Concern-Sensitive Navigation: Improving Navigation in Web Software through Separation of Concerns

Jocelyne Nanard[1], Gustavo Rossi[2], Marc Nanard[1], Silvia Gordillo[2], and Leandro Perez[2]

[1] LIRMM, CNRS/Univ. Montpellier, 161 rue Ada, F34392 Montpellier cedex 5, France
{jnanard,mnanard}@lirmm.fr
[2] Facultad de Informática, Universidad Nacional de La Plata and Conicet Argentina
gustavo@sol.info.unlp.edu.ar

Abstract. Traditionally, the use of good techniques to improve software modularity, such as advanced separation of concerns, has no impact in the user experience, for example while navigating Web software. While the intent of these techniques is to simplify evolution and maintenance, navigation design quality is often seen as an unrelated concern. In this paper we present a novel approach for improving navigation in Web applications by using some of the core application's concerns (called navigational concerns) to derive their navigational structure. Using some realistic examples we show that, by carefully using these concerns, we can improve the user experience. Some implementation issues are discussed and a thorough comparison with related ideas in the Web Engineering field is presented.

Keywords: Separation of concerns, Concern-sensitive navigation, User experience.

1 Introduction and Motivation

Web applications have evolved from being simple information repositories to complex and ubiquitous platforms for performing complex business processes or for publishing and sharing multimedia information. Huge e-commerce sites such as Amazon.com, blogs like Youtube or Flickr or cooperative encyclopedia like Wikipedia are clear examples of this evolution. As these applications are being constantly modified, maintenance implies an additional challenge to software development methodologies. Fortunately the Web engineering community has already discussed and proposed advanced software techniques to simplify design and evolution (See for example [19], [14]); most of them are based on variants of the separation of concerns principle [12].

However, these design techniques are usually considered orthogonal to the problem of application usability. In this way for example, the quality of navigation structures is considered a completely disparate problem with respect to, for example, achieving design modularity. In other word, a Web software which has been conceived with high standards regarding evolution and maintenance does not necessary provide a good navigation experience to the final user.

Z. Bellahsène and M. Léonard (Eds.): CAiSE 2008, LNCS 5074, pp. 420–434, 2008.

In this paper we show how a wise separation of *application concerns* during modeling and design, and the information recorded during those stages can be cleverly used also towards providing a more *flexible* navigational structure and thus improving the user navigation experience. Our work aims at improving the cognitive and rhetoric access to information, which means providing the user with the needed information in each concern, and such that it is organized and presented in a more opportunistic way [13]. Suppose for example an application such as Amazon.com in which users navigate through thousands of products with different concerns (tasks or interests) in mind. In Fig. 1 we show a typical screen of a book with the corresponding information and available functionality.

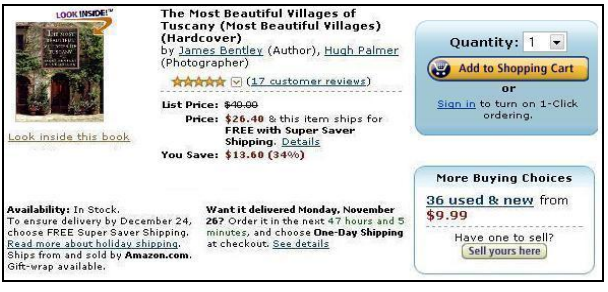

Fig. 1. A Book in Amazon.com

The page for a book (like in Fig. 1) looks exactly the same independently of the reason why the user reached it. For example it could have been accessed as a book on Italy, it could result from an Amazon recommendation according to the user's buying history, or it could be also accessed from the shopping cart because the user wants to be sure about the book's contents before proceeding to pay for it. This "flat" structure, in which every object looks equally regarding the context in which it is accessed, diminishes usability and might also cause errors [3].

The contents of the book page should be improved by taking into account the "dominant" concern in which it is being accessed. For example in Fig. 2.a we show part of a possible Web page for the book when accessed as a recommended item and in Fig. 2.b, the same book when accessed from the shopping cart. In Fig. 2.a there is a link to get an explanation of why the book was recommended, and links to the previous and next recommendations. Notice that these links do not make sense when the book is accessed with other different concern in mind. In Fig. 2.b meanwhile, there is an indication that the book is already in the cart and that if added again, it will imply adding a new unit. In both cases knowing the actual user's concern helps to enrich the information on the target page with new contents and links to simplify or clarify the user's task.

Some Web Engineering approaches have solved sub-sets of this problem using specific ad-hoc techniques and/or notations. For example, OOHDM [23] uses the concept of navigational contexts to enrich hypermedia nodes when accessed in a particular set (for example the set of recommended products). In [24] a similar idea is used to restrict operations in the context of a business process, i.e. to avoid that the same product is added once more in the shopping cart while checking-out. Our approach aims at providing a more systematic context-sensitive navigation.

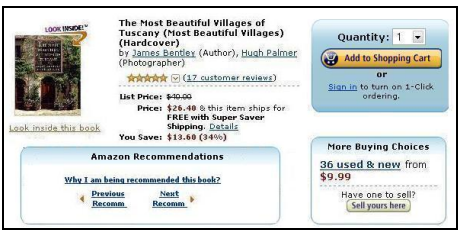

Fig. 2. (a) Book accessed in the Recommendation Concern

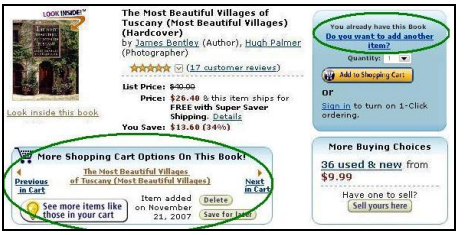

Fig. 2. (b) Book accessed in the Shopping Cart concern

This paper has three main contributions:

- We introduce the concept of concern-sensitive navigation (CSN) as a conceptual and practical tool towards improving the navigational structure of Web applications as perceived by the user.
- We show how to introduce CSN in Web development methodologies, emphasizing how the use of techniques for advanced separation of application concerns can be used not only to improve modularity, but also the user's navigation experience.
- We show the feasibility of CSN by briefly analyzing implementation alternatives.

The rest of the paper is organized as follows. We first introduce concern-driven navigation and illustrate the concept with simple examples. We next discuss how to engineer applications which support concern-driven navigation discussing modeling, design and implementation issues. Finally we compare our work with related approaches and present some further research we are pursuing. We show the design and implementation feasibility of these ideas by providing illustrative examples.

2 Concern-Sensitive Navigation

The main motivation of our research is to show that by separating concerns we can not only improve evolution and maintenance but also produce better navigation structures. In this section we explain deeply which are the issues one has to consider to realize CSN.

2.1 Background

According to [17] an *application concern* is defined by any coherent set of requirements, e.g. all requirements referring to a particular theme or behavioral application feature. More generally [25] defines a concern as a "matter of consideration in a software system". Concerns may reflect functional or non-functional aspects of an application such as recommendations and checkout in E-commerce, topic areas such as history or geography in an Encyclopedia. Concerns may be generic, when they appear in a broad number of applications (e.g. adaptivity, usability), domain specific when they only apply to a set of applications (payment in e-commerce), or even application specific when they only show up in a particular kind of software (e.g. Marketplace in Amazon.com).

A *navigational concern* is an application concern that affects navigation, i.e. it manifests itself in the navigational structure of the application (the exhibited contents and links), and which therefore impacts in the way users navigate the application. In this paper we focus on navigational concerns and ignore others which are nevertheless important but do not affect navigation (such as persistence or security).

Most Web applications deal with a myriad of navigational concerns and usually (for improving usability though many times only for marketing reasons) they exhibit information pertaining to more than one concern in the same page, i.e. in the same page we might find contents, links and functions which belong to different concerns. Modern software engineering techniques such as aspect-oriented development [9] promote a clear separation of concerns during specification, design and programming and their late weaving either during compilation or even execution. In this way one can diminish the impact of crosscutting concerns during software evolution. However, and as we showed before, the page in Fig. 1 looks the same independently of the concern the user has in mind when accessing it (searching a product, being recommended a product, as part of an offer, etc).

Users navigate in Web applications to perform a specific task; mature Web design methods have already prescriptive approaches and notations to map task descriptions (e.g. specified as use cases) into conceptual and navigational models [26]. Our approach complements these ideas with a strategy to enrich the navigation objects with information specific to the concern that the user is traversing. The mechanics of CSN, as well as its scope should be defined by the designer according to the user's need and convenience. In what follows, we show how to use the information on navigational concerns to improve the navigation structure of the application. For the sake of conciseness we only focus on navigation aspects.

2.2 Definition

To formalize our notion of CSN we refer to a navigation object type Nj (the realization of an atomic or composite hypermedia node type) as comprising a set of properties; these properties may be further classified in media contents, anchors for links or operations exhibited by the node and can be divided in two groups:

- properties intrinsic to the object [15] (i.e. which are present regardless the concern in which an object is accessed). We call them core properties;
- properties which, given a concern Ci, correspond to the set of perceivable properties of Nj when accessed in the Concern Ci and which is the result of applying a function P (Ci, Nj).

For each meaningful pair (Ci**,** **N**j) the set of properties should be a superset of the core properties of Nj. In Fig. 3 we illustrate the definition for the examples in Fig. 1 and 2 using a UML-like notation. Notice that the same node instance exhibits different properties according to the concern in which it is accessed. By adjusting the node's properties to the concern in which it is being accessed we improve the navigational structure, by making contents more focused to the actual concern the user is navigating (i.e. the intended task).

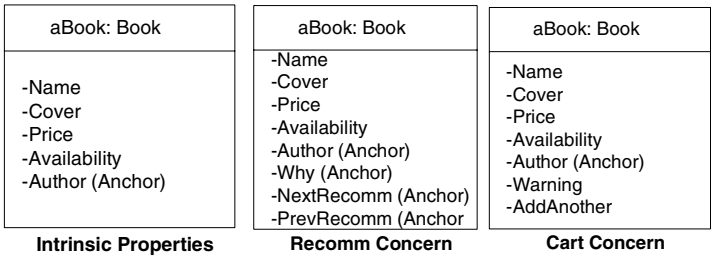

Fig. 3. Intrinsic Properties vs. Properties in Recommendation and Cart Concerns

Regarding this initial definition there are some additional issues to consider:

- Notice that the perceivable properties do not depend on the user profile or identity, which means that CSN is slightly different from adaptive navigation (See the related work section).
- Besides the so called intrinsic properties, there might be properties which pertain to different concerns and which we want to exhibit permanently (e.g. the Add To Wish List operation), i.e. regardless the navigation path. Though this is a design issue not fully related with concern-driven navigation but with concern composition, we only give an overview of it (See Section 3.2).
- Defining the concerns which affect a node type requires a clear understanding of the application concerns, their relationships and the way they reflect in navigation (See next section).

2.3 Which Categories of Concerns Affect Navigation?

As explained in [25] there are many different kinds of concerns which may arise in the process of Web software development. For the sake of conciseness we enumerate here the most important types of concerns which are exposed to the user and therefore affect navigation:

- *Task concerns* are the broadest category of navigation concerns; they abstract those concerns which relate to the different high level actions that the user can perform in a Web application, for example exploring products, managing the shopping cart, adding reviews, checking out, managing lists, etc. Some of these tasks are related with finer grained application features such as *services* offered by the application; in the case of Amazon, recommendations, marketplace, lists, etc. Notice that while involved in a service the user is always performing a task.
- *Topics:* Pure informational sites might introduce even finer-grained concerns; for example topics or themes such as in an Encyclopedia. Topic-based concerns are also present in the context of tasks; for example while searching books in Amazon.com, the genre of the book (thriller, travel, technical) or its theme area (Software Engineering, Programming, etc) might itself become a concern.
- *"Pure" navigational concerns*, like Guided Tours or sets. These are usual abstractions in navigational design and therefore can be considered also as specific concerns.

Deciding which is the appropriate level of granularity for choosing concerns during the modeling stage is, as well as choosing the "right" concerns, part of the designer job, and it is outside the scope of this paper. However, the reader can find good guidelines in the literature on Early Aspects [8], particularly in [2].

2.4 Which Kinds of Concern "Enrichment" Improve the User's Experience?

Though the answer to this question strongly depends on the specific concern, there are two broad categories of enrichments:

Basic enrichments. We found three kinds of enrichments, namely:

- *New or modified contents*: As shown in the cart concern of Fig. 3, we can enrich the node instance with new attributes
- *Anchors and Links*: Also in Fig. 3, in the Recommendation concern we added a new link and the corresponding anchor to improve navigation
- *Operations*: A node instance might exhibit additional operations when accessed in a concern; for example we could have added a (deleteFromCart) operation in the Cart concern in Fig. 3.

Enrichment Patterns. For each of the previously mentioned concern types, the following patterns are the most recurrent:

- *For Task-Based Concern***:** When the concern is defined by a business process (like in [24]), and operating on the target node might conflict with the process, it is advisable either to eliminate operations which collide with the concern or to add specific warnings (e.g. the shopping cart or checkout concerns in Amazon).
- *For Thematic concerns*: when a node is accessed in that concern, add information and links specific to the topic which is related with the node. For example the book in Fig. 1 could be enriched with links to other books on Italy (or related to the higher level concern, Travel)
- *For Pure Navigational Concerns***:** When the concern can be represented as a set as in OOHDM navigational contexts [23] (e.g. the set of recommendations, etc.), it is wise to enrich the node with links to the index of the current set, and to the previous and next elements of the set. Another example of this kind of enrichment can be found in tag-based navigation like in Flickr (e.g. by providing links to other photos with the same tag).

3 Engineering Web Applications Supporting CSN

Web Engineering approaches, like those in [22], support separation of the most outstanding concerns in this kind of software: requirements' capture, content or application modeling, navigation and presentation design, business process modeling, etc. (they correspond to methodology-related concerns). Some of them have also introduced elements of advanced separation of concerns (such as aspect-orientation) to deal with cross-cutting concerns [4]. Even though the kind of application concerns which might be reflected in CSN structures does not necessarily correspond to "aspects"

(as they may not crosscut in the standard way), we claim that the most relevant identified concerns (e.g. following the classification in 2.3) should be designed separately. We next explain how to map concerns into navigational structures. Though we use OOHDM as the exemplary method, the ideas can be applied to other well-known approaches like UWE [14], OOWS [19] or WebML [5]. In the following, we discuss mainly the Requirement, Modeling and Navigational design issues; some Implementation aspects are then outlined. Presentation issues can be read at [10].

3.1 Requirement and Modeling Issues

In [11] we presented an approach to model navigational concerns in Web applications; the approach which derives from well-known ideas in the Early Aspect community [8] helps to elicit, identify and specify the interactions which emerge in each navigational concern. In our work, each concern is explicitly represented using a XML-template and for each use case in the concern a User Interaction Diagram (UID) is built. UIDs show how the interaction proceeds in a high level way. In Fig. 5, we show part of the definition of the Recommendation concern (on the left) and the corresponding UID (on the right). The UID shows in a simple state diagrams which items are presented to the user, either as simple structures such as Book and its attributes or as sets of structures (those which begin with "…") and the transitions corresponding to user's actions such as selecting the "Why" option in the right part, or the "Next" and "Previous" recommendation below state "C". When comparing the UID in Fig. 4 with one in the core application concern (not shown for conciseness), we will find that the information exhibited by books is slightly different (See state "C"); this information will be used to define CSN as described in Section 3.2.

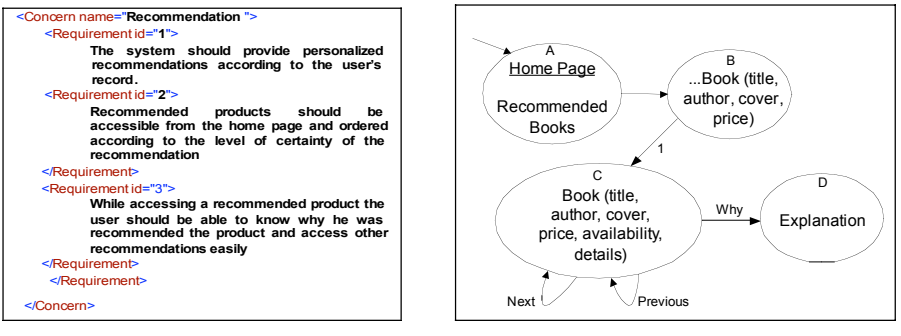

Fig. 4. Requirements corresponding to the Recommendation concern

Once the whole set of requirements have been elicited and modeled, a conceptual OOHDM model is built using the information collected from the UIDs, which as explained in [23], allow to define the attributes and methods of conceptual classes. Following the Theme approach [7] we propose to partition the conceptual model in submodels, one for each of the relevant concerns (See [11]). However, other approaches (aspects, object decorators, etc) can be also used according to the kind of concern crosscutting; this discussion is outside the scope of this paper (See for example [16]).

For the sake of conciseness, we will concentrate on navigational concerns and use as an example a fragment of the e-store shown in previous Figures. In Fig. 5 we present a simplified OOHDM conceptual model corresponding to this example. We only present the conceptual sub-models corresponding to the Core and Recommendation concerns emphasizing class structure over relationships, as they are sufficient for illustrating the rationale and mechanics for building the concern-sensitive navigational model. Following [7], each model presents the view of the application classes according to the concern. Notice that there are two classes in the Recommendation concern which do not appear in the Core concern. When these models are weaved some classes may be transformed in a class containing the union of the definitions of each model (e.g. Product), others might evolve into aspects, etc.

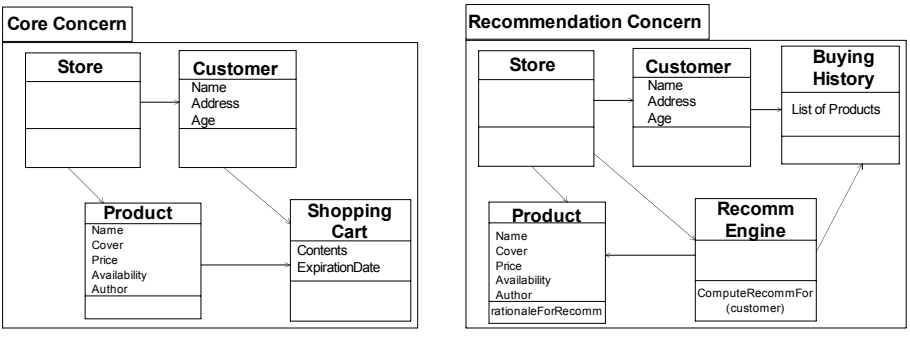

Fig. 5. Conceptual Model for the e-store site

3.2 Navigational Design

Web engineering approaches provide primitives for describing the navigational structure of the application, i.e. for defining nodes, links, indexes and higher-level structures such as landmarks, guided-tours, etc. Nodes contents are usually defined opportunistically to improve usability; for this reason, many times the same node exhibits information pertaining to different concerns (See Fig. 1). In OOHDM we do this by defining the node's attributes as views on the corresponding conceptual classes (eventually "viewing" their definitions in the different involved concerns) [23].

We are interested however, in the information and links which are only meaningful in some specific concerns. As defined in 2.3, CSN allows enriching the contents and links of a hypermedia node according to the current user's concern. A navigational model expressing concern-sensitive navigation should take into account what follows:

Identify which node types are "affected" by existing concerns. The first output of a navigational diagram (in OOHDM and other methods) is a navigational schema formed out of node and link types, representing the type of objects the user will perceive with their attributes and the navigable relationships. As said above, these types are obtained from the conceptual model using a view mechanism [23] which allows gathering information according to users' needs. A simplified "flat" navigational schema for the e-store site is presented in Fig. 6. Again, we emphasize node's structures and do not indicate link types' names.

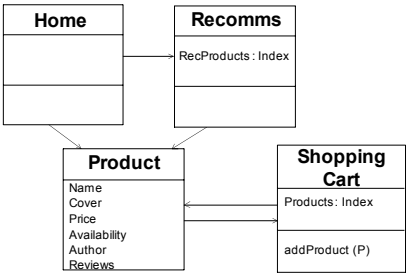

Fig. 6. Navigational Schema for the e-store site

The navigational schema shows the specification of "natural" node types (i.e. those which stand independently of any concern, See [15]), and the links which provide navigation paths between node instances. For the sake of simplicity we don't present the OOHDM navigational contexts model in which indexes are further specified (See [23]).

By analyzing the requirement model, particularly the specification of concerns and their realization with UIDs, we can identify which of the node types are affected by each concern. We do this by building a table (See Table 1) which makes explicit the function P, described in Section 2.2. Each line corresponds to a concern. For each concern, we add a new column for each node type affected in this concern.

Define the information, links and services to be added when accessing a node in each particular concern. For each pair (concern, node type) we indicate the corresponding enrichment when a node instance is accessed in that concern. This decision takes into account the nature of the concern (i.e. the current user's task); for example we might decide to add more specific information to improve user's understanding, links to related information objects (corresponding to the same concern) to improve the completion of the task, etc. As mentioned in 3.1, UIDs are the first source for this information as they collect most of the data and possible interactions corresponding to each concern. Table 1 shows a sketch of the enrichment corresponding to the Node type Product when accessed in the Recommendation concern (omitting the anchor's specification for conciseness).

We represent concern-sensitive navigational diagrams with the notation of role-enrichment. A role type (indicated as a rounded rectangle) shows, when attached to a node, the additional information and links that will be shown in the corresponding

Table 1. Table showing the enrichment for each concern and node type

Nodes / Concerns	Node Type 1	…	**Product**	…
Concern 1				
…				
Recomm			**Why (Anchor)** **Next (Anchor)** **Prev (Anchor)**	
…				

concern. Roles of a node type act as decorations adding the concern-specific information. In Fig. 7 we show how we enriched the navigational diagram of Fig. 6 with concern information, represented with roles. There are two roles, one for products accessed from the recommendation list (i.e. in the recommendation concern) and one for products accessed from the cart (i.e. in the Cart concern); in both cases the role contains the additional features as part of its specification. Notice for example the two links defined from the *RecommProduct* role into itself and the additional Explanation node type, which reflect the specification in the UID of Fig. 5 and the corresponding table entry of Table 1. Also the role *ProductInCart* adds a behavior *AddToCart* which possesses a slightly different semantics with respect to the "normal" addToCart behavior in Product, asking the user if he really wants to increase the number of units of the product. The use of roles in Web Engineering has been discussed in our previous work in [21].

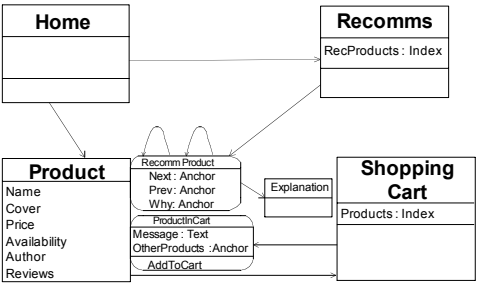

Fig. 7. A concern-sensitive navigational schema

3.3 Further Issues

An interesting modeling (and also implementation) issue arises when dealing with families of navigational concerns. There are some concerns which are "atomic" because there is only one instance of the concern, and as a consequence all nodes affected by the concern either have the same enrichment or the enrichment only depends on the affected node. The best example of this kind of concerns is the Cart concern: there is only one Shopping Cart and therefore all products navigated from the Shopping Cart (i.e. in the Cart Concern) will have the same enrichment. Meanwhile, in the recommendation concern, the enrichment depends on the node instance (the product) as links to other recommendations are a function of the product. In this case, in our modeling approach, the corresponding role type can be considered a singleton (it has only one instance) which adds information somewhat parameterized by the target node instance. This added information (See Fig. 2 and 7) is obtained by collaboration with the node instance.

Meanwhile, certain concerns, particularly Topic or Thematic concerns have usually many instances, one for each possible topic. In our example of Fig. 1 and 2, we could have been exploring books on Italy, and therefore when accessing the book, the actual

concern is Italy (which is a sub-category of the concern Travel). Even though the designer should decide which the suitable level of granularity is, and eventually choose if Italy is a possible concern (e.g. Travel might be preferred), it is obvious that we expect different additional information while exploring books on Web Engineering. In any cases, the most elegant design solution is to consider that the concern role type (Topic) has many instances, one for each topic; these instances are created dynamically, when accessing the target node instance. If necessary, there might be a hierarchy of role types to cope with variants among Travel, Technical Books, Software, etc. with respect to the specific enrichment for each concern. Fig. 8 shows the e-store screen with this enrichment and the corresponding role and type specification to cope with this situation; the parameter in the role specification is used at instantiation time, i.e. when the corresponding role instance is created.

Fig. 8. Role Type parameterized with the concern instance

3.4 Implementation

We show the feasibility of mapping the previously described navigational structures onto a running application, by briefly describing two implementation alternatives.

The first alternative consists (as most Web design methods do) in translating the specification of nodes into XML files which, when being populated, are themselves translated into final interfaces by using XSL specifications. Similarly, we map the role types into XML files which are aware of the specifications they enrich. In the simplest case, producing the intended node instance requires the injection of the role file into the corresponding node instance. In [10], we described how to use XML transformations (described using XSLT) to weave two concerns together. Transformations are easy to specify and only use standard technologies. There might be cases however, in which the enrichment requires more subtle processing, for example to compute links which depend on the target node instances (as in the recommendation concern), or on the combination between the concern and the node instance (as in the case of topic-based concerns). This processing can be also specified using rules in the transformation files. Though the details are outside the scope of the paper, we show in Fig. 9 the simplified XML files for the core and recommendation concern and in Fig. 10 the XSLT transformation to weave them together; finally in Fig. 11 we present a UML activity diagram illustrating the process from the user's request to the generation of the necessary structure to realize concern-sensitive navigation.

```
<book>
<name>The Most Beautiful Villages of Tuscany </name>
<cover>TheMostBeautifulVillagesofTuscany .jpg</cover>
<price>40.00</price>
<availability >24 hs</availability>
<author>/search .do?author=JamesBentley </author>
</book>
```

```
<recommendation >
<why>/recommendation .do?id=2345</why>
<nextRecomm >/product?id=254</nextRecomm >
<prevRecomm >/product ?id=168</prevRecomm >
</recommendation >
```

Fig. 9. XML files corresponding to core and recommendation concerns

```
<xsl:template match ="/book">
<book>
<xsl:copy-of select ="*"/>
<xsl:copy-of select ="document ('recomm .xml')/recommendation /*"/>
</book>
</xsl:template>
<xsl:template match ="@*|node()">
<xsl:copy>
<xsl:apply-templates select ="@*|node()"/>
</xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

```
<book>
<name>The Most Beautiful Villages of Tuscany </name>
<cover>TheMostBeautifulVillagesofTuscany .jpg</cover>
<price>40.00</price>
<availability >24 hs</availability>
<author>/search .do?author=JamesBentley </author>
<why>/recommendation .do?id=2345</why>
<nextRecomm >/product ?id=254</nextRecomm >
<prevRecomm >/product?id=168</prevRecomm >
</book>
```

Fig. 10. XSLT transformation and the result of its application

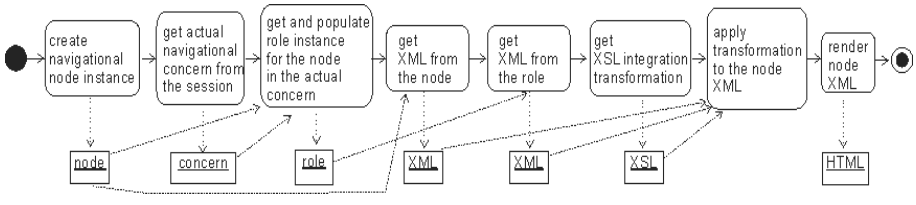

Fig. 11. Activity diagram for weaving a role into the core concern

A more systematic alternative is to use a model-driven approach [26], in which the semantics of role diagrams, such as the one in Fig. 7, feeds the transformation engine to produce the intended behavior. In [20] we presented a framework, CAZON, which injects irregular functionality into node instances. We have used CAZON to incorporate volatile functionality in node instances; such services are included for short periods of time (such as draws, promotions, holiday's offers, etc.). Insofar as CAZON uses XML files to describe nodes (both core and volatile), its underlying engine uses the same basic ideas as the previously described transformations, relieving the designer from the job of specifying them. We are currently extending CAZON to provide role enrichment, by expressing the relationships between role and node types using CAZON's built-in mechanisms, to show relationships between base and injected structures and behaviors.

As a summary of the previous explanation, we stress the importance of specifying in a declarative way the relationships between role types (as realization of concern enrichment), and node types. Even though this specification can be done using XML we think that model-driven approaches are preferable because they relieve us of implementation details.

4 Related Work

Separation of concerns has been a driving force in Software (and Web) Engineering for years. As mentioned in Section 1, the main rationale for improving separation of concerns has been simplifying software evolution and maintenance by achieving modularity. Treating concerns as "first class" artifacts in the software development process helps to better understand the underlying domain and produce composable modules which reflect the requirements in each concern and the relationships between them. Approaches like Aspect-Oriented Software Development [7], [9] guarantee modularity and seamless composition of core and aspectual functionality.

All Web Engineering approaches recognize the need to separate the design in layers which deal with clearly different concerns such as navigation, presentation and also business processes (We call this *horizontal separation of concern*) and even adaptation [4]. An interesting example of the use of advanced horizontal separation of concerns in Web Engineering is [6] which proposes a formal specification of the connections between different models during the development cycle to produce seamless weaving; the weaving model itself is described as a meta-model. The main difference between our work and existing approaches in Web engineering, is that we use also a *vertical separation of concerns* related to the essence of the application, to systematically produce better navigational structures and not just to ease evolution. While we rely on well known separation of concern techniques to specify and design each concern, we use the information collected from requirements to navigational design, to realize an improvement in the information and links perceived by the user while navigating in the context of a concern. Our work also generalizes some existing approaches to enhance navigation in specific contexts such as sets of related objects (called Navigational Contexts in OOHDM [23]) or business processes [24].

CSN has some points in common with the work on adaptive hypermedia [1]. Adaptive hypermedia approaches seek to improve user's navigation by taking into account the user's profile and needs. In an adaptive hypermedia application, nodes and links vary according to the characteristics of the user, his navigation history, etc. Adaptive hypermedia systems rely on a user model which represents the meaningful user's features and an adaptation model in which the adaptation rules and algorithms are specified. Our work meanwhile, while also producing an improvement with respect to "flat" navigation structures, does not pose additional requirements to Web software, such as recording the user's features or elaborated rules or algorithms. We rely on the use of well-known and mainstream software practices (such as separation of concerns) to generate a better navigational experience. Additionally, as the underlying designs are modular (e.g. by the use of aspects and/or roles), adding new concerns and their corresponding navigational adaptations is straightforward, as the core functionality is oblivious with respect to the new (and of course the "old") concerns.

5 Concluding Remarks and Further Work

We have presented an approach to use separation and composition of concerns, not only to enhance Web software modularity, but also to improve its cognitive and rhetoric access. While we recognize the importance of using advanced separation of

concerns techniques to ease application evolution, we claim that these techniques can be further applied in the context of navigational design to obtain better navigational models; for this aim we introduced the concept of concern-sensitive navigation. Applications supporting concern-sensitive navigation can offer the user more focused information, links and services according to his actual concern. While this idea shares some of the objectives of adaptive and context-aware hypermedia approaches, it introduces an orthogonal concept: the navigational concern. We have shown, with simple examples, that producing a concern-sensitive navigational model is rather simple, and we have provided a proof of concept of its implementation feasibility. For this, we have relied on standard tools and representation techniques to demonstrate that the idea can be easily put to work without complex engines or frameworks.

We are now studying several aspects, which belong to finer grained details of concern-sensitive navigation. One of them is dealing with the extent of a concern during navigation; while the "entrance" to a concern is clearly defined at design time as shown in the diagram of Fig. 7, many times it is not obvious when the user concern changes, for example when navigating links which are defined in the core concern. Related with this issue and with the enrichment patterns (See Section 3), we are also researching on patterns for concern selection to enrich existing catalogues of Web patterns such as [27]; in some kinds of concerns, particularly topic-based concerns, the user can select the actual concern for example when choosing a menu option (e.g. categories of books, subjects in an encyclopedia, etc). Knowing these patterns can help the designer to further improve the navigational structure. We are also researching on the concern-based improvement of applications in which most contents are provided by user, such as Wikipedia. Further usage analysis is necessary to have a clear understanding of the impact of concern driven navigation in users.

We are also finally researching on different ways of implementing concern-sensitive navigation; particularly we are using our ideas of XML transformations [10] and tree transformation through grammar networks [18] to ease the process of concern enrichment both at the navigation and interface levels. Finally we are working in the process of measuring the improvement provided by CSN; this can be done by analyzing how user's tasks are simplified but will also require further experiments with real users.

References

1. Adaptive Hypermedia Reference Library, http://wwwis.win.tue.nl/ah/publications.html
2. Baniasaad, E., Clarke, S.: Finding Aspects in Requirements with Theme/Doc. In: Proc. of Workshop Early Aspects 2004, associated to the ACM Conf. AOSD (2004)
3. Baresi, L., Denaro, G., Mainetti, L., Paolini, P.: Assertions to Better Specify the Amazon Bug. In: Proc. of the 14th Int. Conf. on Software Engineering and Knowledge Engineering. ACM Int. Conference Proceeding Series, vol. 27 (2002)
4. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling Adaptivity with Aspects. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 406–416. Springer, Heidelberg (2005)
5. Ceri, P., Fraternali, P., Bongio, A.: Web Modeling Language (WebML), A Modeling Language for Designing Web Sites. Computer Networks and ISDN Systems 33(1-6), 137–157 (2000)

6. Cicchetti, A., Di Ruscio, D., Pierantonio, A.: Weaving Concerns in Model Based Development of Data-Intensive Web Applications. In: Proceedings of the ACM Symposium on Applied Computing (SAC 2006), pp. 1256–1261. ACM Press, New York (2006)
7. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design. The Theme Approach. Object Technology Series. Addison-Wesley, Reading (2005)
8. Early Aspects Home: http://www.earlyaspects.net
9. Filman, R., Elrad, T., Clarke, S., Aksit, M.: Aspect Oriented Software Development. Addison-Wesley, Reading (2004)
10. Ginzburg, J., Rossi, G., Urbieta, M., Distante, D.: Transparent Interface Composition in Web Applications. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 152–166. Springer, Heidelberg (2007)
11. Gordillo, S., Rossi, G., Moreira, A., Araujo, J., Vairetti, C., Urbeita, M.: Modeling and Composing Navigational Concerns in Web Applications. Requirements and Design Issues. In: Proc. of Latino American Conf. on the WWW (LA-Web 2006). IEEE Computer Society Press, Los Alamitos (2006)
12. Harrison, W., Ossher, H., Tarr, P.: General Composition of Software Artifacts. In: Löwe, W., Südholt, M. (eds.) SC 2006. LNCS, vol. 4089, pp. 194–210. Springer, Heidelberg (2006)
13. Horchani, M., Nanard, J., Nanard, M.: Les Hypermédias comme Paradigme d'Interfaces Adaptatives. In: Saleh, I. (ed.) Les hypermédias. Hermès, pp. 119–146 (2004)
14. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering. In: (22)
15. Kristensen, B.B., Osterbye, K.: Roles, Conceptual Abstraction Theory and practical Language Issues. Theory and Practice of Object Systems 2(3), 143–160 (1996)
16. Marin, M., Moonen, L., van Deursen, A.: A classification of Crosscutting Concerns. In: Proc. IEEE Conf. on Software Maintenance (ICSM 2006). IEEE Computer Society Press, Los Alamitos (2006)
17. Moreira, A., Araujo, J., Rashid, A.: A Concern-Oriented Requirements Engineering Model. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 293–308. Springer, Heidelberg (2005)
18. Nanard, M., Nanard, J., King, P.R.: A structural computing approach to the production of multimedia document series. NRHM 12(2), 165–190 (2006)
19. Pastor, O., Abrahão, S., Fons, J.: An Object-Oriented Approach to Automate Web Applications Development. In: Bauknecht, K., Madria, S.K., Pernul, G. (eds.) EC-Web 2001. LNCS, vol. 2115, pp. 16–28. Springer, Heidelberg (2001)
20. Rossi, G., Nieto, A., Mengoni, L., Lofeudo, N., Nuño Silva, L., Distante, D.: Model-Based Design of Volatile Functionality in Web Applications. LA-WEB, pp. 179–188 (2006)
21. Rossi, G., Nanard, J., Nanard, M., Koch, N.: Engineering Web Applications with Roles. Journal of Web Engineering 6(1), 19–48 (2007)
22. Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.): Web Engineering: Modelling and Implementing Web Applications. Springer, Heidelberg (2008)
23. Rossi, G., Schwabe, D.: Modeling and Implementing Web Applications with OOHDM. In: (22)
24. Schmid, H., Rossi, G.: Modeling and Designing Processes in E-Commerce Applications. IEEE Internet Computing 8(1), 19–27 (2004)
25. Sutton, S., Rouvellou, I.: Modeling of Software Concerns in Cosmos. In: Proc. of ACM Conf. AOSD 2002. ACM Press, New York (2002)
26. Valderas, P., Fons, J., Pelechano, V.: Transforming Web Requirements into Navigational Models: AN MDA Based Approach. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 320–336. Springer, Heidelberg (2005)
27. Van Duyne, D.K., Landay, J.A., Hong, J.I.: The Design of Sites: Patterns for Creating Winning Websites. Prentice-Hall, Englewood Cliffs (2006)