A Semantic Web Content Model and Repository[1]

Max Völkel
(FZI Forschungszentrum Informatik Karlsruhe, Germany
voelkel@fzi.de)

Abstract: There is currently no model that is capable of representing the *content* of web resources *together* with their semantic web meta-data. This paper presents requirements for semantic content management, a unified human-browsable and human-editable *semantic web content model* (SWCM), and its implementation *swecr*.
Key Words: meta-data, semantic web, content management, RDF
Category: H.3.7, H.5.4

1 Unifying Web and Semantic Web

This paper presents a content management meta-model combining the usability of the web with the expressivity and flexibility of the semantic web.

Although the semantic web is typically characterised as an extension of the existing web [1], there is no formal model describing the resulting mix of content *and* meta-data. The web is targeted for direct human usage e. g. by browsing web pages, but the semantic web is not. Therefore a simple merge of the two models does not result yet in a very usable model for content management.

Requirements for content management are discussed and presented in Sec. 2. The resulting model is called the *Semantic Web Content Model* (SWCM) and described in Sec. 4. Sec. 5 shows the architecture and implementation of swecr – a semantic web content repository.

1.1 The Web

REST [2] is the architectural style used for most parts of the world wide web. REST describes a set of addressable resources which are manipulated by sending self-describing representations to them (c. f. Fig. 1). One of the REST constraints is "hypertext is the engine of application state", which means each representation should contain the URIs of related resources. There is no defined way to model *typed* relations between resources, as RDF allows.

In the WWW resources are addressed by URIs and representations are character streams plus metadata describing the encoding, type of content and other

[1] Part of this work has been funded by the European Commission in the context of the IST NEPOMUK IP - The Social Semantic Desktop, FP6-027705. Part of this work has been done in *WAVES – Wissensaustausch bei der verteilten Entwicklung von Software*, funded by BMBF, Germany. Special thanks to Tim Romberg, Heiko Haller and Daniel Clemente for fruitful discussions.

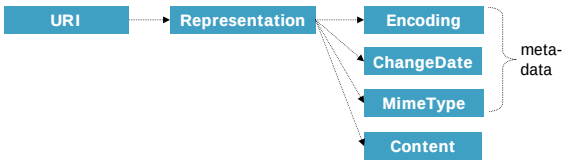

Figure 1: The Web Model: REST

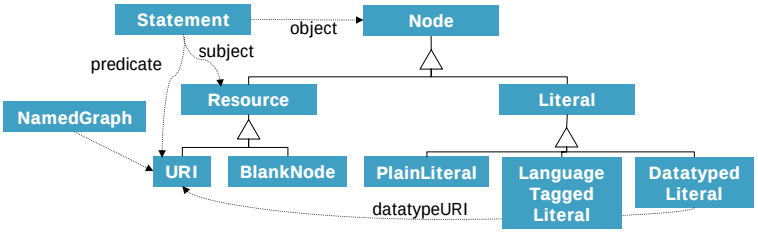

Figure 2: The Semantic Web Model: RDF

metadata such as the last modification date. In practice, there are many more meta-data fields e. g. to control caching or compression of content.

1.2 The Semantic Web

The Resource Description Framework (RDF) [3] is the basic representation format for knowledge on the semantic web. It was originally defined as a format to describe meta-data about resources on the web. Fig. 2 shows the RDF data model together with the notion of Named Graphs [4] as used in SPARQL [5]. As such it was never intended to *contain* the actual content of web resources.

Although RDF can conceptually contain binary data, stored in an `xsd:base-64Binary` data-typed literal, there is no defined way to relate URIs with content. Current RDF triple stores are not meant to store larger binary chunks either. Programming libraries for RDF lack ways to describe, access or change the content of web resources themselves.

A second problem with RDF is its lack of authoring tools. These can be divided into two classes: (1) generic: the user can change the schema at runtime, and (2) fixed-schema: the schema is pre-defined. An example of a fixed-schema tool is an address book editor which outputs its data in a fixed RDF format.

Authoring generic RDF without a pre-defined schema is very flexible, but has usability issues: e. g. each RDF resource can have none, one or multiple labels. It is an application level task to decide how to handle this. RDF can be called an *assembly language for data*, that can represent almost everything but lacks higher-order features to make it efficient for direct interaction with humans.

2 Requirements for a Semantic Content Management

Granularity (1). A Semantic Web Content Model must allow to describe web content.

The web began with small personal home pages and grew up with huge search and shopping portals. Since a few years there is a tendency for smaller content granularity, especially on collaborative websites. The term *micro-content* emerged for this set of addressable content consisting of tags (single terms), comments (often not more than a single paragraph), blog posts (often about half a page), images (including meta-data and a title) or video snippets. For most of these micro-content items, the author and the time of creation or last change are automatically logged and used for searching and browsing.

Expressivity (2). The model should be able to offer the same flexibility and expressivity as RDF to describe and relate content resources. Existing popular schema-free authoring tools such as mind-mapping or outlining tools lack the expressivity and data integration abilities of RDF, e. g. in many mind-mapping tools the user may in fact edit only strict trees. In short, existing (micro-)content management applications have low expressivity.

Compatibility (3). Furthermore, A clear path how to use the SWCM together with existing frameworks is desirable. Especially the re-use of existing background-knowledge expressed in RDF should be possible together with SWCM.

Naming (4). Names allow a user to fetch a unit of information in O(1). This is similar to know e. g. the URL of a certain web page or the file name and path of an office file. Human-usable naming is probably an overlooked area of content management. E. g. wikis allow users to use easy-to-remember names to quickly navigate or link to known pages. The semantic web is fundamentally built on URIs, which are unique names for resources. Unfortunately, they are hard to read and use for humans.

Search (5). Any content model should allow to retrieve content conveniently. Queries are usually convenient ways to retrieve a number of items fulfilling certain criteria. A SWCM needs also the ability to query the content, preferably by building on existing query languages.

Renderable representations (6). The model should be usable by end-users, hence some requirements are imposed for meta-data structures: All meta-data items should have a meaningful human-readable representation.

Mandatory inverse relations (7). In order to allow browsing semantic links in a knowledge model, links must be traversable in both directions. Therefore,

it is desirable that link types have labels for both directions, e. g. "works for" and "employs". Note: In OWL, inverse relations are allowed but not mandatory

Freedom of formalisation (8). The user needs a way to express content in an informal way, e. g. as plain text, formatted text or box-and-arrow diagrams. Then the user should be able to migrate the knowledge into more formal structures, if desired (c. f. [6]).

Access rights (9). In any system used by multiple persons, access rights soon become a necessity. Fine grained access rights management can become very complex. If there are too many resources or rights to manage, the system becomes unusable.

Versioning (10). A model supporting versioning can be used better for collaborative settings, because users do not have to be afraid of applying changes. Changes not accepted by other community members can be rolled back.

3 Related Work

A number of related content models exist. This section describes them briefly and evaluates them with respect to the requirements.

JCR [7] defines the *Java Content Repository* API (JCR), which has quickly gained much industry attention. To date, there are at least four independent implementations of this standard . JCR handles granularity well, even a mix of large binaries and small single-term words has reasonable performance (1). The expressivity of JCR is also rather high, JCR has a concept of node typing and allows to add relations between nodes (2). However, JCR allows only XPath-style [8] queries and does not allow for graph-like queries (as they are supported by e. g. SPARQL), so (3) is not met. Requirements (4), (6), (7) and (8) are not met at all. (5) is met rather well, as JCR allows to use the familiar query languages SQL and XPath. (9) and (10) are well addressed.

Subversion [9] is an open-source versioning system (10) with access rights (9) and a number of interesting properties. It can handle small text files or larger binaries, but single terms are not in the focus. Therefore (1) is only partly met. Subversion allows to attach key-value pairs to resources, but no relations to other resources, so (2) also only partly met. Subversion repositories can be browsed (6) as trees and have meaningful names (4). There are even some best-practices for naming resources in a Subversion repository[2]. Subversion offers no search (5), and does not address (3), (7) and (8).

[2] http://svnbook.red-bean.com/en/1.0/ch05s04.html#svn-ch-5-sect-6.1

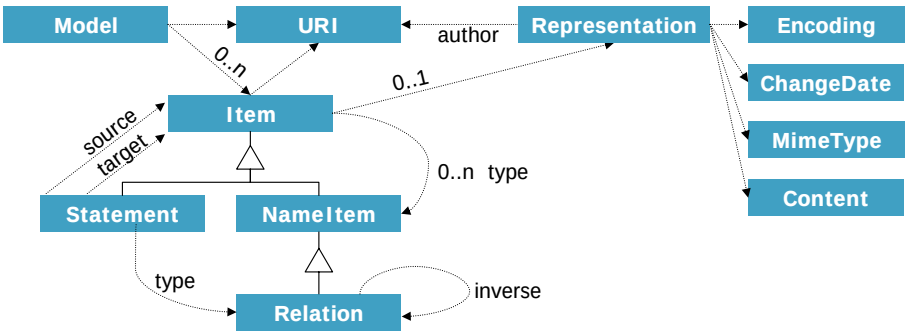

Figure 3: Semantic Web Content Model

4 A Semantic Web Content Model (SWCM)

A Semantic Web content model (SWCM, c. f. Fig. 3) consist of a set of *items*. Item is a central concept which bridges the RDF and the content world as it is both addressable via a URI and can refer to web content. All other elements in SWCM are special kinds of items. Therefore all SWCM entities are addressable and can refer to content of all sizes (Req. 1).

A *NameItem* is a sub-type of Item. It has a content snippet with the mime-type "text/plain". NameItems represent human-readable and -write-able names. No two NameItems may have the same content within a single SWCM.

A *Relation* is a sub-type of NameItem. It has always exactly one *inverse relation* defined. This makes the model much easier to browse and visualise (Req. 7). E. g. in most semantic GUIs incoming links are rendered different from outgoing links. Therefore it makes a difference for browsing whether a user stated ("FZI" "employs" "Max") or ("Max" "works for" "FZI"). For this user, this is often an artificial distinction.

A *Statement* is also modelled as a sub-type of Item. This makes it addressable and allows a user to attach content to it. A statement represents a relation from one item (the source) to another item (the target) and always has a relation type. In other words, a statement is like a typed link (Req. 2). Different from RDF, SWCM statements can be addressed themselves.

SWCM has at least the same expressivity as RDF. Each RDF statement (s,p,o) can be represented as two items (s and o) and a relation (p) with an inverse (-p). However, SWCM has some features, that RDF has not: (1) All SWCM content is addressable – this is not true for RDF literals. (2) SWCM statements are addressable, too – RDF has only a rather unclear concept of reification. The similarity between SWCM and RDF allows to convert RDF to SWCM (creating new URIs for literals and lifting them to items).

5 A Semantic Web Content Repository

This section describes the architecture of *swecr*, a *semantic web content repository* (swecr) available under BSD license from `http://swecr.org`. Swecr implements the SWCM.

5.1 Architecture

Swecr is implemented in two layers: The core layer models the state and the repository layer offers an item-centric view on the state.

The **core layer** consists of a simple binary store (BinStore), an RDF Named Graph repository and a text index.

The BinStore allows to access content in a stream and random access fashion. The BinStore interface also takes care of concurrency and allows either multiple reads or single writes. Note that many existing binary storage APIs offer no random access which makes it impossible to use such store for implementing projects such as Semantic File Systems [10].

The RDF repository is modelled as an RDF2Go ModelSet. RDF2Go is an abstraction layer over RDF triple- and quad stores which relives the programmer from choosing a single RDF store for all times. Currently OpenRDF is used as the underlying implementation.

In order to allow queries over the binary content and to speed up queries on RDF literals, Apache Lucene[3] is used as a full text index. The index stores inverse mappings for (item URI, content) and (URI, property URI, RDF literal content).

For indexing binary content, all binary content is indexed after it has been written to the BinStore. In a similar fashion all RDF literals are indexed after they have been written. The proxy pattern is used here to separate the API from the indexing. Removed resources have to be reflected in the index as well.

SPARQL queries should allow queries which access both the RDF and the fulltext index. Similar systems have been developed for Jena (LARQ [4]) and Sesame (LuceneSAIL [5]) already. The general idea is to split the query in two parts and execute them individually: One query part is delegated to the RDF store, the other part is delegated to the full-text index. Then a join is performed by the item URIs. Depending on the result set size, other join strategies should be favoured, i. e. first performing the full-text query and then binding the item URI in the RDF query to the resulting URIs. We are currently implementing this on top of RDF2Go to be independent of the triple store. Note that neither LARQ nor LuceneSAIL currently provide any API for handling binary content.

[3] http://lucene.apache.org/
[4] http://seaborne.blogspot.com/2006/11/larq-lucene-arq.html
[5] http://gnowsis.opendfki.de/wiki/LuceneSail

The core layer has no other obligations than starting, stopping and running the three core components: RDF store, text index and BinStore.

The **repository layer** has no persistent state on its own, instead all updates and queries are delegated to the core layer. This simplifies debugging and will make sharing of state easier.

The repository layer implements 1:1 the SWCM as outlined in Sec. 4. It allows to create, delete and manipulate Items, NameItems, Relations and Statements. Additionally, queries on models are provided.

The repository, however, maintains the runtime state of items. Each item can be locked. After locking, the item cannot be edited by other users. Locks time out automatically if they are now renewed by the requesting application. The requesting application may read who is currently editing a resources to be able to initiate communication processes. Such communication is outside the scope of *swecr*.

5.2 Implementation

This section describes how SWCM structures are stored in RDF. Two RDF models are used for each SWCM, one for the actual data as modelled explicitly by the user (user model) and one for the resulting plain RDF statements that can be used for queries and inferencing (index model).

An **Item** with URI x is simply stored as

```
<x> a swcm:Item.
```

The optional content of the item is stored in the BinStore. A **NameItem** with URI x and the content (name) "My Thesis" is represented in RDF as

```
<x> a swcm:NameItem;  swcm:hasContent ''My Thesis''.
```

That is, the content of NameItems is currently stored in RDF. In the future, it might instead be stored in the BinStore - this matters only for performance, ease of debugging and the implementation of the query engine. A **Relation** p (e. g. "works for") with its inverse q (e. g. "employs") is represented as

```
<p> a swcm:Relation; swcm:hasContent "works for"; swcm:hasInverse <q>.
<q> a swcm:Relation; swcm:hasContent "employs";   swcm:hasInverse <p>.
```

A **Statement** s from a to b with the relation p is represented as

```
<s> a swcm:Statement;
    swcm:hasSource <a>; swcm:hasTarget <b>; swcm:hasRelation <p> .
```

For this statement, however, some data is written also to the RDF index model:

```
<a> <p> <b>.      and      <b> <p-inverse> <a>.
```

This means that we currently materialise the inverse triples into the index model. In the future, a rule engine or reasoner might be used instead.

6 Summary

This paper presented the Semantic Web Content Model (SWCM) for content management. The model was obtained by combining the features of the expressive model of the semantic web (RDF) with the human-targeted model of the web (REST). The resulting content model has been refined in a number of ways in order to fulfill the requirements for content management.

Although the resulting model has the same expressivity as RDF, the SWCM is targeted for browsing and authoring by humans. This comes at the cost of additional constraints, e. g. all content has a URI, each relation has an inverse, there can not be two relations having the same "label", etc. The SWCM and its implementation swecr are used in the NEPOMUK-project[6] to realize two end-user modeling tools: iMapping [11] and a Conceptual Data Structures (CDS) [6] authoring tool.

The paper also presented existing approaches and its shortcomings, and an implementation of the SWCM.

Future work includes: Implementing SPARQL queries combining text index and triple store, access control, versioning, and carry out performance tests.

References

1. Decker, S., et al.: The semantic web: The roles of XML and RDF. IEEE Internet Computing **4** (2000) 63–74
2. Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine (2000)
3. Hayes, P.: RDF semantics. Recommendation, W3C (2004)
4. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. Technical report, HP (2004)
5. Prud'Hommeaux, E., Seaborne, A.: Sparql. W3C Candidate Recommendation (2007)
6. Völkel, M., Haller, H.: Conceptual data structures (cds) – towards an ontology for semi-formal articulation of personal knowledge. In: Proc. of the 14th International Conference on Conceptual Structures 2006, Aalborg University - Denmark (2006)
7. Nuescheler, D.: Content repository api for java technology specification. Technical Report Java Specification Request 170, Day Management AG, Switzerland (2005)
8. Clark, J., DeRose, S.: Xml path language (xpath) version 1.0. Technical report, W3C (1999)
9. Pilato, C.M., Collins-Sussman, B., Fitzpatrick, B.W.: Version Control with Subversion. O'Reilly Media, Inc (2004)
10. Bloehdorn, S., Görlitz, O., Schenk, S., Völkel, M.: Tagfs - tag semantics for hierarchical file systems. In: Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria, September 6-8, 2006. (2006)
11. Haller, H., Völkel, M., Kugel, F.: imapping wikis - towards a graphical environment for semantic knowledge management. In Schaffert, S., Völkel, M., Decker, S., eds.: Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics. (2006)

[6] http://nepomuk.semanticdesktop.org