SITEMAPS FROM A MODEL DRIVEN PERSPECTIVE
A First Step for Bridging the Gap between Information Architecture and Navigation Design

Antonia M. Reina-Quintero and Jesús Torres-Valderrama
Department of Languages and Computer Systems, University of Seville, Avda. Reina Mercedes, s/n, Seville, Spain

Keywords: Metamodelling, Model transformations, Information architecture, Navigation design, Sitemaps, Web design.

Abstract: Researchers claim that there is a disconnection between information architecture and navigation design. One way of approaching these two fields is to share deliverables. However, it is difficult to change the minds of audiences to make them use deliverables they are not used to. Thus, we propose let audiences use those deliverables they are more comfortable with, and then transform one deliverable into another, as far as possible. To get this aim, firstly, we need to have a deep knowledge of deliverables, and secondly, a set of mappings have to be defined in order to translate the information the source deliverable is covering into the target deliverable. Our approach uses metamodelling as the technique to define the pieces that compose deliverables and their relationships, and model transformations for mapping deliverables. In this context, the paper focuses on one of the most widely used information architecture deliverables, sitemaps, and its main contributions are: (1) a sitemap metamodel, which define the minimum set of elements that can be used for specifying sitemaps; and, (2) a set of model to model transformations to obtain a XHTML skeleton of structural and utility navigation.

1 INTRODUCTION

The two disciplines that face up to some of the greatest challenges when developing internet sites are Information Architecture (IA) and Navigation Design (ND). They try to answer non-technical questions such as: What kind of information should be included on a website? How should the site be organized? Or how will users be able to find their way around? IA is more related to organizing, structuring and labeling information, while ND involves technical development and visual design. However, researchers have argued that there is a disconnection between these two disciplines (Azam et al., 2005), and as a result, bad designs are obtained because information technology is too separated from the design process or too far integrated in it.

A good way to bring these two disciplines closer is to share deliverables. But there are two main problems when different audiences work with deliverables: on the one hand, consumers of documentation do not read deliverables, but they look for nuggets of information they can use to complete their own work (Curtis, 2008); and, on the other hand, it is difficult to change the mind of audiences to make them utilize deliverables they are not used to. In this context, we propose letting the members of the development team use those deliverables they are more comfortable with, and then, transform one deliverable into another, as far as possible. Thus, for example, one common scenario can be the transformation of sitemaps, one of the most widely used IA deliverables. Sitemaps specify the structure of a website, but also hold information related to structural navigation. With our approach the navigational information obtained from sitemaps will be mapped to other deliverables that deal with navigation, such as state machines or flow charts. This strategy makes easier the navigation designer's job, because he or she will not read sitemaps, but deliverables he or she is comfortable with.

Our approach is composed of two main groups of activities: firstly, we need to have a deep knowledge of deliverables, and secondly, a set of mappings have to be defined in order to translate the source deliverable into the target deliverable. To address the first group of activities, metamodelling is proposed, because metamodels capture the essential features and properties of the deliverables that are being modeled (Clark et al., 2008). The translation is obtained by means of a set of model to model transformations.

Thus, this paper is focused on sitemaps, which

are not only one of the most widely used deliverables in IA, but also one of the deliverables understood by a great number of different stakeholders. Firstly, a sitemap metamodel has been specified by means of the Ecore Modeling Framework (EMF) (Steinberg et al., 2008). And, secondly, some transformations specified by means of QVT Operational Mapping (OMG, 2005) are defined. The result of the transformations is a skeleton of a set of XHTML pages. This skeleton is focused on structural and utility navigation and it is used for validation purposes. The rest of the paper is structured as follows: Section 2 introduces sitemaps as an information architecture deliverable. Sections 3 and 4 describe the source and the target metamodel, respectively. After that, the model to model transformations are specified in Section 5. Then, a general overview of related work is introduced in Section 6. Finally, the paper is concluded and some further lines of research are pointed out.

2 SITEMAPS

Sitemaps are one of the most widely used IA deliverables and define structural navigation. 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. However, 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.

The most basic sitemap (Brown, 2006) 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 do not 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. Although there is no standard best practice, there are a set of common elements that usually are specified in many sitemaps (Kalbach, 2007): *nodes*, representing pages in the site; *connectors*, that show nodes relationships; *numbering scheme*, used for identifying pages in an unique way; *labels*, which

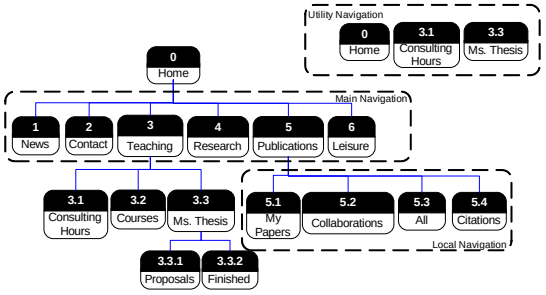

Figure 1: A high-level sitemap for a researcher web page.

represent nodes titles; *page attributes*, showing other page characteristics; *notes and annotations*; *scope* and, finally, *title and key*, that are sitemap properties.

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.

3 SITEMAP METAMODEL

Sitemaps specify site structure and navigation, two related concerns. A detailed sitemap 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 (Kalbach, 2007). The metamodel specified in this section models coarse-grained sitemaps. In addition to this constraint, three additional assumptions have been made: (1) not all pages in a site are going to be modeled. (2) The pages represented in the sitemap imply navigation. That is, if there is a relationship between two nodes, then there will be a link in the web site. And, (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[1]. The Sitemap metaclass is the root of the metamodel. One sitemap is composed of one 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

[1]The ecore file that holds the metamodel can be downloaded from: http:// www.lsi.us.es/ ∼reinaqu/ org.mwacsl/ siteMap/metamodel/SiteMap.ecore

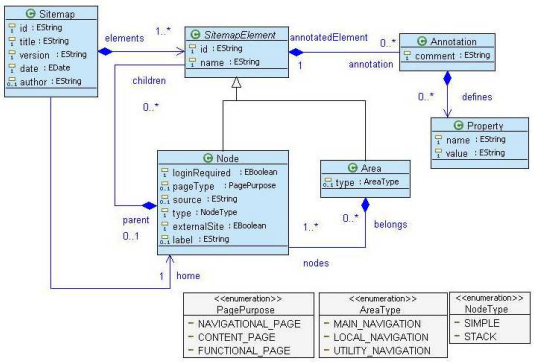

Figure 2: Sitemap metamodel.

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 SitemapElement. 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 by means of the home relationship that specify which relates Sitemap and Node.

Areas and nodes can have a parent (parent association) or not. 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 tility Navigation, respectively. The navigational aim of an area has been modeled by means of the enumeration named AreaType. These navigational aims are inspired on (Kalbach, 2007), where three main categories are defined: structural navigation, associative navigation and utility navigation. Usually, sitemaps represent structural navigation, that is, navigation that links pages that belong to the site hierarchy. We have made an extension and our sitemap metamodel also addresses utility navigation. Utility navigation connects pages that help users utilize the site and it supplements main navigation. Usually, it contains links to pages that you would like to have on every page, but do not have the same visual weight as main navigation. Utility navigation is out of the site hierarchy. As a consequence, utility navigation is depicted as an area that is drawn out of the site hierarchy (Figure 1).

The AreaType enumeration has three different values: MAIN_NAVIGATION, LOCAL_NAVIGATION and TILIT _NAVIGATION. Main navigation and local navigation are two kinds of structural navigation. Main navigation is the menu or set of links that appears on every page of the site and links to all of the main sections. Local navigation guides a user to certain sections in a long page. As it can be seen, local and main navigation have a behavior a bit different; as a consequence, they have been modeled separately. Finally, it has to be highlighted that associative navigation has not been modeled because this kind of navigation is out of the scope of sitemaps.

Annotations are the mechanism used to attach additional information to a sitemap element (Annotation metaclass). Annotations are composed of properties (Property metaclass) that represent a pair name-value which is used to express additional node properties. The annotations association 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, annotatedElement.

Finally, two enumerations have been defined: PagePurpose and NodeType. PagePurpose specifies the main aim of a node. According to this criterion, pages (and, as a consequence, nodes) can be classified in three main categories (Kalbach, 2007): 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 the ones that allow users to finish certain tasks, such as searching for some information or sending an e-mail. Node Type indicates if the node represents a simple page (SIMPLE) or a similar set of pages (STAC).

In order to maintain the metamodel as simple as possible, some design decisions have been specified by means of OCL constraints. As the metamodel has been specified by means of EMF, the OCL constraints have been added to the metamodel with the implementation of the Object Constraint Language (OCL) for EMF-based models (EMP, 2007). These constraints express the following ideas: (I) as utility navigation is out of the sitemap hierarchy, utility navigation areas do not have a parent. (II) The main navigation area must have a parent. (III) A node cannot be parent of itself. (IV) The node that is the sitemap home cannot be the children of any other node. (V) A node that belongs to an area has the same parent as the area in which is included, only if the area is not a utility navigation area. (VI) Nodes that belong to the main navigation area cannot be included in other areas. Links that belong to the main navigation area have to appear in every page, so it has no sense that nodes in the main navigation area are in other areas, because they will appear twice. (VII) If

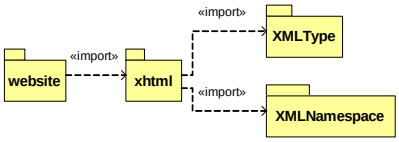

Figure 3: Metamodels Relationship.

the node is an external site, then its source property must have a value (the URL that points to that external site). (VIII) The root of the sitemap does not have a parent; its pageType property should be set to F NCTIONAL_PAGE and its type to SIMPLE. (IX) There only can be one main navigation area in the whole site. (X) Nodes and areas are disjointed elements.

4 WEBSITE METAMODEL

In order to avoid building the whole website metamodel from scratch, three other metamodels have been reused. Thus, the only package defined from scratch has been website. Figure 3 shows the relationships among the different packages which contain the metamodels. The ecore file that holds the website package has been defined manually[2], while the other three metamodels have been obtained automatically. The html, MLType and MLNamespace packages have been obtained by means of the XSD model importer component included in the EMF Eclipse plug-in, as it is mentioned in (Gronback, 2009). The importer only needs a XML Schema Definition file which defines the structure and constraints of XHTML files. One of these files is the one specified by the W3C consortium and it can be found at http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd. As a result of the importation an `xhtml. ¬ ecore` file is obtained.

Figure 4 depicts the metaclasses included in the website metamodel. The ebSite metaclass is the root of the metamodel. A website is composed of a set of TMLFile. An XHTML file holds some content. The content is modeled by means of the relationship named content. The DocumentRoot metaclass is at the other end of this relation. This metaclass belongs to the html metamodel, which has been depicted by means of the `<<from xhtml>>` stereotype. The DocumentRoot metaclass is also related to the TMLType metaclass, but as these two metaclasses belong to the html, the relation has not been depicted in Figure 4.

[2]This file can be downloaded from: http://www.lsi.us.es/∼reinaqu/ org.mwacsl/webSite/metamodel/website.ecore

In addition to this, a set of minor modifications have been done to the original `xhtml.ecore` file obtained from the importation. The AContent, Flow, Inline and TitleType metaclasses are affected by these modifications as it is stated in (Gronback, 2009). This set of changes is needed because there is no way to declare text elements between TML labels such as `<li>` or `<p>`. Finally, there is also another set of modifications whose goal is to make easier the model to model transformations that will be introduced in the next section. The AType metaclass is affected by this changes, concretely, there is an attribute named class whose type has been modified from NMTO ENS to NMTO EN.

5 M2M TRANSFORMATION

The transformation has as input a model that conforms to the sitemap metamodel explained in Section 3 and returns a model that conforms to the website metamodel explained in Section 4. The main idea behind the transformation is that, objectively, utility and structural navigation are a list of links to various pages on the website (Grannell, 2007). As the focus of the transformation itself is to obtain a draft of utility and structural navigation, the web pages that are going to be generated are only a skeleton containing the structural and utility navigation links without any other content. Thus, the generated pages are structured into different areas that have been separated by means of TML `<div>` tags (Figure 5). These tags are part of the `<body>` section and each one will have a class attribute whose value will be used later to apply some visual style, by means of Cascade Stylesheets (CSS) files. Figure 5 depicts the values of the class attributes corresponding to the different `<div>` sections. Thus, container is the class of the outer `<div>`, while the class of the inner ones are: header, utilityNavigation, mainNavigation, localNavigation, mainContent and footer.

Figure 6 shows how a node is transformed into an xhtml file. The left hand of the figure shows the

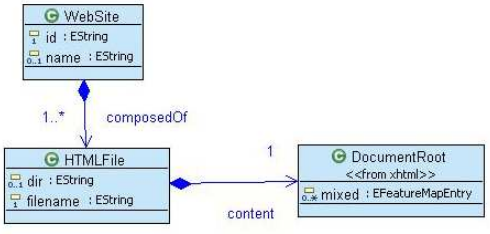

Figure 4: Website package in the website metamodel.

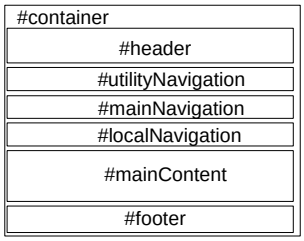

Figure 5: Structure of the generated web pages.

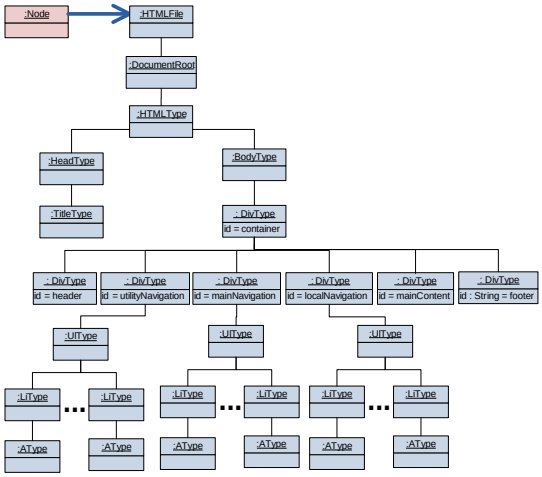

Figure 6: Result of the mapping applied to a node.

piece of the source model that is going to be transformed, while the right hand shows the result of the mapping. As it can be seen, each node generates an instance of the TMLFile metaclass. A HTML file has a content, whose type is DocumentRoot. The root of the HTML document is related to an instance of the TMLType metaclass, which represents the <html ¬ > tag. The figure also shows how different instances of the DivType metaclass are generated, one for each tag mentioned in the previous paragraph. It is also worth to be noticed that the only instances of DivType related to instances of lType are the ones that represent utility, main and local navigation. In this case, the lType metaclass models <ul> tags. Finally, each lType instance is related to a number of LyType instances. The number of instances will depend on the number of nodes included in each navigation area. Thus, for example, if the utility navigation area includes three nodes, three instances of LyType will be generated; each one will represent a <li> tag containing a link (modeled by means of AType). If we look at the node 5. (Figure 1), the XHTML file generated should looks like the one depicted in Figure 7.

The transformation has been implemented with the Operational QVT (The Eclipse Foundation, 2010) included in the M2M subproject of the Eclipse Modeling Project. The transformation[3] is composed of nine mappings, two constructors and twelve queries. Finally, it can be noticed that until now there are no details about presentation. They are specified by means of different CascadeStyleSheet (CSS) files. Thus, a set of CSS files have been defined for different presentations and layouts of navigation. This information is included in the transformation by means of four configuration properties, named cssContainer, css tilityNav, cssMainNav, cssLocalNav. Each property should hold the URI of the CSS file that will define the visual aspect of the item. For example, the property cssMainNav will have the URI of the CSS file that defines the visual aspect of the main navigation.

Figure 7: File 5.3-all.html obtained from the node whose id is 5.3.

6 RELATED WORK

As one of the aims of the approach is to bring closer two areas of research, information architecture and navigation design, different areas of knowledge have been surveyed in order to found related work. As far as we know, information architecture researchers do not formalize sitemaps, as there is no a standard best practice. The only attempt of formalizing information architecture deliverables can be found in (Bogaards, 2004), where the building blocks of many IA deliverables are specified using Dublin core attributes. Therefore, it can be stated that the focus of most of the IA publications is on describing how sitemaps can be used. Thus, for example, in (Brown, 2006), sitemaps are explained in a communication design context. In this book, and also in (Kalbach, 2007), the Visual Vocabulary defined by J.J. Garret (Garrett, 2002) is proposed as a modeling notation. This notation is broader than the one we have proposed in this paper, because in addition to sitemaps it deals with web flows. We prefer to use the separation of concerns principle and define a metamodel whose concerns are related only to sitemaps.

Concerning to Web Engineering, there are several approaches that bet for a model-driven development process such as WebML (Ceri et al., 2003),

[3]The source code can be downloaded from: http://www.lsi.us.es/ reinaqu/org.mwacsl/siteMap/transformations/sitemap2website.qvto.

UWE (Koch, 2001), OOWS (Pastor et al., 2005), OOHDM (Schwabe and Rossi, 1998).

WebML authors have not defined an explicit metamodel. The concepts managed in this approach have been defined in the XML technical space by means of a Document Type Definition (DTD). However, other authors have defined these concepts by means of an explicit metamodel (Schauerhuber et al., 2006) or a UML 2.0 profile (Moreno et al., 2007).

UWE also has been specified by means of a metamodel implemented as a UML profile (Kroib and Koch, 2008). Furthermore, a set of model to model transformations has been proposed (Koch, 2006). The transformations map Web requirement models to UWE functional models. They also define mappings between requirement models and architectural models. Finally, they provide mappings to platform specific models.

OOHDM has been extended to follow an approach based on MDA (Schmid and Donnerhak, 2005). Thus, in (Schmid, 2004) a behavioral semantics definition of the OOHDM core features and business process is made. The models proposed in this approach are used as PIM in a MDA-based development process.

OOWS has also been extended by means of a model-driven extension in order to support business processes integration (Torres and Pelechano, 2006). The business process model is the start point for transformations. This model is mapped into a default navigation model. Later, the designer has to refine this model in order to be transformed into a concrete web technology. Transformations have been specified using QVT Operational Mappings.

Other approaches that are applying a model-driven approach for web development are: HyperDE (Nunes and Schwabe, 2006), MIDAS (Cáceres et al., 2004), Moreno et al. (Moreno and Vallecillo, 2005), Muller et al. (Muller et al., 2005), W2000 (Baresi et al., 2006) and WebSA (Meliá and Cachero, 2004).

As it can be seen in the previous paragraphs, there is a great number of web engineering approaches. In order to bring closer many of these approaches, a project named MDWEnet has come up (Vallecillo et al., 2007). One of the main aims of this project is to achieve interoperability of the different model-driven web engineering methods. We also want to achieve interoperability between web engineering deliverables and information architecture deliverables. Thus, we can state that the main difference between our approach and the previous ones is that they do not have into account information architecture deliverables and concepts.

7 CONCLUSIONS AND FURTHER WORK

The main contribution of this paper is to take a step forward to bring closer information architecture and navigation design in the context of a model driven approach. The general approach consists in metamodeling and transforming by means of model transformations the deliverables used by the different audiences in a project, specially information architects and designers. The paper is centered on sitemaps, one of the most widely used IA deliverables, and it proposes a metamodel focused on structural and utility navigation, in such a way that thanks to a set of model to model transformations, a skeleton of XHTML pages is obtained. Although the aim of the transformations proposed in the paper is to obtain XHTML, this approach is being applied to other deliverables, such as statecharts or webflows.

ACKNOWLEDGEMENTS

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.

REFERENCES

Azam, F., Li, Z., and Ahmad, R. (2005). Introducing UML profile for modelling information architecture of web applications. In *Proceedings of the 11th Joint International Computer Conference - JICC 2005*, pages 245–250.

Baresi, L., Colazzo, S., Mainetti, L., and Morasca, S. (2006). *Web Engineering*, chapter W2000: A Modeling Notation for Complex Web Applications, pages 335–408. Springer.

Bogaards, P. J. (2004). Info design/arch deliverable schemas. http://www.bogieland.com/ infodesign/resources/misc/iadelschemas.htm.

Brown, D. M. (2006). *Communicating Design: Developing Web Site Documentation for Design and Planning*, chapter 8, Site Maps. New Riders.

Cáceres, P., de Castro, V., and Marcos, E. (2004). Navigation modelling from a user services oriented approach. In *Proceedings of the Conference on Advances in Information Systems (CAISE04)*, pages 150–160.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2003). *Designing Data-Intensive Web Applications*. Elsevier Science.

Clark, T., Sammut, P., and Willans, J. (2008). *Applied Metamodelling. A Foundation For Language Driven Development. 2nd Edition*. Ceteva.

Curtis, N. (2008). Audiences & artifacts. *Bulletin of the American Society for Information Science and Technology*, 34(6):20–26.

EMP (2007). EMF Technology OCL Project. Avalaible at: http://www.eclipse.org/emft/projects/ocl‡ocl.

Garrett, J. J. (2002). Visual vocabulary for information architecture. version 1.1b. Avalaible at: www.jjg.net/ia/visvocab.

Grannell, C. (2007). *The Essential Guide to CSS and HTML Web Design*, chapter 5. Using Links and Creating Navigation, pages 147–232. friendsof. APress.

Gronback, R. C. (2009). *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Addison-Wesley Professional.

Kalbach, J. (2007). *Designing Web Navigation*. O'Reilly Media, Inc.

Koch, N. (2001). *Software Engineering for Adaptive Hypermedia System: Reference Model, Modeling Techniques and Development Process*. PhD thesis, Ludwig-Maximilian-Universitat Munchen.

Koch, N. (2006). Transformations techniques in the model-driven development process of uwe. In *In Proc. Of 2nd Model-Driven Web Engineering Workshop (MDWE 2006)*, volume 155.

Kroib, C. and Koch, N. (2008). UWE metamodel and profile: User guide and reference. Technical Report 0802, Ludwig-Maximilians-Universitt Mnchen (LMU).

Meliá, S. and Cachero, C. (2004). An MDA approach for the development of web applications. In *Proceedings of the International Conference on Web Engineering (ICWE04)*, pages 300–305.

Moreno, N., Fraternali, P., and Vallecillo, A. (2007). WebML modelling in UML. *IET Software*, 1(3):67–80.

Moreno, N. and Vallecillo, A. (2005). A model-based approach for integrating third party systems with web applications. In *Proceedings of International Conference on Web Engineering (ICWE'05)*, number 3579, pages 441–452.

Muller, P. A., Studer, P., Fondement, F., and Bézivin, J. (2005). Platform independent web application modeling and development with Netsilon. *Software and System Modeling*, 00:1–19.

Nunes, D. A. and Schwabe, D. (2006). Rapid prototyping of web applications combining domain specific languages and model driven design. In *Proceedings of International Conference on Web Engineering (ICWE'06)*.

OMG (2005). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Technical report, OMG.

Pastor, O., Pelechano, V., Fons, J., and Abrahao, S. (2005). Conceptual Modelling of Web Applications: the OOWS Approach. In *Web Engineering. Theory and Practice of Metrics and Measurement for Web Development*, pages 277–302.

Schauerhuber, A., Wimmer, M., and Kapsammer., E. (2006). Bridging existing web modeling languages to model-driven engineering: A metamodel for webml. In *Proceedings of the 2nd Int. Workshop on Model-Driven Web Engineering (MDWE'06)*.

Schmid, H. A. (2004). Model Driven Architecture with OOHDM. In Matera, M. and Comai, S., editors, *ICWE Workshops*, pages 12–25. Rinton Press.

Schmid, H. A. and Donnerhak, O. (2005). OOHDMDA. an MDA approach for OOHDM. In *Proceedings of International Conference on Web Engineering (ICWE'05)*, number 3579, pages 569–574.

Schwabe, D. and Rossi, G. (1998). Developing Hypermedia Applications using OOHDM. In *Proceedings of Workshop on Hypermedia Development Processes, Methods and Models, Hypertext'98*.

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2008). *EMF: Eclipse Modelling Framework, 2nd Ed.* Addison-Wesley.

The Eclipse Foundation (2010). Operational QVT. Available at: http://www.eclipse.org/m2m/qvto/doc/.

Torres, V. and Pelechano, V. (2006). Building business process driven web applications. In Dustdar, S., Fiadeiro, J., and Sheth, A., editors, *Proceedings of the BPM2006*, volume 4102 of *Lecture Notes in Computer Science*.

Vallecillo, A., Koch, N., Cachero, C., Comai, S., Fraternali, P., Garrigs, I., Gmez, J., Kappel, G., Knapp, A., Matera, M., Meli, S., Moreno, N., Pröll, B., Reiter, T., Retschitzegger, W., Rivera, J. E., Schauerhuber, A., Schwinger, W., Wimmer, M., and Zhang, G. (2007). MDWEnet: A practical approach to achieving interoperability of model-driven web engineering methods. In *In Proc. of 3rd Model-Driven Web Engineering Workshop (MDWE 2007)*, volume 261. CEUR-WS.