A Semantic Web Content Model and Repository[1]

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

Abstract: The semantic web extends the world wide web with the ability to precisely describe meta-data of web resources and ontological concepts in a unified way. However, their is currently no model that is capable of representing the actual *content* of web resources together with their meta-data. In a similar way, there is a lack of tools for management of actual web content and abstract concepts in a unified way.

This paper presents a human-browseable and human-editable *semantic web content model* and an implementation of a *semantic web content repository* (swecr).

Key Words: metadata, semantic web, content management
Category: H.3.7, H.5.4

1 Motivation

The semantic web is currently evolving in two places: On the web and as desktop [1] applications. The Resource Description Framework (RDF) 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. The RDF model and programming libraries that implement it offer good ways to manipulate the meta-data of resources but lack ways to describe, access or change the content of resources. In fact, the RDF model does not even have the notion of content at all.

A second problem with RDF is its lack of authoring tools. Those can be divided into generic RDF authoring tools, where the user can also define the schema at runtime, and another class of tools that allow authoring only data according to fixed schema. E. g. an address book editor which outputs its data in a fixed RDF format.

The lack of generic authoring tools for RDF is likely due to the lack of some constraints that would improve the usability. As an example, authoring tools currently have to deal with RDF models that contain no labels, so items can only be displayed as inconvenient URIs.

Popular content and structure editing tools such as mind-mapping tools or outliners allow end-users to create complex models rather easily. However, they lack the flexibility of RDF (arbitrary typed arcs in a graph). In many mind-mapping tools the user may in fact edit only strict trees. Even concept mapping

[1] Paper will be shortened to 8 pages

tools that go beyond this and allow full box-and-arrow modelling still lack the smartness that ontological inferencing on top of RDF can provide. Other structured content management systems (CMS) face the same problems: Not enough flexibility and not enough "smartness".

Some approaches aim to bring RDF and content together for the benefit of end-user modelling. Examples for such tools are iMapping [2], Conceptual Data Structures (CDS) [3]. They face the problem of low or no tool support. Currently there is neither a model that allow to refer to both meta-data (RDF) and content (binary), nor a tool implementing such a model. A detailed analysis of this situation can be found in Sec. 4.

The remainder of this paper presents the *Semantic Web Content Model* (SWCM). First a definition of web content is given in Sec. 2. A number of requirements for a SWCM are listed in Sec. 3. Sec. 4 compares related work with our requirements. Section 5 presents a model that unifies RDF and content access. In Sec. 6 an architecture and implementation for a semantic web content repository (swecr) is shown. The paper concludes with Sec. 7.

2 Web Content

Typical content on the web changed in the last years. The web began with small personal home pages and grew up with huge search and shopping portals. Since 2004, blogs and wikis brought a new kind of content to the web. This change was perceived by some a such a drastic change that they considered it as a new version of the web and called it "Web 2.0".

Content in Web 2.0 is often more fine-grained than full web pages, hence the term *micro-content* emerged for this emerging 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.

The actual content of a resource is usually addressable via an HTTP-accessible URI (URL). HTTP is the widely deployed protocol for accessing a URI and finding out the mime-type of a resource and to read and write binary content streams. From these findings we define web content as a quadruple (binary content, mime-type, time, author).

2.1 Structured Text

Many content formats of Web 2.0 re-invent parts of HTML. Wikis for example allow users to create headlines, lists and some inline formatting. Most wikis however do not allow for setting the font size or colour. Many blog engines allow to use BB-code, which allows for bold, hyperlinks and few other formatting [2]. Coming from an analysis of wiki interchange problems, a wiki interchange format was defined in [4]. For exchange of semantic web content one needs a a general *structured text interchange format* (STIF) which allows only for structural and logical markup, but no purely visual effects. STIF should work as an inline format, which means it does not require a full XML document.

3 Requirements for a Semantic Web Content Model

Usually, content is stored for the purpose to be retrieved. The users are in fact storing and retrieving symbolic, externalised *knowledge cues* (c. f. [5]), that is symbols which remind a user about some knowledge.

There are several ways to retrieve content:

Content names allow a user to fetch a unit of information in O(1). This is similar to know the URL of a certain web page.

Keyword search in the full text of the content. This method is easy to implement, but has some disadvantages. Users must remember parts of the content to be able to retrieve it. If they do, retrieval can be fast ($O(1)$) or slow ($O(n)$), depending on the result set size, which is proportional to the keyword occurrence frequency.

Folders , categories or collections describe ways to group content items together. As item sets often contain other such sets (as it is the case for e. g. file systems or wiki categories), the content can be browsed in a tree. If the user browses the tree from the root and knows at each browsing step which direction to take, then item retrieval takes $O(logn)$.

An list of all content is always the last escape. Here the user simply browses through *all* items, which is $O(n)$.

In reality, the methods outlined above will be used together, leading to more complex retrieval cost functions. However, the importance of remembered keywords or remembered names becomes obvious. Therefore, the management and usage of names should be supported.

(R1) Granularity: First, a Semantic Web Content Model must allow to describe semantic web content, as defined in Sec. 2. Content granularity ranges from single terms to full (hyper-)text or multimedia resources.

(R2) Expressivity: Second, it should be able to offer the same flexibility and expressivity as RDF to describe and relate content resources.

[2] http:// www.phpbb.com/community/faq.php?mode=bbcode

(R3) Usability: Third, it 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.

(R4) Naming: In order to support intuitive access to items, human-readable and -write-able names are needed. Naming is in fact a very important part of information management. Wikis allow users to use easy-to-remember names to quickly navigate or link to known pages. Human-usable naming is probably an overlooked area of content management. The semantic web is fundamentally built on URIs, which are unique names for resources. Unfortunately, they are hard to read and use for humans. Human readable names will likely to work only within a certain context.

(R5) Search: 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.

(R6) Compatibility: 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.

(R7) Gradual Formalisation: 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 [3].

(R8) Inverse relations: 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".

4 Related Work

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

REST [6] is the theoretical underpinning of the architectural style used for most parts of the world wide web. REST describes a set of addressable resources which are manipulated by sending representations to them.

There is no defined way to model relations between resources, although REST defines "hypertext is the engine of application state". The REST models fulfills requirement (1) and (6), but fails for all other requirements,

RDF was invented as a meta-data format and hence was never intended to represent the actual content.

It fulfills requirements (2), (5) and (6). But RDF cannot represent large binary resources (1), is in its generic form not human-usable (3), has no concept for human usable names (4) or gradual formalisation (7). Inverse relations are allowed (e. g. in OWL) but not mandatory (8). In fact, RDF can be called *data assembler* language, that can represent almost everything but lacks some higher-order features to make it appealing for direct interaction with humans.

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 (Alfresco, Apache Jackrabbit, eXo, and Jeiceira).

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 (5) is not met. Requirements (3), (4), (7) and (8) are not met at all. (6) is met rather well, as JCR allows to use the familiar query languages SQL and XPath.

Subversion [9] is an open-source versioning system with a number of interesting properties. Subversion 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 as trees and have meaningful names (3). There are even some best-practices for naming resources in a Subversion repository[3].

Subversion offers no search (5), and does not address (7) and (8).

The mismatch between RDF and content management systems (CMS) is the subject of a number of blog and email postings. Mc Schraefel describes[4] semantic web content as a "note book + memex" [10].

The proposal for the Apogee project[5] describes an Enterprise Content Management (ECM) built on top of Eclipse, a JCR store, and and RDF store among other parts. The project seems to make slow progress, but has not articulated yet a clear model fulfilling requirements (1) – (8).

[3] http://svnbook.red-bean.com/en/1.0/ch05s04.html#svn-ch-5-sect-6.1
[4] http://dig.csail.mit.edu/breadcrumbs/node/184
[5] http://www.eclipse.org/proposals/apogee/

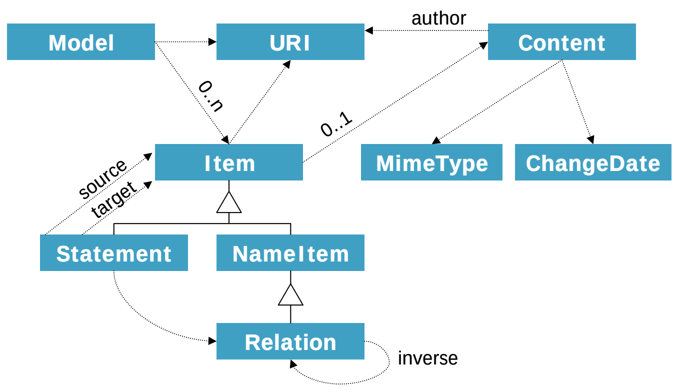

Figure 1: Semantic Web Content Model

5 A Semantic Web Content Model (SWCM)

A semantic web content model (SWCM) consist of a set of *items*. An item is a central concept which effectively bridges the RDF and the content world as it is both addressable via a URI and can refer to content as defined in Sec. 2. All other elements in SWCM are special kinds of items. Therefore all SWCM entities are addressable and can refer to content.

A special kind of item is the *NameItem*. A NameItem has a content snippet with the mime-type “text/plain”. As NameItems represent human-readable and -write-able names, no two NameItems may have the same content within a single SWCM.

A special kind of NameItem is the *Relation*. A relation is a NameItem, which has always exactly one *inverse relation* defined. This makes the model much easier to browse and visualise. 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 a kind of item, which makes it addressable and allows a user to attach content to it. A statement represents an 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. Different from RDF, SWCM statements can be addressed themselves.

5.1 Comparing SWCM and RDF

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 featurs, 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). Later, SWCM models can be re-exported as plain RDF, of course loosing e. g. relations between former literals.

5.2 Structured Text

This paper defines STIF (c. f. Sec. 2.1) as the following set of XHTML tags which may occur in plain text according to HTML nesting rules: <`h1 - h6`> for headlines, <`p`> and <`hr`> for content chunking, lists, tables, `em`, `strong`, `code`, `img` with `src`, and `a` with `href`. And `span` and `div` for other groupings of test. Finally, the `ID` attribute coming from XML can be used to give elements a local identity.

Using local XML IDs one can address parts of an items content. This is sometimes more convenient than having to split an item into three items (part before annotated text, annotated text, part after annotated text). Therefore SWCM *infers* from an element with an XML ID the existence of the annotated item. As an example, if an item x contains text in STIF, and this text contains an `span`-element with an XML ID of a and the content “z”, then SWCM infers the existence of an SWCM item with the URI xa and the content “z”. Furthermore, SWCM infers a triple x `swcm:hasPart` xa.

This *content inferencing* allows a softer migration from large to small items, as it makes parts of items addressable at low costs (no need to split items into smaller parts before parts can be annotated). This allows e. g. for semantic annotations on parts of a text.

6 A Semantic Web Content Repository

This section describes the architecture (core layer and repository layer) and the implementation “swecr” of the SWCM.

6.1 Architecture

The *semantic web content repository* (swecr[6]) 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. Fig. 2 shows the architecture of the core layer.

[6] Available under BSD license from http://semweb4j.org/swecr

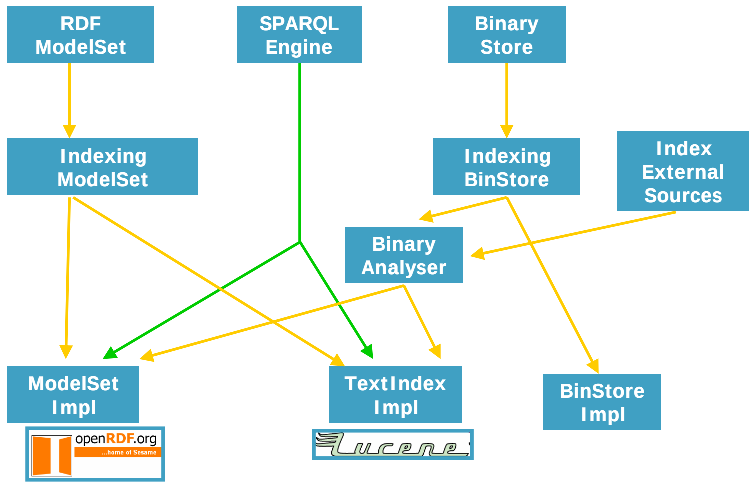

Figure 2: Swecr core architecture

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 [11].

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 OpenRDFis used as the underlying implementation.

In order to allow queries over the binary content and to speed up queries on RDF literals, Apache Lucene[7] 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 [8]) and Sesame (LuceneSAIL [9]) already. The general idea is to split the query in two parts and execute them individually: One query part is delegated to the RDF

[7] http://lucene.apache.org/
[8] http://seaborne.blogspot.com/2006/11/larq-lucene-arq.html
[9] http://gnowsis.opendfki.de/wiki/LuceneSail

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.

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. 5. 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*.

6.2 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).

Item An item with URI x is simply stored as

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

The optional content of the item is stored in the BinStore.

NameItem A NameItem with URI x and 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.

Relation 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>.
```

Statement 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.

7 Summary

This paper presented requirements for a semantic web content model. Existing work does not address the uniform management of actual content together with its meta-data. A unifying semantic web content model was presented as well as an implementation of it.

As **future work**, a number of additional issues have to be solved:

Access Rights. Fine grained access rights management can become very complex. If there are too many resources or rights to manage, the system becomes unusable. I propose to manage access rights on the level of "spaces". Each space contains a number of items and a number of users. Each user has either read-only or read- write rights. Note that the idea of spaces is almost a 1:1 mapping to Named Graphs[12].

Versioning. Two levels of data have to be considered: Item content and item structures. For each items content, we plan to employ a strategy similar to Subversion or DeltaV [10] where each version is internally managed as a distinct resource. For models, it seems more plausible to store change logs, where each change entry consists of a number of added and deleted triples. We must of course also keep track of which user performed which change, to be able to generate meaningful recent-changes-feeds from it. Swecr will allow to restore and earlier versions. For the sake of simplicity, queries will always be executed on the current (possibly restored) version.

Synchronisation. We plan to built a sync-server which allows users to share an SWCM and read it off-line, too. To allow users also to model together in near-real-time, we plan to use a dedicated synchronisation server – which could be one of the peers participating in a shared modelling session. The synchronisation receives update requests from clients which are handled on a first- come first-serve basis. Once an update is accepted by the server, it is re-issued as a change-feed to requesting clients. Hence, in an extreme case the server needs not even to look into the contents of the change requests.

[10] http://www.ietf.org/rfc/rfc3253.txt

Only the time-stamps matter. For the clients, updating the local models is similar to reading an RSS feed.

References

1. Decker, S., Park, J., Quan, D., Sauermann, L., eds.: The Semantic Desktop – Next Generation Information Management & Collaboration Infrastructure. In Decker, S., Park, J., Quan, D., Sauermann, L., eds.: The Semantic Desktop – Next Generation Information Management & Collaboration Infrastructure, Galway, Ireland (2005)
2. Haller, H., Völkel, M., Kugel, F.: imapping wiks - 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)
3. 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)
4. Völkel, M., Oren, E.: Towards a wiki interchange format (wif). In Völkel, M., Schaffert, S., eds.: Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics. (2006)
5. Haller, H.: Mappingverfahren zur wissensorganisation (2003) `http://heikohaller.de/literatur/diplomarbeit/`.
6. Fielding, R.T.: Architectural Styles and the Design of NetworkBased Software Architectures. PhD thesis, U. Mass. (2000)
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. Bush, V.: As we may think. Atlantic Monthly **176** (1945) 101–108
11. 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)
12. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. Technical report, HP (2004)