Journal of Universal Computer Science, vol. 14, no. 8 (2009), 1711-1721
submitted: 17/11/08, accepted: 15/1/09, appeared: 28/4/09 © J.UCS

ActiveTM
The Factory for Domain-customised Portal Engines

Lutz Maicher
(Topic Maps Lab, University of Leipzig, Germany
maicher@informatik.uni-leipzig.de)

Benjamin Bock
(Topic Maps Lab, University of Leipzig, Germany
bock@informatik.uni-leipzig.de)

Abstract: Our goal is increasing the users' value and experience *and* decreasing the implementation time for web portals. To achieve this goal we adopt a subject-centric perspective on information architecture. The fundament of this approach is that portals should be driven by subject-centric models of the portals' domains. Out of these domain models, the interaction and interface design of the portals is self-evident. Amongst others, the international industry standard Topic Maps is a portal technology and an implementation of the subject-centric modelling paradigm. With ActiveTM we introduce a technology, which implements a Model-driven approach to automatically create domain-customised, subject-centric portal engines, based on Topic Maps. Extending ActiveTM, Information Shapes provide domain-specific views of the ActiveTM data, increasing the reusability of view components. ActiveTM and Information Shapes have proved as techniques for reducing the implementation cost of portals enormously and the implied subject-centricness increases the users' value and experience significantly.

Keywords: subject-centric, Information Architecture, Ruby, Topic Maps, Ruby Topic Maps, RTM, ActiveTM, Shapes, Information Shapes, Semantic domain models, Web 2.0, portal federation, XML, XTM, model-driven development, localisation, internationalisation
Categories: D.2, H.1, H.5

1 The Benefits of a Subject-centric Information Architecture

The Information Architecture Institute defines Information Architecture (IA) as "*the structural design of shared information environments*".[1] Further, it is defined as "the art and science of organising and labelling web sites, intranets, online communities and software to support usability and findability". The efforts of IA are focused on two goals: increasing the value of the users *and* decreasing the costs building and managing information systems.

Though years of discussing are passed by, today's portals or websites are often still driven by documents or fixed structures. Not the subjects of the content drive the users' experience and navigation, but strict hierarchies of sites reflecting organisation charts or arbitrary content categorisations. David Weinberger's discussion about the

[1] http://iainstitute.org/en/about/our_mission.php (April 7, 2008)

miscellaneousness of everything [We07] illustrates impressively the power which can be unleashed when disburden the information systems from these rigid structures.

In light of this potential we argue, that behind the scenes IA must change to a *subject-centric* information architecture. What does it mean? It is obvious, that a portal anyway represents a domain. Consequently, the portal must be driven by the *domain model*. Each domain consists of different subjects which are variously *intertwingled* by differently typed relationships. Such subject-centric domain models must define the interaction and interface design. These portals, reflecting the domain as it is, will directly result in an increased experience, with added value for the users. This value increasing effect has already been proofed empirically [OP07].

Within this article we further show that the subject-centric approach allows decreasing the implementation time and costs for new websites or portals. In ActiveTM, the Ruby based technique presented here, the ontology of the domain models is directly used for the model driven production of a domain-customised portal engine.

Summarised, building websites or portals around the subject-centric domain models they represent, will increase the portals' user value and experience *and* simultaneously decrease the implementation time and costs for these information systems.

The remainder of this paper contributes the following. In chapter 2 we sketch an existing reference model for information architecture, which reveals the central position of the domain models in the development process of the sites. In the following chapter 3 we introduce the subject-centric approach for domain modelling and present Topic Maps as one implementation of this approach in short detail. In this context, the term *subject-centric information architecture* is defined. In chapter 4 the relationship between the subject-centric domain models and the resulting subject-centric interaction and interface design of the driven websites is shown by the example of the implemented TMportal *Musica migrans*. Chapter 5 consolidates the discussion in ActiveTM, the Ruby based factory for subject-centric, domain-customised portal engines. In the concluding chapter 6 it is fast forwarded to the time when TMportals will start gossiping by using the built-in information exchange and integration facilities of the underlying portal engines.

2 The Elements of User Experience

As already discussed above, IA is defined as the "the structural design of shared information environments". Under the title "Elements of user experience" Garrett [Ga02] introduced a reference model for the process of implementing information environments. The whole process consists of five panes, starting from the definition of the strategy and ending up with the surface design. Remarkably, the whole process is always viewed from two perspectives: on the one hand the produced portal is considered as hypertext system, on the other hand as software interface.

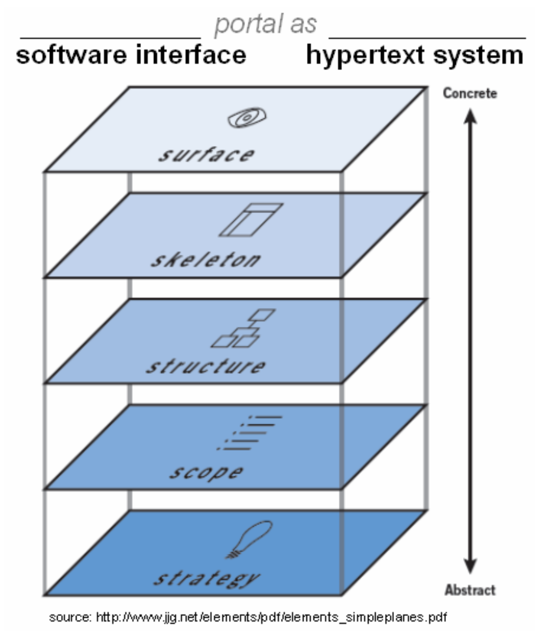


Figure 1: Elements of User Experience according to Garrett [Ga02]

Figure 1 summarises the reference model, more details are given in [Ga02]. Here we want to discuss the importance of the *structure* pane. This pane defines the domain model. It is the linchpin of the whole reference model. Most agreements in the strategy and scope pane will be condensed in the domain model, and the interaction and interface design developed in the skeleton and surface panes completely depend on the domain model defined in the structure pane. By exposing the importance of the structure pane, Garrett's reference model is very well suited as basis for a subject-centric information architecture.

But the weak point in Garrett's approach is the laissez-faire concerning the modelling technique to apply for creating the domain models in the structure pane. In the next chapter we introduce the subject-centric modelling paradigm as proposed fundament for the structure pane.

3 The Subject-centric Modelling Paradigm and Topic Maps

Each domain consists of a set of subjects. A subject is *"anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever"* [TMDM]. Summarised, a subject is anything that can be a topic of conversation. According to the subject-centric modelling paradigm, for each relevant subject exactly one proxy is created within the domain model. All information about a certain subject is always appended to the according proxy. In most cases these information are relationships to other subjects. Within the domain models, these relationships are represented as associations between the according proxies. As a result, each proxy becomes the unique information access point for all information about its subject. And the domain models will become highly interlinked.

ISO 13250, the international industry standard Topic Maps (TM) is an implementation of the subject-centric modelling approach. A topic map is a subject-centric domain model, consisting of *topics*, as subject proxies, and *associations* between them. Each topic can represent a set of typed names for the subject. Furthermore, occurrences allow representing typed properties of the subject. The associations between the topics are typed, role-based and n-ary. Summarised, TM provides a subject-centric modelling approach *and* a full set of basic modelling constructs, which is called a *legend* [DN07], like names, occurrence and full-featured associations for convenient domain modelling. For a deeper introduction into TM we refer to [AM05, Ma07]. Further we recommend a methodology for domain modelling in TM proposed by Garshol [Ga07].

Besides the expressive and flexible modelling constructs, TM provides a powerful *integration model* [Ma07b, TMDM]. This integration model assures that two topics representing the same subject will always be merged. Hence it is guaranteed that in a topic map there is always only one information hub for each subject. This powerful integration model is the fundament for the usage of TM as integration technology.

In the following chapter we demonstrate how the interaction and interface design of subject-centric TMportals are directly driven by the underlying subject-centric domain model, a topic map.

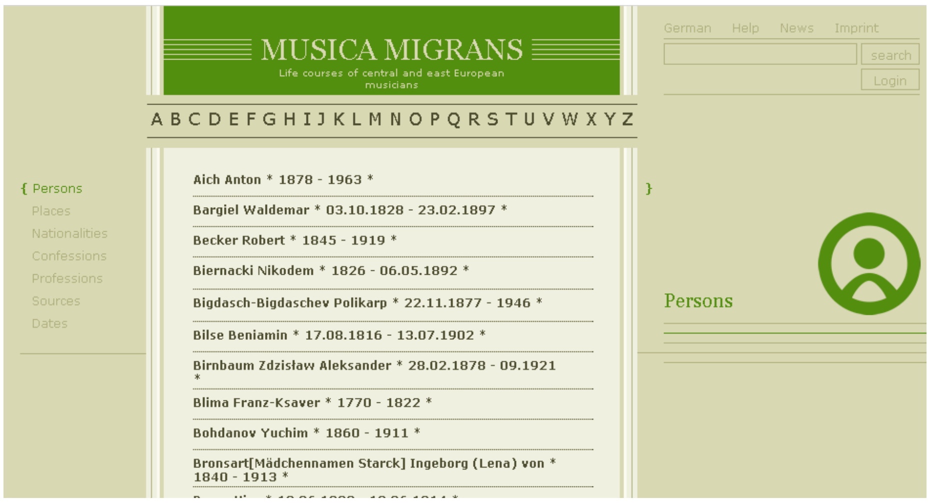
4 TMportals – how the Subject-centric Domain Model drives the Interaction and Interface Design

The interaction and interface design of a subject-centric information environment are directly driven by the *ontology* of the underlying domain model. A domain ontology defines the main subject types and the main relationships in the domain [Ga07].

A subject-centric portal mainly consists of two types of pages: index pages for each type in the domain ontology and individual pages for each subject in the domain model [Tr08].

Musica migrans[2] [Ma08] is a TMportal[3] designed and implemented by the Topic Maps Lab[4] at University of Leipzig.

[2] http://www.musicamigrans.de (April 7, 2008)
[3] http://www.fuzzzy.com/tag/?id=2238 (April 7, 2008) for links to further TMportals
[4] http://www.topicmapslab.de/


MUSICA MIGRANS
Life courses of central and east European musicians

German Help News Imprint
search
Login

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

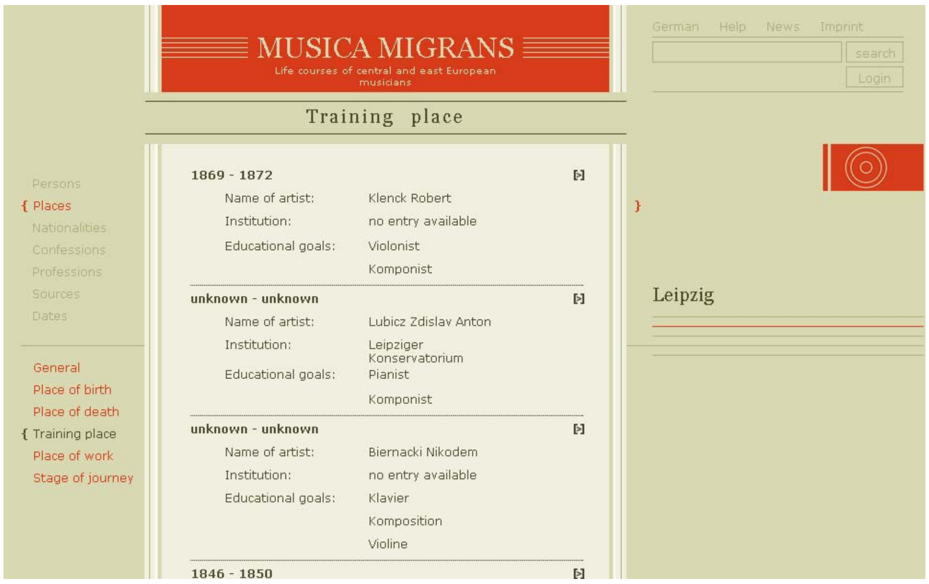
{ Persons
Places
Nationalities
Confessions
Professions
Sources
Dates

Aich Anton * 1878 - 1963 *
Bargiel Waldemar * 03.10.1828 - 23.02.1897 *
Becker Robert * 1845 - 1919 *
Biernacki Nikodem * 1826 - 06.05.1892 *
Bigdasch-Bigdaschev Polikarp * 22.11.1877 - 1946 *
Bilse Beniamin * 17.08.1816 - 13.07.1902 *
Birnbaum Zdzisław Aleksander * 28.02.1878 - 09.1921 *
Blima Franz-Ksaver * 1770 - 1822 *
Bohdanov Yuchim * 1860 - 1911 *
Bronsart[Mädchennamen Starck] Ingeborg (Lena) von * 1840 - 1913 *

Persons

Figure 2: Index page for the type "person" in the TMportal Musica migrans

The domain of this TMportal consists of the life courses of Eastern European musicians in the 19th century. The figure above depicts the *index page* for "person", the main type in the domain ontology. Each entry in the index provides a link to the subject page of the according person.


MUSICA MIGRANS
Life courses of central and east European musicians

German Help News Imprint
search
Login

Training place

Persons
{ Places
Nationalities
Confessions
Professions
Sources
Dates

General
Place of birth
Place of death
{ Training place
Place of work
Stage of journey

1869 - 1872
Name of artist: Klenck Robert
Institution: no entry available
Educational goals: Violonist
Komponist

unknown - unknown
Name of artist: Lubicz Zdislav Anton
Institution: Leipziger Konservatorium
Educational goals: Pianist
Komponist

unknown - unknown
Name of artist: Biernacki Nikodem
Institution: no entry available
Educational goals: Klavier
Komposition
Violine

1846 - 1850

Leipzig

Figure 3: Subject page for "Leipzig" in the TMportal Musica migrans

The figure 3 above depicts the *subject page* for Leipzig. In the portal this page is the central information access hub for this subject, with all its facets. In the domain model, Leipzig acts as a place for birth and death of musicians, but also as place for their educational and working periods as well as for the concerts they gave. All this information is represented in the subject page rendered out of the Leipzig topic in the domain model. In the underlying topic map all relationships of Leipzig to other topics are represented as typed associations, in the front-end they come up as links to the according subject pages in the portal. Out of the networked domain model a strongly interlinked portal is automatically produced.

After walking away from understanding the basic idea of subject-centric domain models to concrete portals driven by them, we want to define the term subject-centric information architecture *as the structural design of shared information environments with subject-centric interaction and interface designs, based on subject-centric domain models.*

Subject-centric information architecture is an approach to increase the users' value and experience of information environments. But only if supporting software exists, this approach has the full potential to revolutionise portal creation. In the next chapter we introduce ActiveTM, which provides the required technology.

5 RTM – the Ruby Topic Maps Engine

A TMportal like *Musica migrans* is based on a Topic Maps engine. In general, a TMengine provides access to the standardised data model using a comprehensive Application Programming Interface (API). This API allows programmers to create and modify topic map structures while transparently maintaining semantic consistency through the standardised integration model. The information is kept in memory or persistent storages [Ma07a, ch. B.8]. An advanced TMengine may have specialised facilities for convenient web application development [Bo08b, ch. 5].

Ruby is an interpreted, object-oriented programming language, which has gained remarkably popularity in the web application community within the last years. With Ruby Topic Maps (RTM)[5], we developed a TMengine in and for the Ruby programming language, to fuse the subject-centric approach with appropriate web technologies. For more details about RTM we refer to [Bo08a].

RTM is an implementation of the ISO-standardised Topic Maps data model [TMDM] and features an in-memory as well as a persistent database back-end. It relies on the object-relational mapper ActiveRecord which is part of Ruby on Rails[6]. The persistent back-end supports all databases supported by ActiveRecord.[7]

RTM is developed using Model-Driven Development (MDD) techniques at two layers of abstraction. The standardised Topic Maps data model is implemented in a custom Domain-Specific Language (DSL) which describes all TMDM items, like topics, occurrences, or associations, including their properties, equality rules, constraints, and the rules for merging. It also includes the core relationships defined by the TMDM, namely type-instance and supertype-subtype. At this layer of abstraction, RTM provides a non-conventional, but convenient way to create generic

5 http://rtm.rubyforge.org/ (April 7, 2008)
6 http://rubyonrails.com/ (April 7, 2008)
7 Databases currently fully supported: MySQL, PostgreSQL, SQLite, SQL Server, Sybase, and Oracle.

TMengines, in any required programming language, as well as for Ruby. The main bottleneck of such generic TMengines is that the provided APIs stick strictly with the terms used in the TMDM (the legend), *but they do not provide convenient programming interfaces for a certain domain*

6 ActiveTM – the Model-driven TMportal Engine Factory

ActiveTM provides an innovative solution to create a convenient programming interface for a certain domain, based on the ontology of the domain. Additionally, it allows sophisticated code generation based on the domain model.

The domain ontology, which for example should drive a certain portal, can be specified by using a custom DSL called ActiveTMML (Active Topic Maps Markup Language). ActiveTMML is the basis not only for the Topic Maps ontology but also provides an object-oriented and a relational view on the modelled data. This allows the reuse of the model and the information saved therewith in the back end beyond ActiveTM itself. These different technical perspectives enable interoperability between different software components beyond generic Topic Maps formats.

Based on an ActiveTMML model, automatically, the source code of a full-fledged TMportal engine is produced, which is *customised* to this domain ontology. If, for example, the ontology specifies, that there are persons which might have relationships to their place of birth, the customised TMportal engine does not only have generic TMengine interfaces to create topic and association objects, likewise classes for representing persons are available. These domain specific classes provide convenient getter and setter methods for handling the relationship to the birth place, which may be a simple characteristic as well as a fully specialised domain object.

ActiveTM provides many predefined facilities to define and generate all common characteristics and relationships of domain objects. The domain model classes and their methods are generated completely automatically. Using these classes and methods, the web application developers have a convenient interface directly to the domain model.

Additionally, creating highly customised characteristics and relationships is easy. This extra functionality is available both for generation of domain classes and within single generated classes. Including domain constraints inside the definition of characteristics and relations aids the validation of user-input at a central place.

With the features described, ActiveTM is a comprehensive solution for the Model part of a Model View Controller (MVC) architecture. An obvious step is the extension using a generic controllers and views. A common way to use structured data is the cycle of create – read – update – delete, also known as CRUD. ActiveTM provides a web interface to perform CRUD operations on an ActiveTMML model. With minimal view annotations, this CRUD interface can serve as an administrative frontend for experienced users.

This Model-driven approach for producing domain-customised portal engines has proved as technique for reducing the implementation time of subject-centric portals enormously.

7 Subjects in Shapes

InformationShapes, or shorter *Shapes*, are domain- and context-specific views of informations. Organisationally, Shapes are (part of the) View component within a MVC-Architecture. A Shape is a representation of some information in a specific context for a specific type of publication or a specific publication channel. Shapes are used to represent any information from a model, specifically ActiveTM, in a user-defined way.

With the addition of Shapes the generic administrative interface introduced in the previous chapter becomes more domain-centric. While the relative importance of the properties of the subjects is not obvious to ActiveTM, it becomes so with Shapes. The representation of a person as a row in a table of all people in the data store would contain all properties of the person model. A table-row-shape for the Person-model would define that only first name, last name, email address and department should be shown. Also, the detailed views and edit forms benefit from shapes. The grouping of properties as well as the representation to use for associated subjects is essential for a well arranged interface.

Shapes can be used recursively, from a small, atomic shape at the deepest level up to rich complex shapes at a higher level. A small shape is for example the name of a person (composed of academic title(s), firstname(s) and last name), the email address of a person or a link to the web site of a person. A bigger (and possibly more complex) shape could for example contain biographic information, composed of a history of professional positions and educational certifications.

From a simplified technical perspective, Shapes are specialized HTML-Templates. Shapes are defined using the Shape Definition Language (SDL) which basically works like a tag library in Java Server Pages (JSP). Instead of Java-Code, the SDL-code allows to identify subjects and their relevant property or properties. When the Shape is executed, the TMengine is queried for the information to create the rendered Shape. Calls to embedded Shapes cause these embedded Shapes to be rendered. It is possible to embed single shapes or collections of shapes. Embedding collections allows iterating over multiple properties with a single call.

Between the elements of a collection, domain-specific connectors can be used. This allows for example an enumeration of items to be separated by commas, while the last item is separated by ", and ".

Rendered Shapes can be cached to reduce response time. Using an Invalidate-After-Update strategy, only the Shapes which contain changed data have to be re-rendered. Also, if there are no requests within two change cycles, nothing has to be rendered. Of course, other cache invalidation strategies could be implemented, too.

Besides HTML, other output formats can be used. Examples are PDF, SVG and XML-based Office formats. The combination of HTML with JavaScript allows extending the functionality from passive to active components. A HTML-Form containing the controls of edit form of a domain object can be the basis of an interactive web portal or of an administrative interface. The use of predefined edit-Shapes, combined with generic controls and some generic CRUD code allows creating simple, dynamic web portals without programming skills.

A continuative concept is a Shape editor. While Shapes can be created without a special editor, it is easier and more comfortable to use an editor. Similar to the use of

IDEs for programming, an editor with syntax highlight and code completion functionality flattens the learning curve.

A query builder component allows menu-based creation of selection, filtering, sorting, pagination and atomification statements. There are two modes of operation:

- class-based selection, e.g. YYYY-MM-DD
- instance-based selection, e.g. 2008-11-27

While the former may be more appropriate for a skilled user, many users would prefer instance based selection. Even the trivial example of a date selection shows the ambiguity of instance-based selection: as soon as the day falls in the range of the months (1-12), the instance-based selection is no longer clear.

The query builder partially extends the Shapes from the view component to a controller component of the MVC architecture. Similar to the query builder, a template builder can simplify the creation of nested shapes. The provision of existing Shapes (from the set of Shapes for the current and related shapes' subjects) allows easier finding and re-using existing shapes.

The close coupling of Shapes to the model and to the context of their usage aids the normalisation of the view components and the reuse of views in a wide range of use cases.

8 Localisation through the domain ontology

As much the internet is infiltrating the world, it emancipates from English as lingua franca. Coming closer to the users by localisation is a very important issue the global internet industry is confronted with. In future, content reuse is not only serving several publication channels, but also serving these channels in several languages.

The model forms the most important biggest part of a web portal. In subject-centric web applications the menu structure and the embedded navigation of the user interfaces is closely coupled with the domain ontologies. In this case, translating the domain ontology is the translation of the embedded menu structure of the whole web application.

Furthermore, the content of subject-centric web applications is more structured than document centric applications usually are. This structure eases the organisation of the content translation. And for the part of the content which can be completely structured and formalized, automatic translation can be foreseen.

The benefit of localisation is obvious. Developing on top of subject-centric domain models, localisation of web the applications into a bunch of natural languages is straightforward or at least becomes a manageable challenge.

9 Fast forward

We have argued that a subject-centric information architecture, where the developed portals are driven by the subject-centric models of the domain they represent, will increase the users' value and usability. Furthermore we have introduced ActiveTM as technique for creating domain-customised portal engines, which allow reducing the implementation time of the systems enormously. The reusability of Information Shapes within a portal and in other applications adds additional value.

The dissemination of subject-centric information architecture and the availability of scalable and productive software technology might lead to an advent of TMportals.

Looking fast forward, the emergence of crowds of TMportals will yield a new level of information exchange in the web. This is due to the twofold interfaces of each TMportal: besides human-centric representation of all information about a certain subject in an individual subject page, the same information can always be requested as Topic Maps fragment. This TMfragment is an XML-document (using a standardised syntax) which combines all information about a certain subject in a computer-readable representation.

The built-in integration model of TM allows that any TMfragment can automatically be merged into the current back end. The integration model which must be implemented by a full-fledged TMengine assures that after merging there is always only one topic which grasps all information about a certain subject. It is obvious, that when federating distributed portals the back-ends and the front-ends can be updates automatically by using an appropriate subject-centric portal technology. The door for new content-oriented business models is wide open.

References

[AM05] Ahmed, K.; Moore, G.: *An introduction to Topic Maps.* In: The Architecture Journal 5, 2005.

[Bo08a] Bock, B.: *Ruby Topic Maps.* In: Maicher, L.; Garshol, L. M.: *Scaling Topic Maps.* LNAI 4999, Springer, Berlin (2008).

[Bo08b] Bock, B.: *Topic-Maps-Middleware. Modellgetriebene Entwicklung kombinierbarer domänenspezifischer Topic-Maps-Kompenenten.* Diploma thesis at University of Leipzig (2008).

[DN07] Durusau, P.; Newcomb, S.: *The Essentials of the Topic Maps Reference Model (TMRM).* In: Maicher, L.; Sigel, A.; Garshol, L. M.: *Leveraging the Semantics of Topic Maps.* LNAI 4438, Springer, Berlin (2007).

[Ga02] Garrett, J. J.: *The Elements of User Experience.* New Riders (2002).

[Ga07] Garshol, L. M.: *Towards a Methodology for Developing Topic Maps Ontologies.* In: Maicher, L.; Sigel, A.; Garshol, L. M.: *Leveraging the Semantics of Topic Maps.* LNAI 4438, Springer, Berlin (2007).

[Ma07a] Maicher, L.: *Autonome Topic Maps. Zur dezentralen Erstellung von implizit und explizit vernetzten Topic Maps in semantisch heterogenen Umgebungen.* Doctoral thesis at University of Leipzig (2007).

[Ma07b] Maicher, L.: *The Impact of Semantic Handshakes.* In: Maicher, L.; Sigel, A.; Garshol, L. M.: *Leveraging the Semantics of Topic Maps.* LNAI 4438, Springer, Berlin (2007).

[Ma08] Maicher, L.: *Musica migrans - Mapping the Movement of Migrant Musicians.* Presentation held at the Topic Maps User Conference 2008, Oslo. Slides available at (April 10, 2008): http://www.topicmaps.com/tm2008/maicher.pdf

[OP07] Oh, S. G.; Park, O.: *Design and Users' Evaluation of a Topic Maps-Based Korean Folk Music Retrieval System.* In: Maicher, L.; Sigel, A.; Garshol, L. M.: *Leveraging the Semantics of Topic Maps.* LNAI 4438, Springer, Berlin (2007).

[TMDM] ISO/IEC IS 13250-2:2006: *Information Technology - Document Description and Processing Languages - Topic Maps - Data Model*. International Organization for Standardization, Geneva, Switzerland. http://www.isotopicmaps.org/sam/sam-model/

[Tr08] Trainor, K.: *An Information Architecture for Frank Lloyd Wright.* Presentation held at Topic Maps User Conference 2008, Oslo. Slides available at (April 10, 2008): http://www.topicmaps.com/tm2008/trainor.pdf

[We07] Weinberger, D.: *Everything is Miscellaneous.* Times Book, New York (2007).