Lightweight Linked Data

Erik Wilde and Yiming Liu
School of Information
UC Berkeley

Abstract

Much of the Web's success rests with its role in enabling information reuse and integration across various boundaries. Hyperlinked Web resources represent a rich information tapestry of content and context, instrumental in effective knowledge sharing and further knowledge development. However, the Web's simple linking model has become increasingly inadequate for effective content discovery and reuse. At the same time, rigorous but heavyweight solutions such as the Semantic Web have yet to garner critical mass in adoption. This paper analyzes the relative strengths and shortcomings of existing linked data approaches. It proposes a novel, lightweight architecture for the modeling, aggregation, retrieval, management, and sharing of contextual information for Web resources, based on established standards and designed to encourage more efficient and robust information reuse on the Web.

1 Introduction

The *Hypertext Markup Language (HTML)* [21] and the Web's success were the ultimate proof that "linked data" is immensely useful, and that, thanks to the network effect, this usefulness becomes much more apparent if the linking mechanism is simple enough to allow many people to create and use linked data. Web document authoring has proven to be simple enough that the average user need not become learned in the intricacies of Web technologies to both contribute content and provide context to existing resources.

Yet the problem with the simple linking model became more obvious as the volume of data on the Web increased. The majority of linked data on the Web is human-oriented and thus difficult to use as a foundation for machine-based processing based on understanding. Contextual meaning must be *extracted* via algorithms such as PageRank [19], based on linking structures and link anchor text. When context is available, it is often bound to the particular resource or presentation of a resource.[1]

[1]Context, in this case, could be specified as an element within a particular HTML or XML document

In this paper, we distill a set of design principles for the creation of useful and usable linked data models, based upon analysis of the current field of linked data solutions and approaches. We then describe a novel, lightweight linked data approach, *SLink*, consisting of a simple, extensible data model which conforms to these principles. SLinks are stored in *linkbases*, and an associated protocol provides well-known means of accessing and querying these linkbases. Users may choose to subscribe to one or several context-providing linkbases, depending on their information needs. Though we present a relatively simple use case — the integration of contextual information for Web pages as linkbases — the SLink model can also serve as a generalized backbone architecture enabling robust information aggregation, integration, and reuse on the Web.

2 Linked Data Approaches

The Web has given rise a number of approaches and technologies for linked data models, each with its own advantages and weaknesses. HTML and XML, the staple formats of the Web, have provided limited facilities for linking other resources to a given resource for context and reuse.

More recently, two major, divergent approaches to linked data on the Web has emerged at two opposite ends of the architecture design spectrum, and has since made it hard to search for middle ground. On the one end, the *Semantic Web* — a central focus of the W3C's activities — is a comprehensive solution for anything related to structured data, but is based on a complex foundation and a completely new model and toolset. On the other end, the developments often summarized as *Web 2.0* have emerged from the bottom up to serve as pragmatic solutions for adding machine-readable structures to the Web, but often lacking the design and coherence which is required for larger linked data scenarios.

2.1 HTML and XML

Even though it focuses on human-readable documents, HTML has mechanisms for link semantics, most notably the `link` element in the document head. HTML itself defines a number of possible `rel`ations to associated resources, and some of these are also used in a more de-facto standard way

to detect certain common resource types.[2] HTML also defines structural links between Web resources, though few Web sites and browsers expose this information natively. This is still a very simplistic linking model, expressing a one-way linkage between a given resource and some contextual resource or association. Link information is exclusively controlled by the author or publisher of the HTML document, and is bound to the specific document that contains it.

When the *Extensible Markup Language (XML)* [5] was introduced, XML's success as a format for machine-readable data on the Web quickly became apparent. Originally intended as a presentation-neutral format for human-readable documents, XML was accompanied by the *XML Linking Language (XLink)* [7], which was mainly designed from a hypermedia point of view. XLink's lack of success can be attributed to a number of reasons, likely candidates are the markup design, the lack of a well-defined data model, the lack of standardized protocols for retrieving links, the focus on hypermedia issues, and the lack of coordination with other standardization efforts.

2.2 Current approaches

The *Semantic Web* [2] proposes layers of semantic information on top of the Web. Technically, these layers have little in common with the Web apart from the fact that the concept of a *Uniform Resource Identifier (URI)* [1] is used for identifying resources. The data model of the Semantic Web is the *Resource Description Framework (RDF)* [14], which is different from the Web's tree based data models.

In terms of expressiveness, the Semantic Web model is quite powerful, so there almost never is a problem with a lack of expressiveness of Semantic Web technologies. The biggest problem in terms of practical applicability is the fact that Semantic Web applications require a complete new toolset for working with them at the model level. Established Web technologies do not work well on RDF graph structures, and storing and querying it requires RDF stores and the specialized query language *SPARQL* [20]. The search for the *killer app* that will turn the Semantic Web into technology mainstream is still ongoing.

Contrasting the top-down development of Semantic Web technologies, the *Web 2.0* landscape focuses on developing technologies based on concrete demands, and often produces ad-hoc implementations which can be incorporated in browsers through scripting or as plug-ins. One of these developments are *microformats* [13], which evolved out of the the practical need to semantically mark up structures in Web documents. HTML contains some "semantic elements,"[3] but their semantics and syntax are not well-defined, and their use is too undisciplined to be useful for automated tasks such as the extraction of address data from a Web page.

Another Web 2.0 concept is *tagging* [10]. Tagging annotates a resource (for example, a Web page or an image) with some tag, which simply is a term chosen by a user. Tagging is often described as putting Web resources into context (for one user, by just looking at his tags, and for all users, by datamining the tag space).

However, tags simply create a loose association of resources (all the resources sharing a tag), whereas links have more expressive power. Links can be *typed*, and the resources participating in a link can play *roles* in the context of a link. There can be a link which represents all resources belonging to a compound document, and another one aggregating all the home pages of students taking a course. Similarly, for a resource role within a link, there may be one resource that is playing the role of the starting page, whereas another one is playing the role of the glossary. Representing these kinds of structures in tags leads to the use of "structured tags" and ad hoc rules for using these, which basically indicates that a solution other than tags would be appropriate.

The core of the problem with most Web 2.0 developments, including these 'linking models', is similar: they are insufficiently designed and specified for efficient resource discovery, reuse, and integration. In the case of microformats, for example, a search engine such as Google must crawl the entirety of the Web to discover microformat resources scattered therein. The semantics problems surrounding tagging are well-known [15], and its selling point — the lack of structure — also constitutes its weakness. The lack of standard interfaces for Web 2.0 services and the specificity of those interfaces to particular services contributes to the fragility of models for information integration and reuse on the Web, such as within Web *mashups*.

3 The Plain Web

There is some middle ground which could support many of the scenarios requiring the association of Web resources, without introducing the complexity of an entire new layer of technologies, and without incurring the costs of unreliability and inefficiency. This approach is based on the idea of the *Plain Web* [24] and is based on simple standard Web technologies.

3.1 The Argument for Simplicity

A well-known example for the rather high price of complexity is *XML Schema* [22]. While the XML structures described by that language can be accessed using XML tools, there is no standard for how the model of a schema itself can be accessed. As a consequence, schemas are opaque

[2] `rel="alternate" type="application/atom+xml"`, for example, is usually interpreted as referring to feeds for the HTML page.

[3] Examples for these are `address`, `acronym`, or `cite`.

for Web technologies, and only few of the language's various features are used by most schemas [4]. There have been some experiments to make the XML Schema model more accessible [23], but the current revision of the language [9] adds more features to the language, instead of making it simpler and more open to the Web itself.

The 80/20 principle is quoted frequently, stating that for 20% of the cost of solving a problem, 80% of the use cases can be addressed. Both Semantic Web and XML Schema are attempting to be 100% solutions, which influences the solutions, the tools required to implement the solutions, and the skills required to use these tools. The Web's earlier technologies were much more inspired by the 80/20 model (HTML and CSS are excellent examples for that).

A W3C TAG Finding [3] states: "When designing computer systems, one is often faced with a choice between using a more or less powerful language for publishing information, for expressing constraints, or for solving some problem. [. . .] The 'Rule of Least Power' suggests choosing the least powerful language suitable for a given purpose." RDF/OWL is then recommended, based on a comparison with imperative languages. It could be argued that being declarative alone is not sufficient to satisfy the "rule of least power," and that the same rigor of choosing the least powerful language that is suitable for a given purpose should also be applied to various ways of how powerful declarative languages are.

3.2 Surface vs Deep Models

Making resources on the Web more connected is one of the big challenges on the Web. As an example, increasingly, companies see their IT infrastructures moving away from the more classical top-down centralized design, towards a structure that very much resembles the Web itself — growing organically, loosely coupled, decentralized, heterogeneous, and thus in need of linking all of this together.

Recently, *Enterprise 2.0* has been coined as a term to introduce more Web 2.0 concepts into the world of enterprise IT infrastructures. Enterprise 2.0 not only emphasizes the value of social information in the enterprise, it also is an answer to former more heavyweight approaches to knowledge management, which often did not live up to their expectations [6].

The question is how this vision should be implemented. So far, adoption of Semantic Web technologies within enterprises has been poor. One of the questions is whether such a heavyweight approach is required. Looking into the requirements of any approach involving semantics, the question is how many of the real-life scenarios require *controlled vocabularies*, *taxonomies*, *thesauri*, or *ontologies*. Following the Rule of Least Power, It would be better to only reserve such complexity to those schemes which really require, for example, the full expressiveness of an ontology.

Even if IT systems sometimes will be based on very sophisticated classification schemes, which would allow very sophisticated reasoning based on ontologies, there should be a simple format, accessible to basic Web technologies, which would at least express the surface of such a model. Users of that simplified model might not be able to explore and use the model in depth, but they would still gain access to the essential information, such as which resources are connected based on certain concepts.[4]

4 Designing Lightweight Linked Data

The idea of lightweight linked data is to represent links (the essence of linked data) in a format which is easily accessible with basic XML technologies. Such a format cannot replace sophisticated semantic frameworks such as the Semantic Web, but it can be used to represent the surface of the Semantic Web (simply ignoring its depth), and for simple scenarios not requiring the depth and complexity of a more sophisticated framework, such a link format is sufficient.

Several existing or proposed link formats are currently available as reference points, specifically *XLink* [7], *METS* [8], *DIDL* [12], and the recently published draft of the *Open Archives Initiative Object Reuse and Exchange (OAI-ORE)* [18]. While all of these formats have specific features introduced for the intended application area, it is possible to distill a "core" of structural link features:

- *Link Types:* Can links be typed, and if so, what is the classification scheme for these types?
- *Resource Types:* Can resources participating in a link be typed, and if so, what is the classification scheme for these types?
- *Resource Roles:* Can resources participating in a link play roles in the context of that link, and if so, what is the classification scheme for these types? Is it possible to play multiple roles?
- *Substructures:* Is the link a flat collection of resources, or is there a hierarchical concept? If so, is it recursive or is the number of levels limited?
- *Connections:* Is it possible to explicitly connect a link's resources? If so, what are the identifiers of a connection, resources themselves, resources types, resource roles, some combination of these, or some different identification? Can connections be typed, and if so, what is the classification scheme for these types?

[4] What would be invisible to these users would be the fact that some concept is a specialization of another one, and that because of this connection, more conclusions could be drawn from the resource associations.

- *Traversal:* Is it possible to add information to the link and/or to connections which is specific for link traversal? If so, is there any specific information that controls or augments traversal through additional annotations?

- *Conditionality:* Is it possible to have links that are configured differently based on some external context? If so, what are the link structures that can be controlled by that feature?

This core is the starting point for the design of an XML-based link format which is applicable to a variety of Web technologies. While the Semantic Web may still find its "killer app" and take off eventually, a more lightweight sibling, focusing on the very core of linked data, the connections between Web resources, might even help it, serving as an easy entry point and an accessible and usable surface view of the Semantic Web.

Looking at the lessons learned from previous link formats, the Semantic Web, and Web 2.0, the following issues should be guiding such a development of a lightweight approach to linking data on the Web:

- *Separation of Model and Syntax:* There should be an abstract model of how links are structured, and there should be a normative syntax for that model. Layered technologies should build on the model, not the syntax.

- *Based on simple Web Technologies:* It should be possible to fully process the syntax and construct the model-level structures using basic Web technologies.

- *Simplicity over Completeness:* Being explicitly designed as a lightweight format, the 80/20 rule should be used to eliminate all features which are not considered essential.

- *Extensibility:* For applications requiring more than just the basic features supported by the format itself, there have to be clear extensibility guidelines for a well-defined interpretation of all links.

In conjunction with such a lightweight format, there also should be a protocol for interacting with this data if it is managed as standalone data. While the detailed description of this protocol is beyond the scope of this paper, we believe that Atom [16] and the *Atom Publishing Protocol (AtomPub)* [11] are ideal candidates for such a protocol for interacting with collections of lightweight links.

5 Simple Links

Based on the design considerations described in Section 4, we have defined a lightweight model for linked data on the Web. Its data model is described in Section 5.1, and the protocol for accessing collections of links (which we call *linkbases*) is described in Section 5.2. Since one main design goal is to create a lightweight model that is accessible by standard XML technologies, the links are called *Simple Links (SLinks)*, and the access protocol is called the *Simple Link Access Protocol (SLAP)*. Section 5.3 describes the processing model for clients, briefly outlining how clients can use SLink data and the access protocol to retrieve context information for Web resources.

5.1 Simple Link Model

Our link data model is simple, and even takes simplicity a step further than XLink. The important difference to XLink is that our link model also supports *link schemas* and *link views*. A link schema is a set of constraints that can be used to describe a collection of links. A link view is set of navigable paths between link resources which can be applied to a link collection described by a link view. There can be more than one view for a collection.

```
<slink type="person" slink-namespace="...">
 <resource href="tag:..." role="identity"/>
 <resource href="http://..." role="homepage"/>
 <resource href="http://..." role="school"/>
 <resource href="http://..." role="course"/>
 <resource href="http://..." role="course"/>
</slink>
```

Figure 1. SLink Example

A *Simple Link (SLink)* (shown in Figure 1) identifies a set of resources by URI which are participating in the link. The link can be identified by a `type`, which may be described and associated with a link schema. Each resource participating in the link may have a `role`, which describe the role that the resource plays in the context of that link.

As a special case, a resource may be an SLink, in which case the SLink in effect becomes a nested structure. There are no special mechanisms in the data model to support nested links, but the data model allows them,[5] and the processing model (described in Section 5.3) describes how SLinks must be processed to correctly handle nested links. Because nested links make processing harder, linkbase providers can make explicit whether links might be nested or not. This can be done in a link schema, which generally describes constraints for a collection of links. Figure 2 shows a schema for the link shown in Figure 1, which defines the cardinality constraints for role occurrences within a link (default values for minimum and maximum are `0` and `unbounded`).

An *SLink Schema* describes constraints for one or more link types. Constraints can be defined in a variety of ways,

[5]Nested links can simply be pointed to as resources, optionally the `resource` element may contain an attribute signalling that a resource is a nested link.

```
<slinks slink-namespace="...">
 <slink type="person">
  <role name="identity" minOccurs="1"/>
  <role name="homepage" maxOccurs="3"/>
  <role name="school" minOccurs="1"/>
  <role name="course" minOccurs="0"/>
 </slink>
</slinks>
```

Figure 2. SLink Schema

most importantly in terms of role occurrences and nested links. In addition, a schema can specify that the schema is `open` or `closed`, which allows or disallows the occurrence of other roles than the ones specified in the schema. In addition to defining constraints, a schema can also define `arcs`, which represent navigable paths between resources. Since different users of a linkbase may have different preferences regarding arcs, the linkbase schema does not have to specify arcs (these only act as the default set of arcs unless overwritten by a user), and users may have their own set of arcs, which are defined in an *SLink View*, which essentially is a schema without any role constraints.

```
<slinks slink-namespace="...">
 <slink type="person">
  <arc from="homepage" to="department"/>
  <arc from="homepage" to="course"/>
  <arc from="course" to="hompeage"/>
 </slink>
</slink>
```

Figure 3. SLink View

All identifiers (link types and resource roles) are defined as QNames, which means that SLink schemas can either define values which are in no namespace, or can define values which are identified by their namespace name and the name within that namespace. Namespaces allows users to agree on vocabularies of link types and roles and thus to identify shared semantics.

5.2 Simple Link Protocol

Interaction with linkbases is based on AtomPub. Linkbases are special AtomPub collections which accept SLinks as resources. In addition, the linkbase's service document contains a `link` with a special relationship of `schema` which identifies the schema for the linkbase (if there is one). The schema for a linkbase allows clients to discover which types of links to expect when querying the linkbase, and which types of links they may submit to the linkbase. Essentially, linkbases in this scenario are AtomPub collections which contain SLinks and implement additional features, such as schema discovery and queries.

A linkbase may contain a large set of entries, and Atom's usual model of accessing feeds and retrieving a time-ordered sequence of entries is not very useful. Thus, the protocol for accessing linkbases supports queries based on four simple parameters: The URI for which to find SLinks, the role(s) in which this URI is being used, and the maximum number of results that should be returned. All of these parameters are used as URI query parameters, so a typical URI query string in a request to a linkbase might look like this:

```
?uri=...&type=...&role=...&results=99
```

`uri` specifies the URI for which to return SLinks, and the two role parameters specify that SLinks should only be returned if the URI occurs in one of the two roles. Based on this model, clients can limit the responses to certain link `type`s, and to only those links which provide navigable paths from the current resource (if no `role` is specified, no such filtering is done). While we are currently using an more traditional syntax, the recently proposed *Feed Item Query Language (FIQL)* [17] may provide an alternative (and feed-specific) syntax for our queries.

The result of such a request is an Atom feed where each entry has an SLink as its content. Ordering is done by relevance (not by timestamp, as is usually the case for Atom feeds), and is decided on by the server.

5.3 Processing Model

Clients typically request links from linkbases when they are processing some resource. For example, a browser supporting linkbase access displays a Web page, and accesses linkbase(s) for additional context. For the examples shown in Section 5.1, this might mean that a user looks at the homepage of a person, and the browser queries the linkbase about SLinks in which that URI occurs. If the URI occurs in an SLink, the browser receives the SLink about that person's connections to the school or courses, and makes them available as navigable information.

Figure 4 shows the data structure that a linkbase client might encounter when establishing the context for a given resource. Context might be retrieved from multiple linkbases, each linkbase might provide various link types, for each of the link types there might exist various links, and the resource might participate in each of these links in various roles. If the linkbase schema permits nested links, it is also possible that links refer to links (not shown in the figure). The processing model defines what clients need to do to construct a complete model of the link information that is available for a given resource.

6 Implementation

Interaction with linkbases is based on Atom and AtomPub. Atom is used for publishing SLinks as content of feed entries, and AtomPub is used to allow clients to submit new

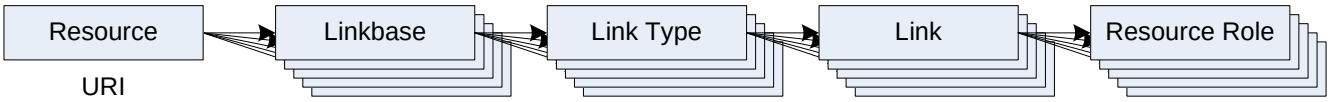

Figure 4. Establishing the Context for a URI-identified Resource

links to linkbases. We extend AtomPub's service document with a new `link` relationship, which allows clients to discover the `schema` of a linkbase. This allows clients to provide interfaces for creating links which are using the schema as a skeleton for valid link structures.

Our current implementation is using an XML database, because for experimenting with link models and different possible features for link queries, it is much easier to work on XML data and use XQuery, than it is to work with a more rigidly defined relational database. Of course, the protocol for interacting with the linkbase does not depend on a specific technology, but it does require that XML Namespaces are handled properly (because of the QNames of link types and roles), which can be cumbersome when working in non-XML environments.

7 Conclusions

This paper looks at the currently available two extremes of the "linked data" design spectrum: On the one side the Semantic Web, which defines a very powerful by rather complicated set of technologies, and requires users to switch to a complete new toolset for working with linked data. On the other side the Web 2.0 approach which is very much bottom-up and implements any kind of "linked data" on an as-needed basis, making it harder for linked data to reach critical mass because of fragmented formats and ways of handling data.

This paper proposes to seek the middle ground, which is still using basic Web technologies, but aims at representing the essence of linked data, which is the association of resources in a link. This lightweight link can be easily processed using standard Web technologies such as XSLT or XQuery, and as such is really part of the Web itself, and not of the higher level layers of the Semantic Web. Nonetheless, such a link could also be used as the surface of a deeper Semantic Web structure, exposing the most basic information about link associations, but still allowing Semantic Web users to explore the richer information available in the higher layers.

References

[1] TIM BERNERS-LEE, ROY THOMAS FIELDING, and LARRY MASINTER. Uniform Resource Identifier (URI): Generic Syntax. Internet RFC 3986, January 2005.

[2] TIM BERNERS-LEE, JAMES A. HENDLER, and ORA LASSILA. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[3] TIM BERNERS-LEE and NOAH MENDELSOHN. The Rule of Least Power. World Wide Web Consortium, TAG Finding, February 2006.

[4] GEERT JAN BEX, WIM MARTENS, FRANK NEVEN, and THOMAS SCHWENTICK. Expressiveness of XSDs: From Practice to Theory, There and Back Again. In *Proceedings of the 14th International World Wide Web Conference*, pages 712–721, Chiba, Japan, May 2005. ACM Press.

[5] TIM BRAY, JEAN PAOLI, C. MICHAEL SPERBERG-MCQUEEN, EVE MALER, and FRANÇOIS YERGEAU. Extensible Markup Language (XML) 1.0 (Fourth Edition). World Wide Web Consortium, Recommendation REC-xml-20060816, August 2006.

[6] MU-YEN CHEN and AN-PIN CHEN. Knowledge Management Performance Evaluation: A Decade Review from 1995 to 2004. *Journal of Information Science*, 32(1):17–38, 2006.

[7] STEVEN J. DEROSE, EVE MALER, and DAVID ORCHARD. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[8] DIGITAL LIBRARY FEDERATION. Metadata Encoding and Transmission Standard: Primer and Reference Manual, September 2007.

[9] SHUDI GAO, C. MICHAEL SPERBERG-MCQUEEN, HENRY S. THOMPSON, NOAH MENDELSOHN, DAVID BEECH, and MURRAY MALONEY. W3C XML Schema Definition Language (XSDL) 1.1 Part 1: Structures. World Wide Web Consortium, Working Draft WD-xmlschema11-1-20070830, August 2007.

[10] SCOTT GOLDER and BERNARDO A. HUBERMAN. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, 2006.

[11] JOE GREGORIO and BILL DE HÓRA. The Atom Publishing Protocol. Internet RFC 5023, October 2007.

[12] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Information Technology — Multimedia Framework (MPEG-21) — Part 2: Digital Item Declaration. ISO/IEC 21000-2:2005, October 2005.

[13] ROHIT KHARE and TANTEK ÇELIK. Microformats: A Pragmatic Path to the Semantic Web. In *Poster Proceedings of the 15th International World Wide Web Conference*, Edinburgh, UK, May 2006. ACM Press.

[14] GRAHAM KLYNE and JEREMY J. CARROLL. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[15] ADAM MATHES. Folksonomies — Cooperative Classification and Communication Through Shared Metadata. Technical report, University of Illinois, Urbana-Champaign, Illinois, December 2004.

[16] MARK NOTTINGHAM and ROBERT SAYRE. The Atom Syndication Format. Internet RFC 4287, December 2005.

[17] MARK NOTTINGHAM. FIQL: The Feed Item Query Language. Internet Draft draft-nottingham-atompub-fiql-00, December 2007.

[18] OPEN ARCHIVES INITIATIVE. ORE Specification — Abstract Data Model, December 2007.

[19] LAWRENCE PAGE, SERGEY BRIN, RAJEEV MOTWANI, and TERRY WINOGRAD. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report SIDL-WP-1999-0120, Stanford University, November 1999.

[20] ERIC PRUD'HOMMEAUX and ANDY SEABORNE. SPARQL Query Language for RDF. World Wide Web Consortium, Recommendation REC-rdf-sparql-query-20080115, January 2008.

[21] DAVE RAGGETT, ARNAUD LE HORS, and IAN JACOBS. HTML 4.01 Specification. World Wide Web Consortium, Recommendation REC-html401-19991224, December 1999.

[22] HENRY S. THOMPSON, DAVID BEECH, MURRAY MALONEY, and NOAH MENDELSOHN. XML Schema Part 1: Structures Second Edition. World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, October 2004.

[23] ERIK WILDE and FELIX MICHEL. SPath: A Path Language for XML Schema. In *Poster Proceedings of the 16th International World Wide Web Conference*, pages 1343–1344, Banff, Alberta, May 2007. ACM Press.

[24] ERIK WILDE. The Plain Web. In *Proceedings of the First International Workshop on Understanding Web Evolution (WebEvolve2008)*, pages 79–83, Beijing, China, April 2008.