Analysing the navigational aspect

A. M. Reina
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Sevilla.
e-mail: reinaqu@lsi.us.es

J. Torres
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Sevilla
e-mail: jtorres@lsi.us.es

Abstract

The Internet and web applications have increased their popularity in the last few years. This boom has provoked the use of new approaches for web-based applications. These new methodologies try to address some new concerns which have appeared in this field and weren't in traditional methodologies. One of these concerns is navigation. At the same time, it has been proved that there are some concerns that aren't well treated with the traditional abstraction mechanisms ((functions, objects). They scatter by all the code of the program. This paper tries to join both approaches, crossing the gap between the design level proposed in the methodologies and the implementation level, using for it the proposed ideas in the area of the advanced separation of concerns.

Key words

Hypermedia, navigation, advanced separation of concerns, aspect-oriented programming.

1. Introduction

Separation of concerns is one of the basic disciplines in software engineering. With the proliferation of Internet and multimedia, new design methodologies that apply this principle have arisen. They address new concerns that weren't treated in traditional applications.

One of the terms that appear in this new environment is navigation. Navigation includes the intrinsic notion of movement through an information space [3]. The current position or context is a crucial factor, because many times the next navigation will depend on the previous navigation.

If we analyse the tendencies of the last web design methodologies (OOHDM [14], RMM [9], HDM [7], etc), we will deduce that navigation is an important concept and it should be treated separately from the basic functionality of the application.

On the other hand, the term aspect has arisen from the field of advanced separation of concerns, as an abstraction to separate concerns that are scattered all over the program code. This paper tries to analyse how these two approaches could be combined, proposing the treatment of navigation like an aspect, in the sense of aspect-oriented programming. We introduce an example to analyse the problems we have when designing navigation, and how the separated definition of navigation could help to solve them.

In section 2 we define the 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 field of separation of concerns. Afterwards we introduce the approaches of the last design methodologies to express navigation. Then we introduce and discuss an example to show the problems of navigation, and how the separation of navigation could help to solve it. Finally, we summarise and conclude the paper.

2. Navigation

Navigation is a concept widely used in web environments. In its primitive meaning it was bound to the hypertext idea, defining the action of jumping from page to page through a hyperlink. The advance of the 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 enlargement of the semantics of the navigation concept invite us not to treat as navigation the fact to push the link More when we browse the results of a query processed by a search machine as *google* or *altavista*, since we wouldn't move from an information space to another one. This problem would be considered as an interface one (we have an information space and we can't display all the information to the user simultaneously). Although we have a link in the hypertext, this link should be considered just as a way to do scrolling.

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

In [15] navigation is defined as the sensation the user has when he navigates through an object space from the application domain. These objects aren't the conceptual objects, but they are customised according to the user's profile using the view mechanism.

3. Separation of concerns

One of the approaches that has emerged strongly in the last few years is separation of concerns, a concept which has been applied in the field of software engineering practically from its beginnings. However, the current technology has been focused mainly on the programming level. Aspect-oriented programming (AOP) [5] idea 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).

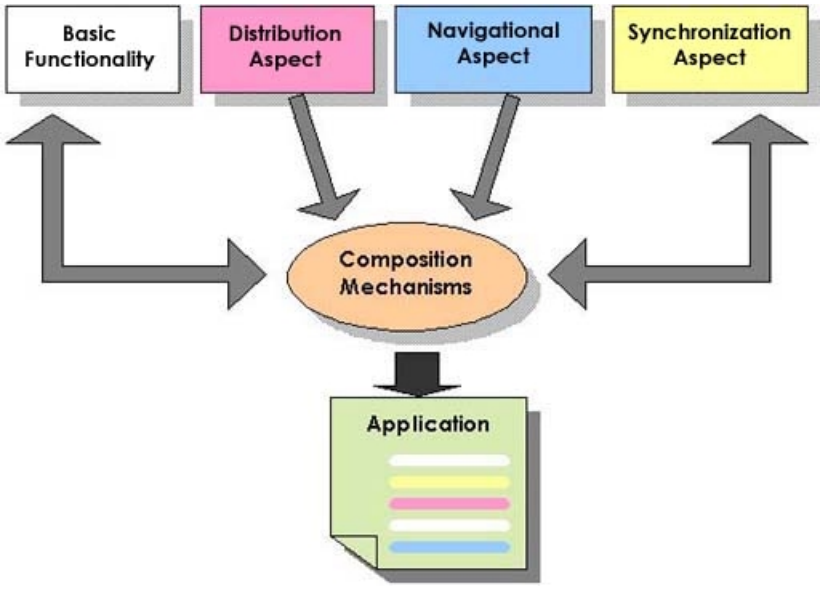


Figure 1. Aspect-oriented programming mechanisms

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 [13], which can be implemented using the DJ library (http://www.ccs.neu.edu/research/demeter/DJ), multidimensional separation of concerns [16], whose result has been Hyper/J (http://www.research.ibm.com/hyperspace/HyperJ/HyperJ.htm), composition filters [1], and aspect-oriented programming [10], 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 scattered all over the program code. These concepts are known as aspects, and to mix them with the code implementing the basic functionality are used pointcuts. The 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 behaviour 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, the adaptive methods express the behaviour like a high level description of how to reach to the participants of the calculation (called traversal strategy) and what to do when each participant has got 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.

The 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. A tool called Hyper/J has been made, which doesn't extend Java, as Aspect does. Hyper/J works with .class files, not with source files.

4. Navigation in methodologies

One of the pioneering methodologies in introducing navigation as a different aspect to treat during the design stage of the development of a web application has been HDM [7]. This methodology presented some primitives. Some of them have served as the base to many 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 [15]. 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).

Apart from these primitives, which already appeared in the first methodologies that treated navigation, OOHDM defines the navigational context, a new primitive to structure the navigational space. The navigational context is a set of nodes, links, context classes and other navigational contexts that are used to organise the 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 cross the gap between design and the implementation and to take this separation to the latest stages in the development of a software project. In order to obtain this separation at the programming level, it has arisen the aspect-oriented programming and the multidimensional separation of concerns. It is interesting to do a study to cross the gap between design and implementation using this technology.

We introduce the next example to study what advantages we will obtain if we succeed separating clearly the navigation aspect. Let us suppose we have made a web application for a museum, and when we designed the navigation for the paintings by author context, we had decided to use the access structure index. (Figure 2(a)).

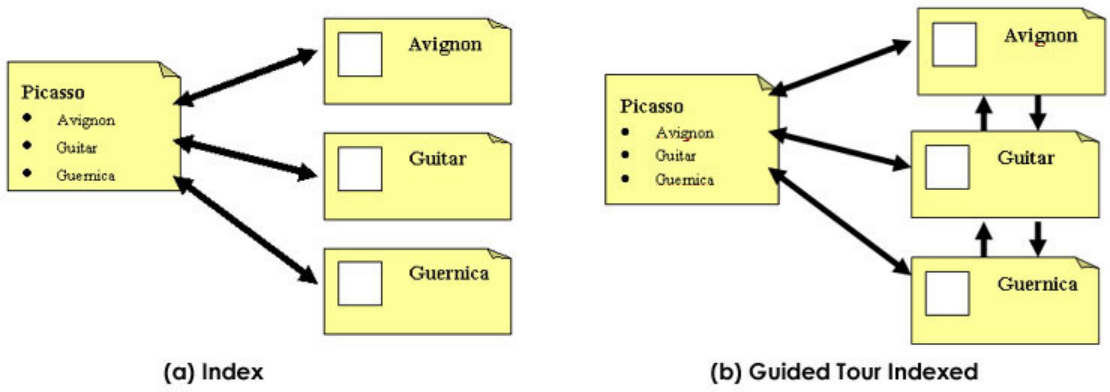
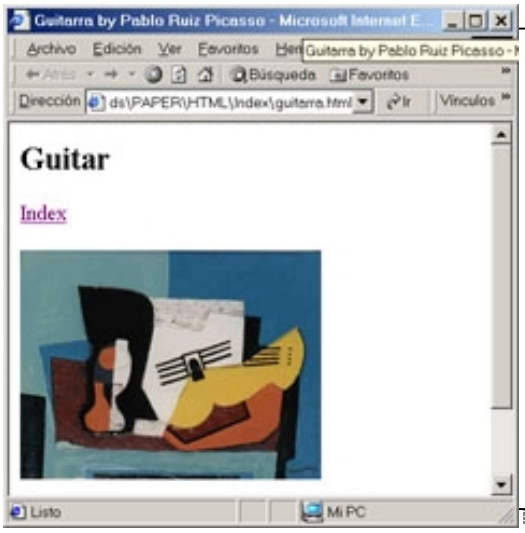


Figure 2. Access structures

Later, when the application was finished, the customer decided that the access structure we had chosen wasn't the best one, and we had to change it for an Indexed Guided Tour (Figure 2 (b)).




```
<HTML>
 <!- Access Structure Index->
 <HEAD>
  <TITLE>Guitarra
         by Pablo Ruiz Picasso</TITLE>
 </HEAD>
 <BODY>
  <H2>Guitar</H2>
  <p><a href="Index.html">Index</a></p>
  <p><img src="guitarra.jpg"></p>
  <hr>
  <cite><br>&copy;2001
         A.M. Reina Quintero</br></cite>
 </BODY>
</HTML>
```

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


```
<HTML>
 <!- Access Structure Guided Tour
Indexed->
 <HEAD>
  <TITLE>Guitarra
    by Pablo Ruiz Picasso</TITLE>
 </HEAD>
 <BODY>
  <H2>Guitar</H2>
  <p><a href="Index.html">Index</a>
     <a href="avignon.html">Previous</a>
     <a href="guernica.html">Next</a>
  </p>
  <img src="guitarra.jpg">
  <hr>
  <cite><br>&copy;2001 A. M. Reina
Quintero</br></cite>
 </BODY>
</HTML>
```

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

Changing something so conceptually simple as the type of access structure, can be an arduous and tedious work when you 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 emphasised the HTML code we have to introduce 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 the other nodes of the context (Guernica and Avignon, in this case).

Figure 5 describes the navigational classes using the notation proposed by Hennicker and Koch in [8], while Figure 6 poses the navigation structure model resulting of the two different access structures chosen. The index access structure is depicted in Figure 6 (a) and the guided tour indexed access structure is showed in Figure 6(b).

Figure 5. Navigation space model.

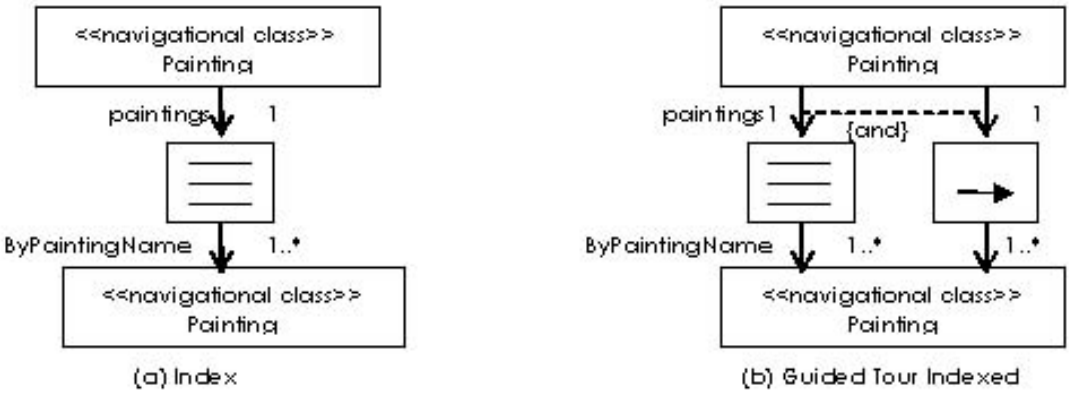


Figure 6. Navigation structure model.

In this paper we propose something like we describe in Figure 7, to separate completely the navigational elements. 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.

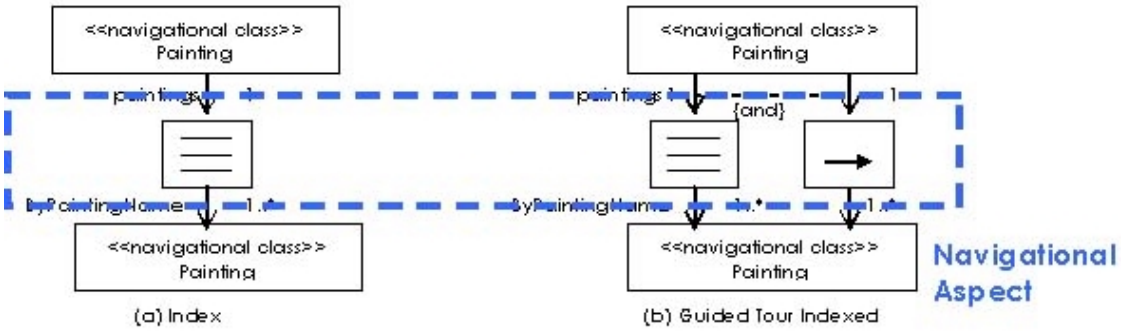


Figure 7. Separation of navigational aspect.

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

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 implement it? 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, which mean, where are we going to join the navigation aspect with the classes of the conceptual model?

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

With the appearance of the Extensible Mark-up Language (XML) [2] we have obtained the separation between presentation and content. Nowadays there are some new technologies coming from the XML world (as XLink [4]) that can help us to obtain this separation of concerns.

6. Conclusions and further work

If we pay attention to the current approaches in web design methodologies, we can realise that navigation is a key concept in this type of applications. The most recent methodologies treat it separately, dedicating a whole stage to the navigation design and creating different models, one for conceptual classes and another for navigation classes.

This separation should be taken from design to implementation. We could use separation of concern technologies to accomplish it. We propose to make a study to check if the aspect-oriented approaches fit well with the separation of this concern. If they do, which of these approaches fits better? and is there any important lack?. We consider it is interesting to see how useful are the new mark-up languages (XML, XLink) to separate this concern.

7. References

1. Bergmans, L., Aksits, M. *Composing Crosscutting Concerns Using Composition Filters.* Communications of the ACM. Vol. 44, n. 10, October, 2001.
2. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation. [Online: http://www.w3.org/TR/REC-xml]. October, 2000.
3. Cunliffe, D., Taylor, C., Tudhope, D. *Query-based Navigation in Semantically Indexed Hypermedia.* Proceedings of the Hypertext'97 Conference. 1997
4. DeRose, S., Maler, E., Orchad, D. *XML Linking Languaje (XLink) 1.0.* W3C Recommendation. [Online: http://www.w3.org/TR/xlink/] . June, 2001.
5. Elrad, T., Filman, R. E., Bader, A.. *Aspect-Oriented Programming.* Communications of the ACM. Vol. 44, n. 10, October, 2001.
6. Gamma, E., Helm, R., Johnson, Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley. 1995.
7. Garzotto F., Schwabe D., Paolini P. *HDM – A Model Based Approach to Hypermedia Application Design.* ACM Transactions on Information Systems, Jan, 1993.
8. Hennicker, R., Koch, N. *Systematic Design of Web Applications with UML.* In Unified Modeling Language: Systems Analysis, Design and Development Issues. (2001). K. Siau and T.Halpin (Eds.) Idea Group Publishing, 1-20.
9. Isakowitz, T., Stohr, E. A,, Balasubramanian, P.. *RMM: A Methodology for Structured Hypermedia Design.* Communications of the ACM. Vol. 38, n. 8, August, 1995.
10. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G. *An Overview of AspectJ.* In Proceedings of the European Conference on Object-Oriented Programming, Springer Verlag, 2001.
11. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J.M. and Irwin, J. Finland. *Aspect-Oriented Programming, in proceedings of the European Conference on Object-Oriented Programming.* Springer-Verlag LNCS 1241, June 1997.
12. Kishi, T., Noda, N. *Analyzing Concerns used in Analysis/Design Techniques.* Proceedings of the Conference on Object Oriented Programming: System, Languages and Applications (OOSPLA), 2001.
13. Lieberherr, J., Orleans, D., Ovlinger, J. *Aspect-Oriented Programming with Adaptive Methods.* Communications of the ACM. Vol. 44, n. 10, October, 2001.
14. Schwabe, D., Rossi, G., Barbosa, S. *Systematic Hypermedia Design with OOHDM.* ACM International Conference on Hypertext' 96, Washington, USA, 1996.
15. Schwabe, D., Rossi, G. *An Object Oriented Approach to Web-Based Applications Design*. TAPOS – Theory and Practice of Object Systems, vol. 4, 1998.
16. Tarr, P., Ossher, H., Harrison, W., Sutton, S.M. *N degrees of separation: Multi-dimensional separation of concerns.* Proceedings of the 21st International Conference on Software Engineering May 1999.