Separating the navigational aspect

Antonia M. Reina Quintero, Jesus Torres Valderrama
Languages and Computing Systems Department
University of Seville
Avda. Reina Mercedes, s/n, Sevilla, Spain
{reinaqu,jtorres}@lsi.us.es

Abstract

The first step given to separate concepts in web environments has been to take apart presentation from data. This split has been gotten due to the appearance of the Extensible Mark-up Language (XML) and the application of style sheets. The new ideas from the advanced separation of concerns community and the new abstractions like aspects make us think this original division isn't rich enough. There are important concepts of Internet applications that should be defined separately. If we look at new web design methodologies, we can realize that one of these aspects is navigation. Following the way started by XML, we propose the use of the XML Linking Language (XLink) as a first stage to obtain the separation of the navigational aspect.

1. Introduction

The divide-and-win principle is one of the basis of software engineering. Some of the results of this principle are the ideas that have been arisen in the field of advanced separation of concerns,whose most representative example is the notion of aspect.

An aspect is a new abstraction to wrap concerns that are scattered all over the program code.

In web application environments a separation between presentation and data has been achieved. This division can be gotten due to the appearance of the Extensible Markup Language (XML) [3] and style sheets (CSS [2], XSL [5]).

If we analyze the trends of the last web design methodologies (OOHDM [14], RMM [10], HDM [9] and so on), we will realize that a very important concept involved with web applications is navigation. The term navigation includes the intrinsic notion of movement through an information space [4], where the current position or context is a crucial factor. That is because many times the next page to visit while you are navigating will depend on the previous navigation.

Web design methodologies treat navigation separately. This fact demonstrates that navigation should be kept apart from other concerns and reinforces the idea that the division between presentation and data isn't enough.

In this paper we propose the separation of the navigational aspect. As a first stage to get this division we use the XML Linking Language to specify links keeping them apart from data and presentation.

In section Navigation we define navigation concept as it is understood in the scope of this paper. Section 3 is a review of the state of the art in the separation of concerns area. Afterwards we introduce the approaches proposed by the last design methodologies to model navigation. Then we discuss an example to illustrate some navigation problems and how the separated specification of links could help us to handle this problems. Later we propose a solution to separate the link structure using XLink. And finally we summarize and conclude the paper.

2. Navigation

Navigation is a concept widely used in web environments. Its original meaning was bound to the semantic defined in hypertext, that is, it represented the action of jumping from page to page through a hyperlink. The advance of technology has made us have a wider vision of the navigation concept, being not only the action of jumping from page to page, but the idea of moving through an information space.

This extension of the semantic of the navigation concept implies that we do not consider all the links as part of the navigational process. For example, when we are using a search machine like google or altavista, and we submit a query, we obtain a web page as the result. This web page usually has a few links on the bottom which are pointing to other web pages with more results. We do not think that we are navigating when we push on one of these specific links, since we are not moving from an information space to another one. These links are just a way to do scrolling.

Context is a very important concept in navigation. Let us suppose we have a web-based application for a museum, and we want some information on a concrete painting. If we got the information navigating through the author, and then we push on a link Next, we will move to the next painting by the same author. However, if we got the painting through a pictorial movement, the result of the navigation will be different. In this case, we should move to another painting related to the same movement.

In [13] navigation is defined as the sensation that the user has when he navigates through an object space from the application domain, but distinguishing these objects from conceptual objects, as they are customized according to the user's profile and the tasks that are being made.

3. Separation of concerns

One of the approaches that have emerged strongly in the last few years is separation of concerns, a principle which has being applied in the area of software engineering practically from its very beginnings. However, the current technology has been focused mainly on the programming level. The aspect-oriented programming (AOP) [8] proposal is that it is possible to make programs better if you specify separately the different concerns, properties or areas of interest of a system, and the description of their relationships, leaving the weaving or composition tasks to the AOP mechanisms (Figure 1).

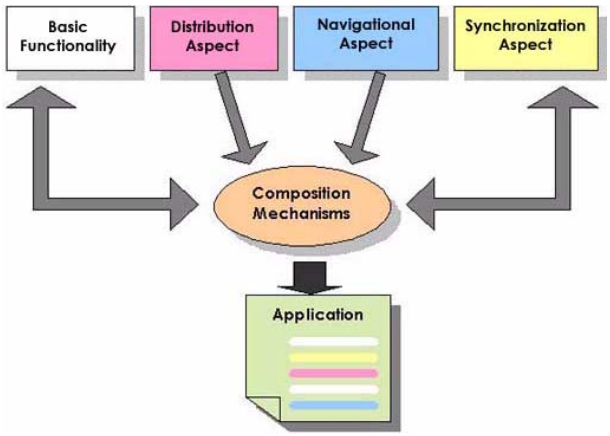

Figure 1. Aspect-oriented programming mechanisms.

Although generally speaking we know it as aspect-oriented programming, there have arisen a few different approaches in different researching groups. The most important are: adaptive methods [12], which can be implemented using the DJ library (http://www.ccs.neu.edu/research/demeter/DJ), multidimensional separation of concerns [15], whose result has been Hyper/J (http://www.research.ibm.com/hyperspace/HyperJ/HyperJ.htm), composition filters [1], and aspect-oriented programming [11], whose main tool is AspectJ(http://www.aspectj.org).

AspectJ is an aspect-oriented, general purpose extension to Java. It tries to abstract the concerns that are scattered all over the program code. These concepts are known as aspects, and they use pointcuts to mix them with the code that implements the basic functionality . Pointcuts are points where the code that implements the basic functionality can be augmented, either before, or after the join point. This code is wrapped into an aspect.

Adaptive methods encapsulate the behavior of an operation in a concrete place, avoiding the scattering problem, abstracting over the structure of classes, and avoiding the tangling problem as well.

To achieve its objectives, adaptive methods express the behavior like a high level description of how reaching to the participants of the calculation (called traversal strategy) and what to do when each participant has gotten it (known as adaptive visitor). As a result, this group of research has developed the DJ (Demeter/Java) library, which is a Java package that gives us some tools to interpret the traversal strategy and the adaptive visitor.

Multidimensional separation of concerns allows us to encapsulate any arbitrary class of concepts simultaneously, and to integrate these concepts. This separation is based on the idea of hyperspaces. They have made a tool called Hyper/J, which doesn't extend Java, as Aspect/J does. Hyper/J works with .class files, not with source files.

4. Navigation in methodologies

One of the pioneer methodologies in introducing navigation as a different aspect to treat during the design stage of the development of a web application has been HDM [9]. This methodology presented some new primitives which have served as basis to many methodologies that came later.

OOHDM (Object-Oriented Hypermedia Design Model) may be one of the most consolidated web design methodologies. It considers navigation as a critical step in the design of a hypermedia application [13]. During the design phase a model is constructed. This model is a view of the classes from the conceptual model, so you can construct different navigation models for the same conceptual model.

A few primitives are used to define navigation: nodes (they are views of the conceptual classes), links (they are views of the relationships from de conceptual schema) and access structures (they are alternative ways to navigate, as indexes, guided tours and indexed guided tours).

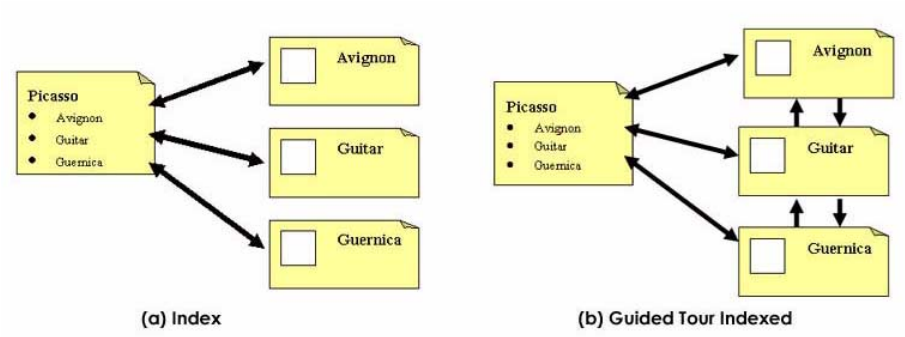

Figure 2. Aspect-oriented programming mechanisms.

Apart from these primitives, which already appeared in the first methodologies that treated navigation, OOHDM has contributed with navigational context, a new primitive to structure the navigational space.

Navigational context is a set of nodes, links, context classes and other navigational contexts that are used to organize navigation space in consistent sets that can be traversed following a particular order.

5. Navigation and aspects

Since navigation is an aspect that already has been treated separately by the new design methodologies for the development of web and multimedia applications, we think the next step is to bridge the gap between design and implementation and take this separation to the latest stages of a software project development. In order to obtain this separation at the programming level, aspect-oriented programming and multidimensional separation of concerns have arisen. It is interesting to do a study to cross the gap between design and implementation using this technology.

We will introduce the next example to study what advantages we will obtain if we succeed separating clearly the navigational aspect. Let us suppose a web application has been made for a museum, and one of the initial requirements was you should navigate from a painter to all his paintings.

To implement this requirement, you decided to use an index access structure. So, you had to implement something like it is shown in (Figure 2(a)). Later, when a prototype of the application was shown to the customer, he decided he also wanted to navigate from one painting to another painting by the same author. This new requirement made you modify the design, because the access structure we had chosen wasn't the best one, and an Indexed Guided Tour (Figure 2(b)) was chosen to fulfill this new requirement.

Such a conceptually simple change (the access structure modification) can be an arduous and tedious work when you

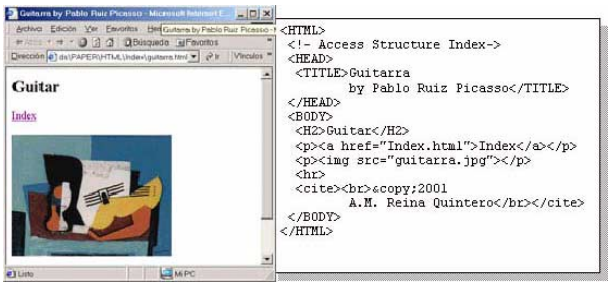

Figure 3. HTML page (code and picture) implementing the Guitar node using an Index access structure.

modify the application. In Figure 3 we show the implementation of the original Guitar node (with the access structure Index), and in Figure 4 we depict the same node but with the Indexed Guided Tour access structure. We have emphasized the HTML code that we have to add to implement the new access structure using bold fonts. Although they seem only two lines of HTML code, it should have been noticed that this web page is very simple (we decided to design it so simply to appreciate clearly the problem) and, also, you should notice this isn't the only page we have to modify. We have to change all the nodes of the context (Guernica and Avignon, in this case).

Figure 5 describes the Index implementation classes (Figure 5 (a)) and the Indexed Guided Tour implementation classes (Figure 5(b)).

In this paper we propose something like it is described in Figure 6 to separate the navigational elements completely. If we define these elements as an aspect, we can use the aspect-oriented mechanisms to specify the new access structure somehow, and weave it with the basic functionality.

If we look at how most of the aspect-oriented tools work, we will deduce the following things:

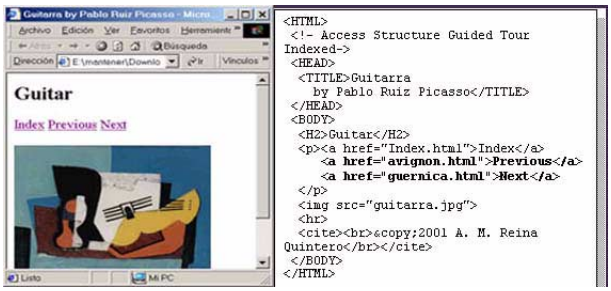

Figure 4. HTML page (code and picture) implementing the Guitar node using an Guided Tour Indexed access structure.

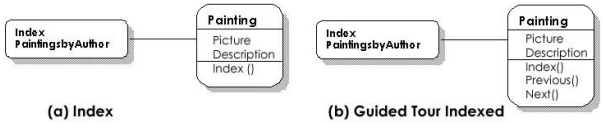

Figure 5. Implementation classes.

1. Somehow we should describe the main functionality of the application. We should implement the conceptual model.

2. On the other hand, we should define navigation separately. Are the current aspect-oriented tools powerful enough to obtain this separation at the implementation level? And, if they are, which of the approaches adjusts better to the definition of this aspect?

3. We should look for one or many join points, that means, where are we going to join the navigation aspect with the classes of the conceptual model?

4. We should implement a composition mechanism to make functionality and navigation become one program. How can we mix both concepts?

6. A first stage to separate navigation

With the appearance of the Extensible Mark-up Language (XML) [3] and the Extensible Stylesheet Language (XSL) [5] we have obtained a separation between presentation and content. Nowadays there are some new specifications coming from the W3C Consortium like XLink [7] and XPointer [6] which are based on XML and can help us to separate navigation.

XLink is an specification which defines the way an XML document should be linked, whereas XPointer describes how to point to a concrete place in a document. In other words, XLink determines the document to access and XPointer determines the exact point in the document.

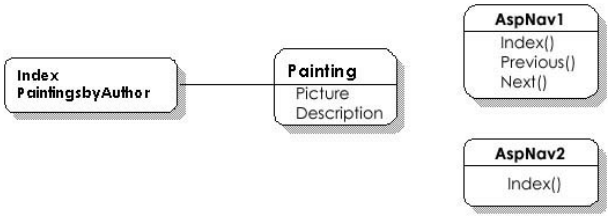

Figure 6. Separation of navigational aspect.

In this paper we propose these new specification languages as a first stage to get the desired separation between navigation and basic functionality. To face this problem, we propose the use of the new link definitions specified in XLink, in order that we can obtain data in one or more XML files, on the one hand, and links in another XML file, on the other hand.

If we apply this principle to the example proposed in section Navigation and aspects, the files depicted in Figure 7, Figure 8 and Figure 9 will be produced. On the one hand, we have the data in the files picasso.xml, and avignon.xml. On the other hand, we have defined the links among the different XML files in the link.xml file.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<?XML-STYLESHEET type="text/xsl" href="painter.xsl" ?>
<!DOCTYPE PAINTER SYSTEM "painter.dtd">

<PAINTER>
   <SURNAME>Ruiz Picasso</SURNAME>
   <NAME>Pablo</NAME>
</PAINTER>
```

Figure 7. File picasso.xml.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<?XML-STYLESHEET type="text/xsl" href="painting.xsl" ?>
<!DOCTYPE PAINTER SYSTEM "painting.dtd">
<PAINTING>
   <NAME>Le Demoiselles d'Avignon</NAME>
   <MOVEMENT>Cubist</MOVEMENT>
   <MATERIAL>Oil on canvas</MATERIAL>
   <HEIGHT>233.7 cm</HEIGHT>
   <LENGTH>243.9 cm</LENGTH>
</PAINTING>
```

Figure 8. File avignon.xml.

The main disadvantage of the use of this technology is its youth, because the browsers aren't ready to work with XLink yet. So we can not appreciate the effect of the mixture of data, presentation and links.

7. Conclusions and further work

If we pay attention to the current approaches in web design methodologies, we can realize that navigation is a

```
<paintings xmlns:link="http://www.w3.org/1999/xlink"
           xlink:type="extended">
    <painter xlink:type="locator"
             xlink:label="painter"
             xlink:href="picasso.xml"/>
    <painting xlink:type="locator"
              xlink:label="painting"
              xlink:href="guernica.xml"/>
    <painting xlink:type="locator"
              xlink:label="painting"
              xlink:href="avignon.xml"/>
    <painting xlink:type="locator"
              xlink:label="painting"
              xlink:href="guitarra.xml"/>
    <link xlink:type="arc"
          xlink:from="painter"
          xlink:to="painting">
</paintings>
```

Figure 9. File links.xml.

key concept in this kind of applications. The most recent methodologies treat it separately, spending a whole stage to design the navigational structure and creating different models, one for conceptual classes and another one for navigational classes.

This separation should be taken from design to implementation. We could use separation of concerns technologies to accomplish it. We propose to make a study to check if aspect-oriented approaches fit well with the separation of this concern. If they do, which of them fits better.

We consider that a first stage to obtain a separation of navigation is to take apart links and we could specify them separately with XLink.

We want to study if the primitives that are used by web design methodologies can be defined using these technologies and how aspect-oriented languages can be embedded in web pages and web applications.

References

[1] L. Bergmans and M. Aksits. Composing Crosscutting Concerns Using Composition Filters. *Communications of the ACM*, 44(10):51–57, October 2001.

[2] B. Bos, B. W. Lie, C. Lilley, and I. Jacobs. Cascading Style Sheets, level 2. CSS2 Specification. W3C Recommendation. (http://www.w3.org/TR/REC-CSS2), May 1998.

[3] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. (http://www.w3.org/TR/REC-xml), October 2000.

[4] D. Cunliffe, C. Taylor, and D. Tudhope. Query-based Navigation in Semantically Indexed Hypermedia. In *Proceedings of the Hypertext'97 Conference*, 1997.

[5] S. Deach. Extensible Stylesheet Language (XSL) Specification. (http://www.w3.org/TR/WD-xsl), April 1999.

[6] S. DeRose, E. Maler, and R. Daniel. XML Pointer Language (XPointer) 1.0. W3C Recommendation. (http://www.w3.org/TR/xptr/), September 2001.

[7] S. DeRose, E. Maler, and D. Orchad. XML Linking Language (XLink) 1.0. W3C Recommendation. (http://www.w3.org/TR/xlink/), June 2001.

[8] T. Elrad, R. E. Filman, and A. Bader. Aspect-Oriented Programming. *Communications of the ACM*, 44(10):29–31, October 2001.

[9] F. Garzotto, D. Schwabe, and P. Paolini. HDM - A Model Based Approach to Hypermedia Application Design. *ACM Transactions on Information Systems*, January 1993.

[10] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8), August 1995.

[11] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An Overview of AspectJ. In *In Proceedings of the European Conference on Object-Oriented Programming*. Springer-Verlag, 2001.

[12] K. Lieberherr, D. Orleans, and J. Ovlinger. Aspect-Oriented Programming with Adaptative Methods. *Communications of the ACM*, 44(10):39–41, October 2001.

[13] D. Schwabe and G. Rossi. An Object Oriented Approach to Web-Based Applications Design. *TAPOS - Theory and Practice of Object Systems*, 4, 1998.

[14] D. Schwabe, G. Rossi, and S. Barbosa. Systematic Hypermedia Design with OOHDM. In *ACM International Conference on Hypertext' 96*, 1996.

[15] P. Tarr, H. Ossher, W. Harrison, and S. Sutton. N Degrees of Separation: Multi-dimensional Separation of Concerns. In *Proceedings of the 21st International Conference on Software Engineering*, May 1999.