It's all around the domain ontologies -
Ten benefits of a Subject-centric Information Architecture for the future of Social Networking

Lutz Maicher and Benjamin Bock,
Topic Maps Lab at University of Leipzig, Germany
{maicher,bock}@informatik.uni-leipzig.de

Position Paper for the W3C Workshop on the Future of Social Networking,
Barcelona 15-16 Jan., 2009

What is around the domain ontologies?

Social networks are complex web applications

Social networks are complex, highly interlinked web applications. The rise of new web-friendly handsets and decreasing costs for air time causes a demand for a bunch of different user interfaces for *each* social network. Without having an appropriate information architecture beneath, the development teams will be completely messed-up by the increasing complexity.

It's all around the domain ontologies

We propagate that subject-centric information architectures are the solution for the evolving complexity. This approach implies that the domain ontologies of the applications are the focal points *everything* has to be geared to. The domain ontologies formalize the available knowledge about the domain of the future application, so all software automation approaches will use them. Besides this, subject-centricity implies highly normalized data-storage with few early-stage ontological commitments. This flexibility, combined with the identity-awareness of subject-centric approaches paves the way for a new generation of the development of complex web application. Summarized – it's all around the domain ontologies.

Outline

After a short introduction into subject-centric information architecture, we discuss ten benefits for it's usage in the development of complex and highly interlinked web applications, such as (mobile) social networks.

Subject-centric information architecture

Domain model is ontology + data

Part of the information architecture is the domain model behind a web application. A domain model consists of the domain ontology and the domain data.

Subject-centricity is exactly one proxy for each subject

Subject-centricity implies identity awareness within the domain models. For each relevant subject (*or:* resource) of the real world, exactly one proxy (*or:* topic, node) exists within the models. All information about a subject is appended to its proxy, as name, characteristic or association to other proxies.

Data models

RDF, Topic Maps, the Dublin Core Metamodel and other approaches implement this subject-centricity. All of these approaches provide differently grained modelling basic constructs (data models) for creating the domain models. For example, in RDF associations are typed, binary and directed, and in Topic Maps associations are typed, n-ary, and role based. In general, all subject-centric approaches are highly normalized data representations which additionally provide a new level of ontological flexibility.

Integration model

The subject-centric approach enforces an identity definition for each proxy. Each approach provides an integration model which defines how the subject equality of two proxies is determined. This subject equality means that two proxies represent the same subject. Whenever this equality is detected, both

proxies must be merged according to given rules. The integration models assure identity awareness: within each model there is always only one access point for all information about a given subject.

Differences of the Integration model in RDF and Topic Maps

The basic integration model of RDF is straightforward. Whenever two URI nodes are the same URI, they are considered to represent the same resource. Each triple store has to implement this integration model. The integration model in Topic Maps differs between Subject Identifiers (the subject is described by the URI) and Subject Locators (the subject URI is the referenced resource). Both kinds of identity identifiers are sets. If two topics have a non-empty intersection of such a set, they are considered to represent the same subject and will be (virtually) merged. Each Topic Maps engine has to implement this integration model.

Benefit 1: Subject-centric information architecture drives the user interfaces

The user interfaces of web applications with a subject-centric information architecture are completely driven by the domain ontology[1]. For each subject type (like persons, tags, places, images, etc.) two kinds of sites are provided: *list sites* and *individual sites*.

List sites

A list site represents an overview of all individuals of a given type within the domain model. Knowing the domain ontology, advanced domain-specific filter and sort mechanisms can be provided for the user customisation of the list views.

Individual sites

Individual sites represent all information about a specific individual (of given a type, i.e. a person) which is available at the related subject proxy. Each association in the domain model is rendered as link between the respective individual sites. The naming in the domain ontology is used to qualify the information and links in the individual sites.

Menu structure of subject-centric web applications

In general, the menu structures of subject-centric web applications are flat. The types of the domain ontology provide the top-level of the menu, but the most navigation is embedded by providing qualified links between the individual sites. This *embedded navigation* can be automatically generated out of the domain ontology.

As one of the main benefits of a subject-centric information architecture the user interfaces are driven by the underlying domain ontologies.

Benefit 2: Subjects in shapes

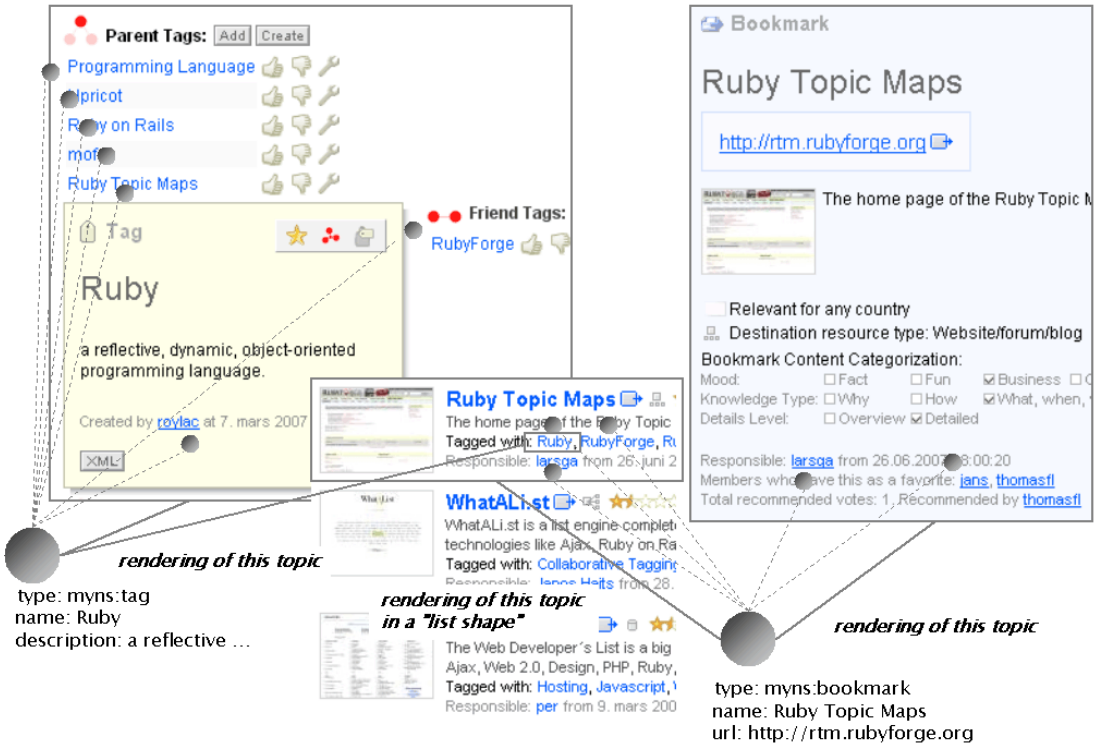
Shapes are context specific representation of subjects

In our projects, we define for each subject type different *shapes*. A shape renders all (or some) information available about a subject. Each shape satisfies a specific usage context. And each site of an application is assembled using nested shapes.

[1] One example is Musica migrans (http://www.musicamigrans.de) which is an subject-centric web application developed by the Topic Maps Lab.

The figure below illustrates this approach at fuzzzy.com, a collaborative social bookmarking tool.[2] In the domain model, the tag “ruby” has a name, a description, typed relationships to other tags, a set of assigned bookmarks and some further metadata. The left part of the figure depicts a shape of this tag which shows all related tags, providing links to the individual pages of these tags. The list in the centre of the figure shows all bookmarks which are tagged with “ruby”. For the creation of this a shape for presenting a bookmark in a list is used. On the right site of the figure the bookmark for our project “Ruby Topic Maps” is depicted, using another shape.

Examples for shapes



How the subject-centric information architecture drives the social bookmarking tool fuzzzy.com

Besides others the shape concept is very helpful for caching purposes. Once the domain model changes, all shapes influenced by these changes are directly known. Only these parts of the whole web application become invalid in the case of changes, all others can be further delivered from the cache.

Shape caching

The benefit of the shape concept is a clear separation of concerns between model and view, by preserving all knowledge about the dependencies of the subjects in the model and the views where the subject-centric information is delivered to the users. Furthermore, the design of the user interfaces is directly driven by the underlying domain models.

Benefit of subjects in shapes

Benefit 3: Identity-awareness

As discussed above, subject-centricity forces identity-aware applications. Within the domain models of the applications there will be always one proxy for each subject. Whenever subject identity of two proxies is detected by the engines, the proxies will be merged. Software architects and developers will

Identity-awareness through integration model

[2] The authors of this paper are not the developers of fuzzzy.com. This subject-centric web application is only used for illustration purposes, because it fits the scope of the workshop. The authors have no knowledge about, whether the shape concept is technically realised in this application.

be forced to develop an identity concept for the applications, which will be enforced by the engines.

Benefit of identity-awareness

The benefit of the identity-awareness is a high quality of data, at least purged from the most homonymy and synonymy problems. There is a homonymy problem if the same tag “chair” is used for a person and a piece of furniture. In subject-centric models these are different proxies with equal names. And there is a synonymy problem, if the tags ‘world’ and ‘earth’ are different proxies in the models.

Benefit 4:
Highest normalization through domain-specific APIs

Subject-centric APIs

Implementing against subject-centric domain models means using standardized APIs to create, modify and retrieve proxies, their names, characteristics and associations (RDF: Jena RDF API; Topic Maps: TMAPI). Highest normalization is gained, because there is no need to modify the schema of the persistent layer if the domain ontology changes. Adding new proxies and new associations, even with new types, is absolutely straightforward.

Domain-specific APIs

Furthermore, knowing the domain ontology (e.g. through introspection) additionally domain-specific APIs can be generated. Developers will completely implement against the objects of the domain ontologies, without any knowledge about the subject-centric technology stack beneath.

Benefit of domain specific APIs

The benefit of these APIs is the very efficient programming of applications on top of the subject-centric domain models.

Benefit 5:
Querying against the domain models

Subject-centric query languages

Besides the APIs the query languages (RDF: SPARQL; Topic Maps: TMQL) are comfortable and abstract interfaces to subject-centric domain models. Knowing the domain ontology (e.g. through introspection), queries against the domain models can be executed, without any knowledge about the technical representation of the information.

The benefit of these query languages is also a very efficient programming of applications on top of the subject-centric domain models.

Benefit 6:
Decreasing the development time with MDD

MDD and domain ontologies

Model driven development combined with generative programming approaches brings automation to software development. Having the domain ontologies as focal point of the information architecture, there are a lot of opportunities for auto generation of code. If the domain ontologies are known, there is no need to code one line by hand to get an admin interface for the domain data. It can be simply generated. The same holds for domain-specific APIs or some kinds of unit tests.

Software automation for handling diversity of user interfaces

With the rise of new web-friendly handsets and decreasing costs for air time, the demand of online interfaces for the social networks will increase. There will be a huge diversity of the emerging handsets. These developments cause a demand for a bunch of different user interfaces for *each* Social Network. It is obvious, that this diversity can only be handled with model driven approaches and software automation. Otherwise, the developers will me messed up by the increasing complexity.

The benefit of having the domain ontologies is the ability to automate the development process of the applications as much as possible. The result is a decline of developing time, a faster time-to-market and standardized, accurate code. With the launch of the "Model-based User Interfaces Incubator Group" the W3C starts evaluating the benefits the domain models based automation of user interfaces creation.

Benefit of software automation

Benefit 7: "One Liner" editing interfaces

For increasing penetration rates of mobile social network services, there is a business need for enabling low-level mobile devices to access these networks. The simplest interfaces are message-driven and text-based.

Need for simple, natural language driven interfaces

Knowing the domain ontology, controlled natural languages can be semi-automatically defined. These controlled languages will be used to provide natural language interfaces to query and manipulate the underlying domain models. We foresee natural language "one liner" interfaces for adding information to the social networks. These interfaces are completely restricted to the defined controlled languages. Guided by the interfaces, users will only be able to add phrases according to the syntax of the controlled language.

Auto-generation of controlled languages out of the domain ontologies

Such Twitter-like interfaces for structured content are not only in interest for low-level mobile devices, but also for web applications or for a tighter desktop integration by providing "one liner" deskbars for social networks.

"One liner" deskbars

The benefit of "one liners" is having a new kind of interface, which can be automatically generated from the domain ontologies. A very interesting aspect is, that with language specific "one liners" users with different natural languages will be able to modify the same domain model.

Benefit of "one liners"

Benefit 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 web emancipates from English as lingua franca

As discussed above, in subject-centric web applications the menu structure and the embedded navigation of the user interfaces is generated from the domain ontologies. In this case, translating the domain ontology is the translation of the embedded menu structure of the whole web application.

Translating the domain ontology is translating the user interface

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.

Automatic translation of formalised content

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..

Benefit of subject-centricity for localisation

Benefit 9: Integrating social networks

Integration models as generic tools for social network integration

The identity-awareness and the integration model of the subject-centric approach allow federating information from different social networks. There are several initiatives for aligning and integrating social networks, like data-portability.org or the OpenSocial foundation. In future, competing alignment initiatives as well proprietary social networks will exist in co-existence. Having the integration model of the subject-centric approach by hand, generic tools for the integration are available. All evolving commitments concerning the integration of different sites can be implemented on top of these tools.

Benefit for social network integration

The benefit of the subject-centric approach is that for each interface or API a social network provides, a virtual subject-centric view can be developed. It's like looking with the same subject-centric glasses at various sources. Using these virtual views empowers very flexible integrated facilities.

Benefit 10: Subject-centric databases at local machines

Subject-centric information architecture at local machines

The described benefits of a subject-centric information architecture are not restricted to web applications. It is also interesting for desktop or handheld applications. Having a subject-centric, highly normalized database as central "brain" of a local machine allows storing and retrieving any data in a human friendly way. Any early-stage commitments on the schema or data model of this databases are needless, highest possible ontological flexibility is permitted. Each local application can provide and use data from this central database in its application-specific usage context. Having one subject-centric database in the kernel, data from diverse social networks can be integrated for the usage at the local machine *and* local data maintained by different applications can easily be pushed into the social networks.

Benefit of going local

The benefit of central, subject-centric databases at local machines is the very flexible storage of any information which is exempted from any restricting ontological pre-conditions. And in future, these central local databases at desktop and mobile machines can be seamlessly plugged to the cloud.

About the authors

Dr. Lutz Maicher is the founder of the Topic Maps Lab at University of Leipzig. His main interests are the Topic Maps driven semantic integration of heterogeneous data and the power of a Subject-centric Information Architecture for web applications. In 2005, Lutz established TMRA – the international conference series on Topic Maps research and applications.

Benjamin Bock is member of the Topic Maps Lab at University of Leipzig. His main interest is combining the coolness of Ruby with the flexibility and power of Topic Maps. Benjamin is member of the German delegation in the ISO standardisation process of Topic Maps.

About the Topic Maps Lab

Topic Maps is a semantic technology for integrating heterogeneous data at the domain level. The Topic Maps approach is equally well suited for the smooth, domain ontology driven development of web portals. Bringing both together, the integration and the portal edges, is the main focus of the Topic Maps Lab. The Lab was founded in 2008 and is hosted by the University of Leipzig.