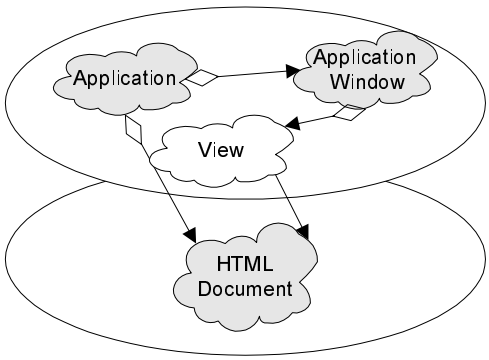

Figure 2. HTML Specialty pattern

Consequences

- You get consistent formatting of the structured information.
- Building a specific HTML editor may be very expensive.
- Once written HTML document class can be effectively reused.

Implementation issues

This pattern drives us to the first obvious but important design decision. First of all hyperdocument development framework should be powered by reusable HTML document class. Development of the powerful CDocument-derived class responsible for HTML parsing

and manipulation allows system designer spare minor efforts introducing alternative views for HTML documents.

2.2 Pattern: MIME Type

Intent

MIME Type pattern describes a type of document that is transmitted to user machine as is and requires special browser extension.

Problem

You want user to receive and manipulate your documents in original format.

Motivation

There exist many document formats that cannot be directly mapped into HTML. Sometimes it is desirable to perform specific operations on document on a client machine.

For example, you developed an application to view and edit video clips in your special format. Now you wish to publish your clips on the Web. Clips cannot be converted to HTML. So, a common solution is a plugin application. It is desirable to reuse code from the editor application in the plugin.

Forces

- It is hard to convert your documents into HTML.
- User needs to manipulate your document in specific way.
- You have already written editor or viewer application for your documents.

Solution

Create a plugin application.

Participants

- **document** encapsulates document structure.
- **application** provides user interface and manipulates document object.
- **plugin** provides user interface and manipulates document object on client machine.

Diagram

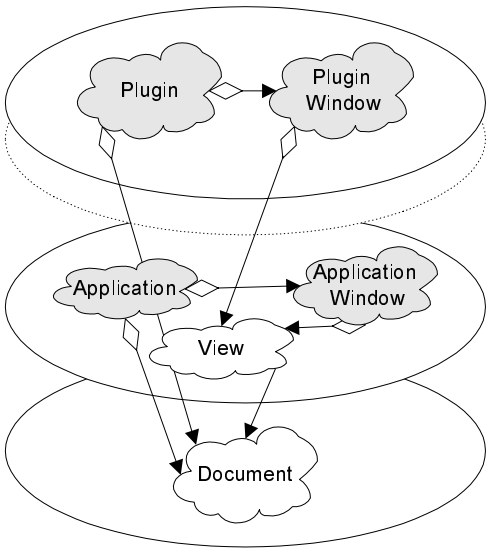

Figure 3. MIME pattern

Consequences

- You get consistent presentation of your documents on the client side.
- Client needs to download and install your plugin application.
- Plugin application is platform-dependent.

Implementation issues

The most popular browser technology at present - Netscape Plugin framework[NNP96] - proposes reuse of the Document-View pattern in both editor and plugin applications. However, classes that implements this patterns in editor and plugin differ and may not be reused. So an important design decision would be to coordinate editor and plugin protocols in such way that they can use the same Document and View classes. It means that after implementing editor classes we can get plugin application for free. This approach is implemented in ActiveX extensions framework from MFC 4.2 [Rau96]. Thus, choosing Microsoft Internet Explorer browsing technology allows higher level of code reuse and cheaper implementation.

2.3 Pattern: Database Shortcut

Intent

Database Shortcut embodies a virtual object, allowing hyperdocument author to inspect virtual objects without involving Web server software.

Problem

You wish to refer to a single record in database in the same way as to a standalone document.

Motivation

Hyperdocuments dynamically created from the database query results (virtual objects) are hard to manipulate. When author wishes to refer to a virtual object, he has to specify a reference to a CGI script along with script parameters. It's inconvenient to remember and type script parameters manually.

We propose another scenario that allows to provide more automation in the authoring process. First of all an author uses an application that allows him to query and browse database records. When a user selects a record he wishes to refer to as a virtual object, he saves on disk a tiny file (shortcut) containing query parameters that identify this record. Then user refers to the shortcut instead of a CGI script. When the shortcut is requested from a client machine, a server filter performs the query using shortcut data as query parameters and sends the resulting virtual object back to client machine.

Forces

- It is desirable to refer to either documents stored in database or standalone documents in uniform way.
- You have already written an application that manages documents stored in database.

Solution

Enhance the database management application with ability to create shortcuts to single records and create CGI script that will extract a record on the user request for shortcut.

Participants

- **extractor** creates virtual object.

- **shortcut** stores brief description of virtual object in a file.
- **application** provides user interface and manipulates virtual object.
- **filter** transmits virtual object on user request.

Diagram

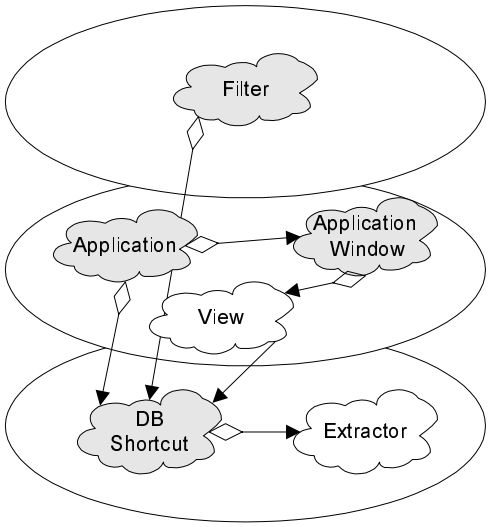

Figure 4. Database Shortcut pattern.

Consequences

Database Shortcut pattern is used for implementation of a very interesting document type that serves as a reference to database query result. Database shortcut allows information system author to become acquainted with virtual object regardless the filter software. For example, database author can review virtual object contents without access to Web server.

Implementation issues

The most important design issue is reuse of Extractor class in viewer. and in filter. It is often convenient to derive CDbShortcut class from CDocument and CExtractor class from CRecordset. Thus, it take minor efforts to implement this pattern on the basis of MFC framework. Note that if we need client-side browsing facility, View class may be also reused.

2.4 Pattern: Searchable Index

Intent

Searchable Index pattern describes document that is not transmitted to user machine because of its size but its pieces are available on user request.

Problem

You wish to give to user an ability to query for parts of a document instead of the whole large document.

Motivation

Some documents, like indices, are too large to be transferred to client machine. However we can enable user to specify what part of the document he is interested in. Depending on user request, server will dynamically generate document containing relevant information.

Forces

- Accessing large documents can slow down user interaction with you Web site.

- Before querying for the parts of the document user should make sense of document's contents.

Solution

Create a CGI script that will provide a query form as a result of user request for the whole document. The form page should contain the information that aids user to make a productive queries.

Participants

- **index** stores data.
- **index filter** sends query form as index request result.
- **query form** passes query parameters back to server.
- **query processor** performs query with user-defined parameters and returns resulting dynamically created document back to user.

Diagram

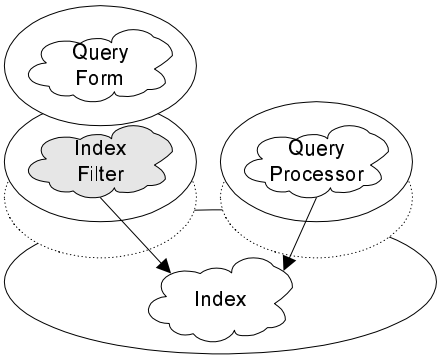

Figure 5. Searchable Index pattern

Consequences

Using this pattern may dramatically speed up system reaction on user requests.

Implementation issues

Searchable index scenario can be easily implemented using ISAPI Server Extension classes from MFC 4.1 [Bla96].

3. Tower model review

To perform the perfect design of data management system we should start with an appropriate data model. Basing on this model, we can describe our design using patterns on a higher level of abstraction than well-known object-oriented design patterns.

De Bra et al. [Bra92] proposed an object-oriented data model, namely, tower model, that provides constructs and can serve as a foundation for hyperdocument types design. The data model is made of two layers.

The lowest layer defines first-class objects within hyperdocument that are nodes, links and anchors.

The highest layer defines constructs that build complex information structures from the simple ones. These constructs are:

- The **composite object** construct describes hyperdocument structure in terms of nodes, links and anchors.
- The **tower** construct packages different levels of hyperdocument description, such as document structure level and visual presentation level.
- The **city** construct describes multiple perspectives of a single document.

So, let us illustrate the tower model by it's application to the description of the HTML document type.

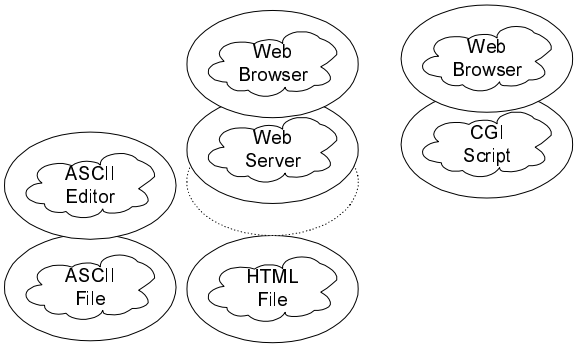

Figure 6. HTML city and virtual object tower.

The lowest layer of data model is the semantics of nodes, links and anchors. This semantics is pre-defined by choosing WWW as hypermedia engine [CG88].

- On document structure level a HTML document has two perspectives. The first one is an ASCII file. The second one is a composite of tags as defined in HTML specification [HTML96].
- On document manipulation level we can use ASCII text editor to produce HTML documents.
- On agent level HTTP daemon transmits HTML file as it is.
- On user-machine level Web browser renders HTML to produce visual presentation.

Though HTML document has two different perspectives, it is modeled as a city.

This city can evolve in two directions. On the one hand, we can use a WYSIWYG HTML editor and disregard the ASCII file tower. On the other hand, document templates and wizards cause a diversity of document perspectives and further "urbanization" of the HTML document model.

Another important notion in tower model is virtual object that is not stored but is generated by the agent. For example, database query results, are not hosted as separate documents. They are generated by CGI scripts. Their virtuality can be easily understood from the figure 6: virtual object are not based on the ground of the file system.

4. Conclusion

We have presented a collection of design patterns that make it possible to create a variety of document types for Web-based hypermedia system.

Some of the scenarios presented in this paper are well known to web programmers. The notion of database shortcut that materializes virtual object is novel. It leads to similar representation and user interface for both real and virtual documents.

The collection of tower patterns provides excellent basis for hyperdocument application framework creation. Dark clouds on patterns are reusable classes and white clouds are abstract

classes in a framework. Concrete class implementations may use lower-level object-oriented design patterns [Joh92].

We suppose that our collection of tower patterns may serve as a foundation for the research of the pattern language describing experience in building Web-based information systems.

5. Acknowledgments

The authors would like to thank Vsevolod Ilyushchenko who provided feedback on early versions of this paper. Special thanks to Peter Sommerlad who's "shepherding" of this paper helped authors to improve the patterns and present them in much more consistent way.

References

[Ber89] Berners-Lee T., Information Management: A Proposal. Available at http://www.w3.org/pub/WWW/History/1989/proposal.html.

[Bla96] Blaszczak M., Writing Interactive Web Apps is a Piece of Cake with the New ISAPI Classes in MFC 4.1, Microsoft Systems Journal, Vol. 11, No. 5.

[Bra92] De Bra, P., Houben, G.J., Kornatzky, Y., An Extensible Data Model for Hyperdocuments, 4th ACM Conference on Hypertext, Milan, December 1992, pp. 222-231.

[Buhr96] Buhr R.J.A., Design pattern at Different Scales, Presented at EuroPlop'96, Kloster Irsee, Germany, July 11-13, 1996.

[CG88] Campbell, B., Goodman, J.M., HAM: A general purpose Hypertext Abstract Machine, CACM, 31:7, July 1988, pp. 856-861.

[Dre87] Drexler K.E., Hypertext Publishing and the Evolution of Knowledge, Social Intelligence, Vol. 1, No. 2, pp.87-120.

[GHJV93] Gamma E., Helm R., Johnson R, Vlissides J., Design Patterns: Abstraction and Reuse of Object-Oriented Design, European Conference on Object-Oriented Programming, Kaiserlauten, Germany, July 1993. Published as Lecture notes in Computer Science #707, pp. 406-431, Springer-Verlag.

[HTML96] Introducing HTML 3.2. Available at http://www.w3.org/pub/WWW/MarkUp/Wilbur/

[Joh92] Johnson R. E., Documenting Frameworks with Patterns, OOPSLA '92 Proceedings, SIGPLAN Notices, 27(10): 63-76, Vancouver BC, October 1992.

[JR91] Johnson R. E., Russo V. F., Reusing Object-Oriented Design, Univ. of Illinois tech report UIUCDCS 91-1696.

[MD96] Meszaros G., Doble J., MetaPatterns: A Pattern Language for Pattern Writing, Presented at EuroPlop'96, Kloster Irsee, Germany, July 11-13, 1996.

[NNP96] Netscape Navigator Live Connect/Plug-in Software Development Kit. Available at http://home.netscape.com/comprod/development_partners/plugin_api/index.html

[Ore95] Orenstein O., Html Pattern Language. Available at http://www.anamorph.com/docs/patterns/newdescrip.html

[Rau96] Rauch S., Unified Browsing With ActiveX Extensions Brings the Internet to Your Desktop, Microsoft Systems Journal, Vol. 11, No. 9.

[SR94] Schwabe D., Rossi G., From Domain Models to Hypermedia Applications: an Object-Oriented Approach.