Metamodelling and transforming sitemaps for reconciling information architecture and navigation design

A. M. Reina Quintero[1] and J. Torres Valderrama[1]

Department of Languages and Computer Systems
E.T.S. Ingeniería Informática.
Avda. Reina Mercedes, s/n.
41007 Seville, Spain
{reinaqu, jtorres}@lsi.us.es
http://www.lsi.us.es/~reinaqu

Abstract. [1] Some of the greatest challenges when developing Internet sites are related to two disciplines: Information Architecture (IA) and Navigation Design (ND). However, there is a disconnection between these two fields, especially due to the misunderstanding that information architecture and web design are mutually exclusive. A way to bring these two fields closer and integrate them into multidisciplinary projects is to focus on deliverables, because they are the artifacts that are shared among stakeholders, customers and all the members of the development team. This paper is focused on one of the most widely used information architecture deliverables: sitemaps. A metamodel for high-level sitemaps is specified, as a way of determining the building blocks used in this kind of deliverable. Furthermore, a set of model to text transformations have been defined in order to have a preliminary website sketch focused on structural and utility navigation, the main concerns that are addressed in sitemaps.

1 Introduction

Some of the greatest challenges when developing a good Internet site are not always of a clear technological character [7], and they are related to what should be included on the site, how the site should be organized, and how users will be able to find their way around. These questions have been addressed from two different disciplines: Information Architecture (IA) and Navigation Design (ND). While information architecture involves organizing, structuring and labeling information, navigation design involves technical development and visual design.

However, researchers have argued that there is a disconnection between functional architecture and information architecture, and also between business models and technical architectures [1]. This is due to considering that information

[1] This work has been partially supported by the *Spanish Ministery of Science and Technology* and FEDER funds: TIN-2007-67843-C06-03 and TIN2007-64119

architecture and web design are mutually exclusive. Thus, many times bad designs are a consequence of the matter that information technology is too separated from the design process or is too far integrated in the design [11].

Although information architecture can be considered as an extension to web design [9], many current information modeling approaches (such as WebML [4], OOHDM [10], and WAE2 [5]) have failed to address high-level aspects, such as architectural and business process modeling and to integrate information architecture into design. This is due to having typically focused on modeling at a relatively low-level and has caused that high-level concepts that are managed in IA, such as main navigation and local navigation, don't have a direct counterpart in these low-level models.

Thus, it is important not only to introduce IA in the web development process, but also to bridge the gap between information technology and design process. A good way to bring these two disciplines closer is to share a common vocabulary, and to do so, a first step is to share deliverables, because they represent the interests of the different stake-holders involved in a project. Having this in mind, it is important to explicitly describe deliverables, in order to see which their main building blocks are. If we know the deliverable components, we can do mappings and transformations. Thus, we propose to allow everyone to use whatever deliverable suits their needs the best, and then, to transform one deliverable into another as far as it is possible. For example, sitemaps and wireframes are two of the most widely used deliverables in IA. While sitemaps deal with the structure of a website, wireframes deal with the structure of individual pages. And, although their focus is different, there is some information they can share, for example, a sitemap with a main navigation specification should produce a set of wireframes (one for each kind of page) with a main menu having the number of items specified in the sitemap.

In this context, metamodelling is a key activity to, on the one hand, describe explicitly deliverables, and on the other hand, lay the foundations for later transformations. This paper is focused on one of the most widely used deliverables in IA, sitemaps, and it proposes its explicit description by means of a metamodel. The metamodel specifies the sitemap building blocks and its relationships. Furthermore, in order to obtain a preliminary sketch of the web site focused on the kind of navigation that is specified in sitemaps, a set of model to text transformations are defined. The transformations will produce a set of `HTML` files that will enable users to get a feel for the site structure, without getting distracted by content.

The rest of the paper is structured as follows: Section 2 introduces sitemaps as an IA deliverable. Then, in Section 3, a metamodel for defining the sitemap building blocks is specified. The model to text transformations are explained in section 4. This section has been divided in three subsections that are focused on the piece of transformations needed to generate the different functionalities of navigation (main, local or utility navigation). Finally, the paper is concluded and some further work is introduced briefly.

2 Sitemaps

The term site map can have different meanings depending on the context [8]. Mainly, there can be three different meanings for sitemap: (1) a navigation mechanism used by site visitors; (2) site maps that are derived from web sites, containing the list of the different pages that can be found on the site; and, (3) a deliverable that documents the architecture of a site usually used by developers and designers. In this section we are going to focus on the deliverable, which is one of the most widely used web information architecture tool for showing the overall structure and hierarchy of a web site.

As deliverables, sitemaps define structural navigation in web sites and one of their main values is that they allow designers to anticipate information organization errors. As a consequence, these errors can be corrected prior to time and money investment. But, in spite of their popularity, there is no a standard best practice for creating sitemaps. The initial sketches are usually drawn by hand, and then, a neater version could be drawn with some kind of visual editor. Some information architects and designers use Visio to lay out pages hierarchies and create connections between them, but they could be also created in Word, Power Point or any other tool the designer is comfortable with. Sitemaps together with wireframes provide the framework upon with to base site navigation.

The most basic sitemap [3] is similar to an org-chart where boxes represent pages or different areas of the site (Figure 1 shows an example of a simple sitemap). Boxes are connected by lines that show the semantic relationships between the areas of the site and may also represent navigation, but they don't have to. For example, in some sitemaps lines represent relationships between content without explicitly suggesting navigation. The place and connections imply that there is a hierarchy between pages, in such a way that the site's home page is usually at the top of the chart.

Sitemaps can be represented in a coarse-grained or a fine-grained way. On the one hand, high-level sitemaps (also called blueprints) show how the main sections of a web site fit together. They do not show all pages. On the other hand, detailed sitemaps document the site finer-grained. As it is impractical to diagram hundreds of pages, detailed sitemaps are written in separated documents, one for each sitemap section.

As it has been mentioned previously, there are no rules on how sitemaps should look like, however, there are some common elements [8] that usually are represented in many of them:

- *Node*. Pages in the site are basic nodes in sitemaps. Usually they are depicted as squares.
- *Connector*. Nodes are connected in order to show their relationships. Sitemaps don't show the associative links, but structural and utility navigation.
- *Numbering Scheme*. It is used to give each page a unique identifier, and usually it is hierarchical (See Figure 1).
- *Labels*. Each node has a title which corresponds to the navigation label for that page.

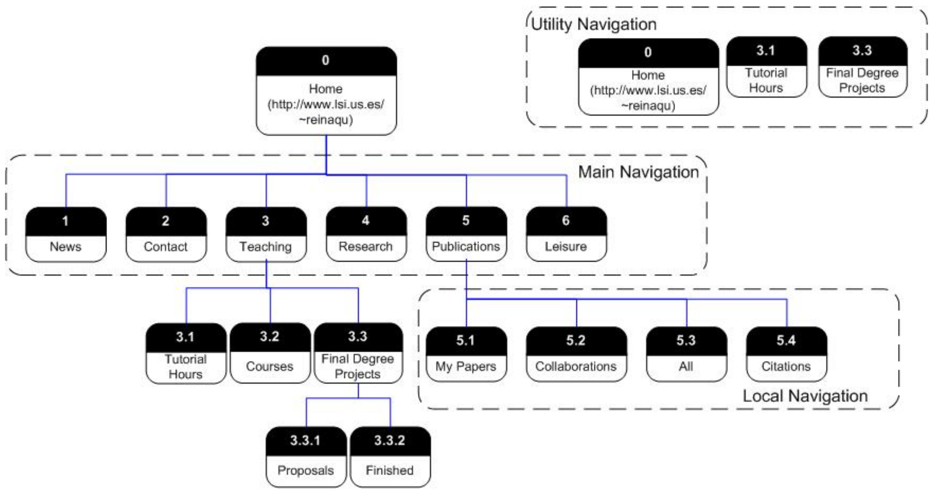


Fig. 1. A high-level sitemap for a researcher web page

– *Page Attribute*. In addition to the title, other page characteristics can be indicated, such as, the content format (HTML, PDF); pages that are to appear in pop-up windows; dynamic content; access rights; the page type and page template; or functionality and special features.
– *Notes and Annotations*. They are used to express some exceptions or special conditions that aren't able to communicate visually.
– *Scope*. Show pages that are in scope or out of scope for the current project.
– *Title and Key*. The site map should have a title and a version number or date.

3 Sitemap metamodel

Site structure and navigation are related, but they aren't the same thing. A detailed site map shows all pages of a site, but the navigation is a limited view into that structure. From any given page in the site, navigation is a constrained window of all available pages [8]. For designing our sitemap metamodel three main assumptions have been made:

1. Sitemaps are going to be modeled in a coarse-grained way.
2. Not all pages in a site are going to be modeled, but the ones represented in the sitemap imply navigation. That is, if there's a relationship between two nodes, then there will be a link in the web site.
3. Only a minimum set of the common sitemap elements introduced in the previous section are going to be modeled, in order to simplify the metamodel and to be the least restrictive as possible.

In order to guarantee these assumptions, the sitemap metamodel shown in Figure 2 has been proposed. Its root element is the `Sitemap` metaclass. One sitemap is composed of zero or more sitemap elements. This is represented by means of the reference named `elements`. `SitemapElement` is the abstract metaclass that contains the properties that are common to all the sitemap elements. That is, every sitemap element has an `id` and a `name`. There are two kinds of elements: `Node` and `Area`. One node represents a page or set of pages in a web site, while an `Area` models a set of nodes that have a similar navigation function. `Node` and `Area` are subclasses of the `SitemapElement` metaclass. The `id` attribute represents a unique identifier, and, in the case of nodes, it is usually a hierarchical numbering scheme. The root of the sitemap has been modeled as a specialization of the `Node` metaclass named `Home`.

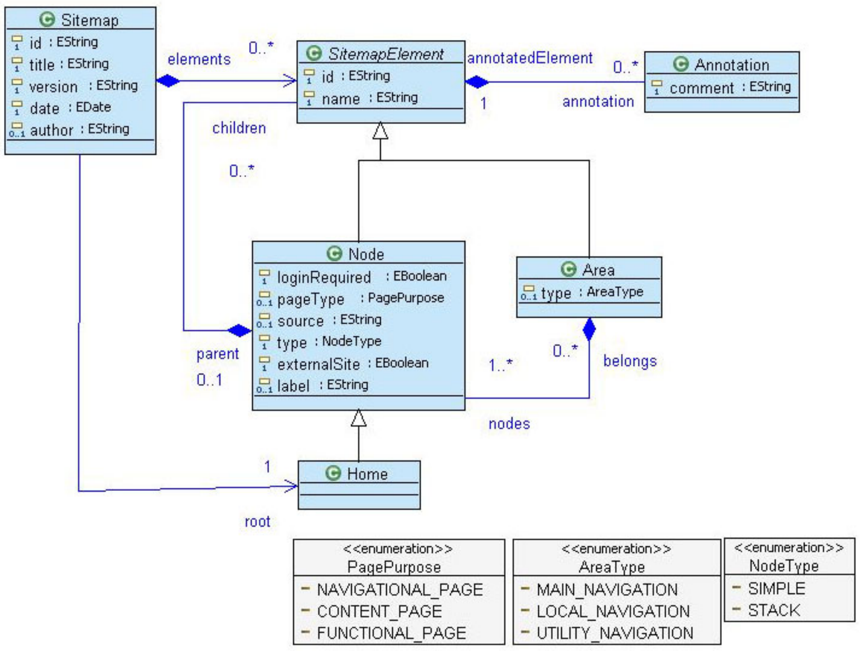


Fig. 2. The sitemap metamodel specified in EMF

Areas and nodes can have a parent (`parent` association) or not. For example, areas that represent utility navigation do not have a parent. This fact has been modeled by means of an OCL constraint. OCL constraints have not been included in Figure 2 in order to improve its readability. All the OCL constraints attached to the metamodel are explained below. One node can access to its children by means of the `children` reference (the opposite to `parent`). Furthermore, one `Area` can contain one or more nodes (`nodes` reference). Figure 1 shows three areas that have been labeled as `Main Navigation`, `Local Navigation` and `Utility Navigation`, respectively.

Annotations are the mechanism used to attach additional information to a sitemap element. The `Annotation` metaclass is used to model annotations. The additional information is textual and it is modeled by means of the `comment` attribute. The association `annotations` represents that one sitemap element can contain zero or more annotations. As this association is bidirectional, an `Annotation` can access to its `SitemapElement` via its opposite reference, `anno-tatedElement`.

In addition to the metaclasses, three different enumerations have been defined (`PagePurpose`, `AreaType`, `NodeType`). `PagePurpose` specifies the main aim of a node or set of nodes. According to this criterion, pages (and, as a consequence, nodes) can be classified in three main categories [8]: navigational pages, content pages and functional pages. The aim of navigational pages is to show the user the path to its final destination. Examples of this kind of pages are the home page or some gallery pages. Content pages are those ones that users are looking for, while functional pages are those ones that allow users to finish certain tasks, such as searching for some information or sending an e-mail.

`AreaType` defines the navigation aim of a set of nodes (the ones that are included in the area). In [8] three main categories of navigation aims have been defined: structural navigation, associative navigation and utility navigation. As it has been mentioned in Section 2, sitemaps represent structural navigation, that is, navigation that links pages that belong to the site hierarchy. It is supposed that every page is linked to the one that is above and the ones that are below in the hierarchy. However, our sitemap metamodel also addresses utility navigation. Utility navigation connects pages that help users to use the site. The utility navigation (or supporting navigation) is used to supplement global navigation. Usually, it contains links to pages that youd like to have on every page, but dont need to have the same visual weight as your global navigation. Usually they're out of the site hierarchy. In our sitemaps utility navigation is depicted as an area that is drawn out of the site hierarchy.

Main navigation and local navigation are two kinds of structural navigation. Main navigation (also known as global navigation) is the menu or set of links that appears on every page of the site and links to all of the main pages. Local navigation guides a user to certain sections in a long page.

Finally, `Node Type` indicates if the node represents a simple page (`SIMPLE`) or a similar set of pages (`STACK`). Editors usually represent this by using two different shapes.

In order to maintain the metamodel as simple as possible, some design decisions have been specified by means of OCL constraints. The constraints expressed in an informal way are the following ones:

- Areas whose `type` is `UTILITY_NAVIGATION` don't have a parent. That is, as utility navigation is out of the sitemap hierarchy, the utility navigation areas have no parent.

- The Home node doesn't have a parent and its property pageType should be FUNCTIONAL_PAGE, by definition.
- A node that belongs to an area has the same parent than the area in which is included, only if the area is not an utility navigation area.
- Nodes that belong to the main navigation area can't be included in other areas.
- If the node is an external site, then its source property must have a value (the URL that points to that external site).
- There only can be one area with its type equals to MAIN_NAVIGATION. That is, main navigation is unique for the entire site.

4 Transformations

If we look at, objectively, structural and utility navigation, we realize that they are essentially a list of links to various pages on the website [6]. The model-to-text transformations described in this section are based on this idea. Thus, they have as input a model that conforms to the metamodel defined in section 3 and the result is a set of HTML files that contains structural and utility navigation. This set of files will enable users to get a feel for the site structure, without getting distracted by content. Each node in the sitemap will produce an html file. Figure 3 shows one of the HTML files obtained from the sitemap depicted in Figure 1. Concretely, it is the one generated from the node whose id and name are, respectively, 5.3 and All. As it can be seen, the name of the file is composed of the node id, a dash , the name and the .html extension, and it shows only the three types of navigation that are addressed by sitemaps. The utility navigation is placed at the top, the main navigation is in the middle (displayed as a tabbed bar), and finally, local navigation is at the bottom.

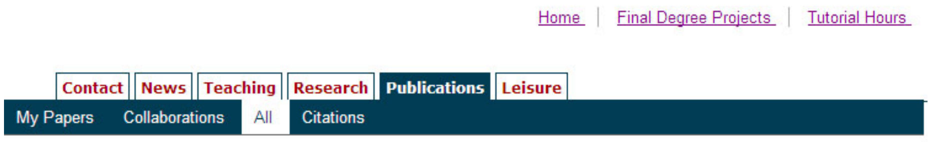
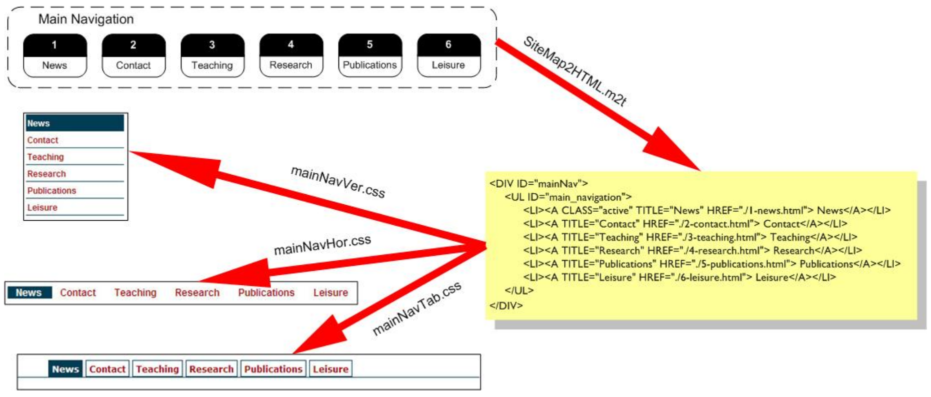


Fig. 3. File 5.3-all.html obtained from the sitemap node with id equals to 5.3

Basically, as navigation can be seen as a list of links, the transformations are going to generate HTML lists for each kind of navigation. Later, these lists will be formatted with different CSS style sheets. The transformations have been implemented with MOFScript [13]. The rest of the section gives a more in depth view of the transformation focusing on the three types defined above (main, local and utility navigation).

4.1 Main Navigation

Main navigation (also known as global navigation) [12] is a navigation that appears on every page of the site and links its main areas. This implies that the HTML list generated for this kind of navigation should be copied into each generated HTML file. Figure 4 shows the piece of HTML code that is generated for the main navigation specified in the sitemap depicted in Figure 1.



```
<DIV ID="mainNav">
    <UL ID="main_navigation">
        <LI><A CLASS="active" TITLE="News" HREF="./1-news.html"> News</A></LI>
        <LI><A TITLE="Contact" HREF="./2-contact.html"> Contact</A></LI>
        <LI><A TITLE="Teaching" HREF="./3-teaching.html"> Teaching</A></LI>
        <LI><A TITLE="Research" HREF="./4-research.html"> Research</A></LI>
        <LI><A TITLE="Publications" HREF="./5-publications.html"> Publications</A></LI>
        <LI><A TITLE="Leisure" HREF="./6-leisure.html"> Leisure</A></LI>
    </UL>
</DIV>
```

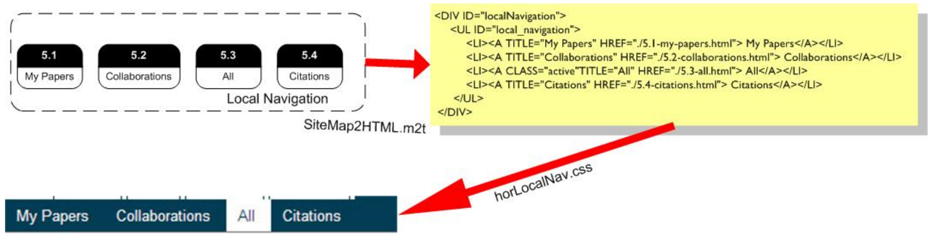
Fig. 4. Transforming main navigation.

The figure also shows how the visual aspect of the main navigation can vary only by applying different CSS style sheets. In this case, three different style sheets have been defined: `mainNavVer.css`, that displays main navigation as a vertical list of links; `mainNavHor.css`, that represents it as a horizontal bar; and, finally, `mainNavTab.css`, which shows main navigation as a tabbed horizontal bar.

4.2 Local Navigation

Local navigation [8] is used to access lower levels in a structure, below main navigation. Local navigation often works in conjunction with a global navigation system and is really an extension of the main navigation. The three more common arrangements of local and main navigation are: inverted-L, horizontal, and embedded vertical. In the first case, main navigation is usually placed at the top of the page, and local navigation is represented as a vertical link list. The second case, the one depicted in Figure 5, represents local navigation as a second row of options under a horizontal main navigation. Finally, the embedded vertical arrangement shows main navigation in a vertical menu and local navigation between the main navigation options in a tree-like structure.

Figure 5 shows the piece of HTML generated for local navigation. The CSS style sheet, displays this navigation as a horizontal bar, that it is placed below



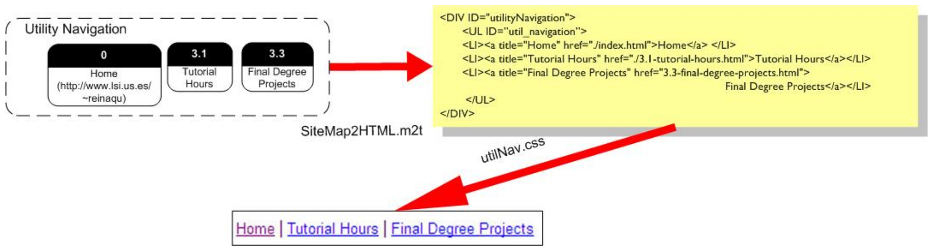
```
<DIV ID="localNavigation">
    <UL ID="local_navigation">
        <LI><A TITLE="My Papers" HREF="./5.1-my-papers.html"> My Papers</A></LI>
        <LI><A TITLE="Collaborations" HREF="./5.2-collaborations.html"> Collaborations</A></LI>
        <LI><A CLASS="active"TITLE="All" HREF="./5.3-all.html"> All</A></LI>
        <LI><A TITLE="Citations" HREF="./5.4-citations.html"> Citations</A></LI>
    </UL>
</DIV>
```

Fig. 5. Transforming local navigation.

the main navigation bar. In this case, the piece of generated `HTML` code is not inserted into every node. It should only be included in the nodes that belong to the local navigation area (nodes 5.1, 5.2, 5.3 and 5.4), the parent of the local navigation area (node 5), and, in case there are some, the children of the nodes that belong to the local navigation area.

4.3 Utility Navigation

Utility navigation (also known as supporting navigation) [12] is used to supplement global navigation. Usually, it contains links to pages that you would like to have on every page, but they do not need to have the same visual weight as the global navigation. As a consequence, the piece of `HTML` code generated for utility navigation should be included in every page.



```
<DIV ID="utilityNavigation">
      <UL ID="util_navigation">
      <LI><a title="Home" href="./index.html">Home</a> </LI>
      <LI><a title="Tutorial Hours" href="./3.1-tutorial-hours.html">Tutorial Hours</a></LI>
      <LI><a title="Final Degree Projects" href="3.3-final-degree-projects.html">
                                                  Final Degree Projects</a></LI>
       </UL>
</DIV>
```

Fig. 6. Transforming utility navigation.

Figure 6 shows the piece of `HTML` code generated for utility navigation. The visual aspect is obtained by applying the `utilNav.css` style sheet.

5 Conclusions and Further Work

One of the ways of approaching information architecture and navigation design is to share deliverables, because it is a way of having a common vocabulary. Thus it is necessary to focus on info design and architecture deliverables [2] and their building blocks, in order to contribute to the integration process of interdisciplinary projects. We propose metamodelling as a way of better understanding of the deliverables building blocks.

In this paper the focus is on one the most widely IA deliverable, sitemaps. We have proposed a minimum metamodel and a set of model to text transformations to generate a preliminar sketch of structural and utility navigation.

As further work, metamodels for other IA deliverables are needed as well as a set of model to model transformations not only to approach deliverables, but also to approach IA and current web design modeling languages. Furthermore, in parallel, we are working on a graphical editor based on Eclipse GMF to draw sitemaps models (in this first stage models have been built with the EMF default editor).

References

1. F. Azam, Z. Li, and R. Ahmad. Using UML profile for connecting information architecture and detailed information design. In *Proceedings of the IEEE Symposium on Emerging Technologies, 2005*, pages 423–428, Sept 2005.
2. P. J. Bogaards. Info design / arch deliverable schemas. Avalaible at:`http://www.bogieland.com/infodesign/resources/misc/iadelschemas.htm`, Jul 2004.
3. D. M. Brown. *Communicating Design: Developing Web Site Documentation for Design and Planning*, chapter 8, Site Maps. New Riders, August 2006.
4. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications.* Elsevier Science, 2003.
5. J. Conallen. *Building Web Applications with UML.* Addison-Wesley, 2nd edition, 2003.
6. C. Grannell. *The Essential Guide to CSS and HTML Web Design*, chapter 5. Using Links and Creating Navigation, pages 147–232. friendsof. APress, 2007.
7. inspera. Information architecture and navigation design. Avalaible at: `http://www.inspera.com/servlets/dispatcher?siteNodeId=585229&languageId=2`, 2009.
8. J. Kalbach. *Designing Web Navigation.* OReilly Media, Inc., 2007.
9. J. Kaufman. Information architecture as an extension of web design. *Digital Web Magazine*, 2005.
10. D. Schwabe and G. Rossi. Developing hypermedia applications using OOHDM. In *Proceedings of Workshop on Hypermedia Development Processes, Methods and Models, Hypertext'98*, 1998.
11. T. J. Shelford and G. A. Remillard. *Real Web Project Management. Case Studies and Best Practices from the Trenches.* Addison-Wesley Professional, 2002.
12. E. J. Stocks. *Sexy Web Design.* SitePoint Pty. Ltd., 1st edition, March 2009.
13. MOFScript Team. The mofscript home page. Avalaible at: `http://www.eclipse.org/gmt/mofscript/`.