The Plain Web

Erik Wilde
School of Information
UC Berkeley
dret@berkeley.edu

ABSTRACT

The Web has become a popular starting point for many innovations targeting infrastructure, services, and applications. One of the challenges of today's vast Web landscape is to monitor ongoing developments, put them into context, and assess their chances of success. One of the main virtues of a more scientific approach towards the Web landscape would be a clear differentiation between approaches which build on top of the infrastructure of the Web, with little embedding into the landscape itself, and those that are intended to blend into the Web, becoming a part of the Web itself. One of the main challenges in this area is to understand and classify new developments, and a better understanding of various dimensions of Web technology design would make it easier to assess the chances of success of any given development. This paper presents a preliminary classification, and presents arguments how those factors influence the chance for success.

1. INTRODUCTION

The Web's success has been analyzed in a variety of ways, and most likely cannot be attributed to one single cause. The rising availability of the Internet, declining computer costs, the demand to share information in platform-independent ways, and the simple technologies based on loose coupling were all contributing factors. Many of these factors are outside of the "engineering realm" of the Web, the Web just "happened to be there at the right time." And yet, the Web's simplicity, its roots in the question of how to solve pragmatic information sharing issues, and the design that was the very opposite of the heavyweight approaches that were proposed for similar application areas from the communications systems [29] and hypermedia [26] communities, were all critically contributing factors to its success.

This paper argues that the Web's simplicity still is its most important asset, and that recent developments in Web technologies as well as future directions should take this into consideration. Specifically, this paper identifies as two important approaches in Web technology development the approaches of *revolution* vs. *evolution*. The revolutionary approach often simply builds on the Web as an existing and convenient transport infrastructure, placing an entirely new layer of functionality on it. In contrast, the evolutionary approach attempts to improve existing Web technologies incrementally, often at the price of having to deal with backwards compatibility and interoperability in a loosely coupled system.

In discussions about future directions for the Web, both approaches are adamantly defended with many arguments, and the debate around these approaches often is heated and controversial. One of the main reasons for this is that any development in a technological landscape as large and as economically significant as the Web not only has technical implications and consequences, but also involves many commercial interests. Furthermore, given the amount of research funding that goes into various areas associated with the Web, money and careers are at stake. This paper is not an approach to look into this wider field of non-technical dependencies of Web technology development, but any inquiry of Web technology development should take these factors into account.

This papers focuses on the contrast of the revolutionary (Section 2) vs. the evolutionary (Section 3) approach of advancing the Web. For each of these approaches, some examples are presented which illustrate typical representatives. Following up with a case study (Section 4), the paper then makes a case for the evolutionary approach called the *Plain Web* (Section 5). Evolving the Web in a far-sighted and cleanly engineered way should be preferred over solutions which build entire new technology landscapes that do not integrate well into the Web and are only accessible through entirely new toolsets. Based on this approach of limiting the Web to evolutionary developments, Section 5 then concludes the paper with a brief sketch of possible research directions, which would help to better understand how Web development is most likely to avoid costly errors.

2. WEB REVOLUTION

The Web's success has clearly shown that it filled a formerly unoccupied gap in the communications landscape, but its simplicity also often is perceived as a deficiency, rather than a property that might have been and still is a contributing factor to its success. The limitations of the Web's basic features in terms of content design and semantics, and its ability to serve as a platform for machine-to-machine interactions, have led to numerous developments to extend the Web's capabilities.

The following sections provide a more detailed look at three fields where revolutionary attempts were proposed to "improve the Web." A common theme of these revolutionary approaches is that they use the Web mainly as a transport system. So instead of trying to integrate into the existing way of how something is done the Web, they introduce something entirely new that is supposed to completely take over that area.

Another commonality for most of these approaches is a fairly monolithic solution. A large problem area is identified, and then some solution is created which intends to be general and powerful enough to solve problems in that area.

Web Science Workshop at WWW2008, April 22, 2008, Beijing, China.
.

Because the problem area tends to be large and complex, the presented solution also tends to be large and complex.

2.1 Content

One of the early complaints about the Web was the simplicity of its main content types. Today's combination of HTML, CSS, and advanced scripting[1] allow rather sophisticated Web design, but earlier versions of these technologies and earlier browsers were limiting Web publishing to a basic set of features, looking to many like a return to the days of character-based terminals.

In particular, when it comes to media types' variety and features, Web technologies and browsers provide a rather small set of options. And from the perspective of *Web designers*, who often only get in touch with Web technologies through tools, there still exists a rather popular idea that the basic Web technologies are just a starting point, and that more sophisticated, multimedia-oriented technologies are the logical next step of Web development. This world view often manifests itself in using entire sites in *Flash* or other technologies which essentially turn the Web into a transport infrastructure for a proprietary media player.

Two activities that were initiated by the W3C in the area of richer media types were the *Synchronized Multimedia Integration Language (SMIL)* [17] and the *Scalable Vector Graphics (SVG)* [13]. Both technologies did not succeed as a universally used content type on the Web, and as always, there is no single reason for that. But one important factor definitely was the development of these formats as standalone formats, with little connections to the existing infrastructure. To this day, there are various ways of embedding SVG into HTML, and each of these ways has its own peculiarities and side-effects. SMIL, on the other hand, created a sophisticated language for synchronizing multimedia presentations, but some of the center pieces of those presentations, audio and video content, were never standardized, and today's de facto standard for playing these media types is to use proprietary media players.

So instead of slowly developing richer content from the ground up, these technologies were developed with little connections to the existing fabric of the Web, and ultimately failed.

Another important mechanism for revolutionizing content on the Web are proprietary runtime environments, which support and use a lot of Web technologies, but are not browsers. These are increasingly referred to as *Rich Internet Application (RIA)*, a term which initially was used for browser-based content using advanced Web technologies such as scripting and Ajax (described in Section 3.1). The proprietary RIAs also represent revolutionary approaches, arguing that they allow developers a richer environment for developing applications,[2] and that they support features which are not available in browser-based applications.[3]

Another area of content-related revolution is the area of schema languages. XML Schema [32] introduced a powerful and complex schema language with its own data modeling layer. While the XML structures described by that language can be accessed using basic XML tools, there is no standard for how the complex model of a schema itself can be accessed. As a consequence, schemas are opaque for Web technologies, and only few of the language's various features are very much used [5]. There have been some experiments to make the XML Schema model more accessible [34], but the current revision of the language [15] adds more features to the language, a step in the opposite direction.

2.2 Semantic Web

The *Semantic Web* [4] aims to turn the Web into a machine-understandable Web. It introduces a model for making statements, the *Resource Description Framework (RDF)* [21], and layers on top of this languages (such as *RDF Schema* [8] or *Web Ontology Language (OWL)* [31]) which can be used to define and constrain the concepts which are used in these statements.

The Semantic Web builds a number of layers on top of the Web infrastructure, and most of the technologies are entirely new. The only major connection between the Web itself and the Semantic Web is the *Uniform Resource Identifier (URI)* [3], which is used for the identification of entities in both architectures. Even for this crucial link between the Web and the Semantic Web, there so far is little actual connectivity; the assignment, management, publishing, and discovery of URIs rarely work outside of closed environments. As an example, many basic RDF examples still assume the ontological equivalence of persons and their home page or email, which points to the basic problem of how identity is handled.

Despite predictions of the ultimate success for the Semantic Web [30], so far there is little use of it outside of the (arguably large) communities of academia and commercial players having direct investments in the technology. This does not necessarily mean that there will not be eventual success; but regardless of that ultimate outcome, one could at least ask why the technology has taken so much longer to catch on and gain momentum than the much more limited XML. The fact that any work in the Semantic Web realm requires a complete new toolset most likely has made it less likely for potential users to start using it, and better support for a more gradual transition probably would have helped to increase the adoption rate.

2.3 Web Services

Originally, the term "Web Service" only referred to services described by an interface described in the *Web Services Description Language (WSDL)* [9], which are invoked using messages based on *SOAP* [27]. Repeating the pattern of the above examples, this model of a Web service reduced the Web to a transport infrastructure for XML-encoded message exchanges. Subsequently, additional standards were layered on top of this model, in many cases replicating functionality of a basic networking protocol stack [19].

Web services were attractive for middleware providers, because they allowed them to simply extend their existing

[1] All of these technologies are considered in more detail in Section 3.1.

[2] It is interesting to note that this notion entirely depends on the definition of *rich*. If *rich* means a better development environment and more functionality in the runtime environment, this is true. If *rich* means fully embedded into the fabric of the Web and able to access and be accessed in that environment, it is not true.

[3] A popular example for this is the ability to work *offline*, which is more or less non-existent in browsers (though frameworks such as *Google Gears* are beginning to change this). But it is inevitably true that the Web as a runtime environment is limited, but as this paper argues, there usually is a high price associated with creating something entirely new to escape these limitations.

software products with new stub-generating software. Customers using these updated software suites would then be able to use Web services without changing anything in their existing systems.

Web services in the WSDL/SOAP flavor require an entirely new toolset, and while they do employ some Web technologies besides the simple transport service, the synergies between the traditional Web and WSDL/SOAP Web services are slim. In a way comparable to the Semantic Web, within certain user communities, there is a real benefit to this technology, but this benefit often does not extend beyond that community, because there is no graceful path leading to the new technology.

3. WEB EVOLUTION

The important common theme for all the examples discussed in the previous section is that they assume that users will fully adopt the new technology in an "all or none" fashion, with no middle ground for easier migration. This section contrasts this philosophy with a more evolutionary approach, where the design of the new technology is informed and influenced by assumptions about backwards compatibility and the ability to migrate to the new technology with a graceful path to a new toolset.

3.1 Content

One of the biggest success stories of Web technologies is the *Extensible Markup Language (XML)* [7], which originally was intended to be used for Web content [28], but has become a universal language for structured data. One of the big influential factors was XML's simple tree model and the ability to work with that through the *Document Object Model (DOM)* [22]. These were both well-known to HTML users as an established API at the time XML was invented. XML's lack of semantics make it a much less ambitious approach than the Semantic Web, but the adoption rate clearly demonstrated that such a simple approach can be successful.

For Web content intended for humans, the big developments were not so much newer and richer datatypes, but instead the development of the browser as a application development platform. Better DOM conformance and the `XMLHttpRequest` [33] method of enabling scripts to communicate with a server (often referred to as *Ajax*) were the main factors of that development. The ability to gradually upgrade a Web page from static HTML to a more dynamic version provided content providers with a graceful path to a more modern Web presence.

With regard to the core standards, the development of *HTML 5* [16][4] also is a evolutionary way of improving HTML. HTML's notoriously weak structuring and form mechanisms are improved in HTML 5, so that many things which today are implemented by scripting libraries will be supported by the browser itself. This way, HTML is being upgraded in a way which reflects current practices, thereby reducing the need for scripting and improving HTML as a declarative language with features grounded in existing practice.

Another area of evolutionary changes in the area of *Cascading Style Sheets (CSS)* [25], a language which has been iteratively developed over the last decade. CSS's features have progressed from a basic vocabulary intended to replace HTML's formatting, to a sophisticated language for designing two-dimensional layouts. While the current developments mainly focus on scrollable screen and print media, HTML has evolved to become a decent language for describing paged, reflowable, and interactive content, and the current *Advanced Layout* [6] and *Paged Media* [24, 23] drafts are pointing into that direction. Once these mechanisms are extended to cover not only printed media (which currently is their main application area), but interactive paged media as well, HTML could become the standard language for e-book content, well-poised to unify a currently highly fragmented market.

Regarding the question of schema languages briefly mentioned in Section 2.1, evolutionary approaches include a modular approach to schema modeling (using lightweight languages for specialized tasks), and simpler schema languages such as *Schematron* [18], which can be implemented using existing Web technologies.

3.2 Semantic Web

The need for more semantics on the Web is something almost everybody can agree on, and the evolutionary alternative to the Semantic Web initiative is the bottom-up development happening in the area of *microformats* [20]. While these formats have no common foundation and no backing framework, they address concrete needs of existing user communities. Microformats evolve out of the need to have a semantic representation for some (most often narrow) set of semantics, and they use various hooks in HTML to reuse some of HTML's weak semantics, and extend them with a more precisely defined way of how to encode data.

The biggest disadvantage of microformats is that there is no common framework, so different microformats often use different approaches for representing their data. This makes it hard to combine them, unless a Web page is designed in advance to support all the different representations. The recently introduced *RDF in Attributes (RDFa)* [1] syntax addresses some common microformat shortcomings and builds a promising bridge between the more formal world of the Semantic Web as introduced in Section 2.2, and the lightweight world of microformats.

Another development in that area is *Gleaning Resource Descriptions from Dialects of Languages (GRDDL)* [10], it defines a method to extract RDF descriptions from any kind of recognizable structure in a Web page. While GRDDL layers a unifying method on top of the variety of ways how semantics can be encoded in Web pages, it does not expose this information in a lightweight format, but instead directly as RDF. The general approach of GRDDL, however, is an interesting one, because it allows any reasonably well-defined structure to be transformed into a unified representation, using simple transformations. While currently GRDDL is defined to directly produce RDF, it could be easily adapted to produce something that is both well-defined and lightweight, such as RDFa.

3.3 Web Services

With the emergence of more advanced and interactive content (as described in Section 3.1), and in particular with the advent of Ajax, the ability to access data on a server became an important part of Web applications. For this scenario, the Web Service technologies described in Section 2.3 are rarely used, because the overhead of the envelope format is

[4]With that development, the more ambitious *XHTML 2.0* [2] and *XForms* [12] approaches to re-create content and forms will probably not gain widespread adoption.

considered a disadvantage. While Ajax has been named because applications are using XML for dynamic content, an even simpler technology is gaining increasing support in this area: *JavaScript Object Notation (JSON)* [11], which allows scripting code to directly interpret received data.

Outside of the Ajax application area, there also is a competitor for the more heavyweight Web Services approaches. The concept of *Representational State Transfer (REST)* [14] has become a popular way of building Web services. REST is based on the idea that Web services should reuse existing components of Web architecture (such as protocol semantics), rather than building additional layers on top of it.

The debate around the more heavyweight Web Services approach described in Section 2.3 and the more lightweight RESTful approach is still active. For a discussion of the evolution of the Web and Web science, it is interesting to note, however, that most arguments in favor of the heavyweight approach are based on more tightly coupled scenarios. Even the most ardent REST advocates would rarely claim that REST is the best way to build every distributed system — it just is the more appropriate choice for loosely coupled scenarios.

4. CASE STUDY: URI SCHEMES

This section presents an example in which the perspectives described in Sections 2.1 and 3.1 lead to different views of the further development of the Web. The question of whether URIs have semantics or not has been discussed extensively. In a more classic view, URIs have some semantics because the URI scheme and maybe the URI itself can be used to "understand" the nature of the resource. On the Semantic Web, such a weak way of representing semantics is not necessary and in fact would collide with the goal to represent semantics using RDF.

From the Semantic Web perspective, a URI is just a name. For proponents of the Semantic Web, new URI schemes are not necessary and not even desirable, because semantics of resources should not be reflected in their URIs, but in RDF statements about them. So here the goals of the plain Web and the Semantic Web conflict, and the question which of the two "perspectives" of URIs should be given preference is an important decision. The W3C TAG recommends that a Semantic Web compliant servers use a special HTTP response (the `303` code) to indicate that a resource may not be an HTTP resource.

This design decision makes it hard for simpler semantics to emerge. IF RDF is exclusively used to represent semantics, and an `303` HTTP code signals that additional semantics about an HTTP resource might be available, there is no way for simpler Web applications to use URI schemes with non-HTTP semantics to identify and act on resources which are not HTTP resources.

While the argument presented here is simplified for this paper, the important thing is to point out that in that case, the perspective of the Web's assumed development shapes the decisions of Web developments today. It is of course still open to debate whether new URI schemes should be deployed or not, but in the interest of an open discussion it might be interesting to clearly identify the underlying assumptions of policies and guidelines.

5. WEB SCIENCE

This paper essentially is a plea for the *Plain Web*, which we conceive as a Web that evolves gradually and with attention to backwards compatibility and ease of transition. This should of course not prevent new developments on the Web, but whenever something is planned to become *a part of the Web*, rather than *an application of the Web*, such a perspective of the Web should be considered important. One of the important tasks of *Web Science* should be to develop a framework to distinguish between evolution *of* the Web and *on* the Web.

From this perspective, one of the main challenges of Web science is that it is a lot about *not* doing things, or doing things in a less perfect or complex way. This is one of the challenges of the development of Web technologies in general: Many developments are not challenging enough from the point of view of computer science, so they are not tackled by the computer science community. Development of Web technologies is often regarded as engineering, not as science. This view has a lot of validity, but ignores the fact that good engineering also needs grounding in principles and methods, and so far little work is done in the area of "how to build systems that are as simple as possible."

For a more scientific approach towards the development of Web technologies, three of the most important design axes can be identified. As in all good engineering, design is a tradeoff between costs, constraints, and requirements, and the goal of a science of the Web should be to work towards evaluation criteria and best practices for the following areas:

- *Complexity ↔ Simplicity:* Using the popular 80/20 principle, a design should trade completeness and complexity for simplicity. Simplicity is the single most important feature of all successful Web technologies, providing the ability to use simple or even no tools to start using a new technology.

- *Cleanliness ↔ Backwards Compatibility:* While new solutions often can be designed in a cleaner and more elegant way, supporting backwards compatibility is important. Not only does it support migration on a technical level, it also makes it easier for developers and users to move towards a new technology.

- *Specificity ↔ Reuse:* Building solutions for specific problems is easier, because the problem is better defined. The Web, however, should be as unspecific as possible, which means that any hidden assumptions in a proposed solution should be made clear, so that this dimension can be better understood.

These items are only rough descriptions of a set of design axes for a science for the Web. The important observation is that many developments in the last few years have not made the design choices among these axes clear. A common scenario is that some proposed solution starts with a single sentence such as "if everybody on the Web started using technology X, then ...", and this assumption is a rather strong one.

The Web's simplicity, and the ways in which the most fundamental technologies interact, certainly is not as impressive in its complexity as the core sciences. But designing for this simplicity, and seeing this simplicity not as just a starting point, but as a virtue in itself, has its own interesting challenges. These might (and probably should) not be the same as those of the core sciences, and the most rewarding task of

a science of the Web from the technical perspective would be to embrace simplicity, not to try to overcome it.

6. CONCLUSIONS

This paper argues for a science of the Web which should emphasize evolutionary development and simplicity over revolutionary approaches and complexity. While the term *Web Engineering* never was used to describe "how to engineer the Web", but instead refers to"engineering for Web-oriented information systems," the most challenging and rewarding task of a science of a Web from the technical perspective would be to develop *Web Engineering* in its core sense — how to engineer *the Web* and not *for the Web* — into a well-described and well-defined discipline. Many Web technologies are driven by community development and see surprisingly little involvement from the academic side, which partly can be attributed to the fact that this core *Web Engineering* often is not regarded as being "real science."

In order to better understand and control the Web's future development, this "new Web engineering" should provide a framework for various design axes of Web technologies, against which new and ongoing development could be measured and benchmarked. A system as big and widely used as the Web will always have a large share of trial-and-error development, but the development of a more systematic evaluation and development of Web technologies would be likely to reduce the overall failure rate.

7. REFERENCES

[1] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing — A Collection of Attributes and Processing Rules for Extending XHTML to Support RDF. World Wide Web Consortium, Working Draft WD-rdfa-syntax-20080221, February 2008.

[2] Jonny Axelsson, Mark Birbeck, Micah Dubinko, Beth Epperson, Masayasu Ishikawa, Shane McCarron, Ann Navarro, and Steven Pemberton. XHTML 2.0. World Wide Web Consortium, Working Draft WD-xhtml2-20060726, July 2006.

[3] Tim Berners-Lee, Roy Thomas Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic Syntax. Internet RFC 3986, January 2005.

[4] Tim Berners-Lee, James A. Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[5] 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.

[6] Bert Bos. CSS Advanced Layout Module. World Wide Web Consortium, Working Draft WD-css3-layout-20070809, August 2007.

[7] 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.

[8] Dan Brickley and Ramanathan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. World Wide Web Consortium, Recommendation REC-rdf-schema-20040210, February 2004.

[9] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. World Wide Web Consortium, Recommendation REC-wsdl20-20070626, June 2007.

[10] Dan Connolly. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). World Wide Web Consortium, Recommendation REC-grddl-20070911, September 2007.

[11] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). Internet RFC 4627, July 2006.

[12] Micah Dubinko, Leigh L. Klotz, Roland Merrick, and T. V. Raman. XForms 1.0. World Wide Web Consortium, Recommendation REC-xforms-20031014, October 2003.

[13] Jon Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. World Wide Web Consortium, Recommendation REC-SVG-20010904, September 2001.

[14] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.

[15] 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.

[16] Ian Hickson and David Hyatt. HTML 5 — A Vocabulary and Associated APIs for HTML and XHTML. World Wide Web Consortium, Working Draft WD-html5-20080122, January 2008.

[17] Philipp Hoschka. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. World Wide Web Consortium, Recommendation REC-smil-19980615, June 1998.

[18] International Organization for Standardization. Information Technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based Validation — Schematron. ISO/IEC 19757-3, April 2006.

[19] Mario Jeckle and Erik Wilde. Identical Principles, Higher Layers: Modeling Web Services as Protocol Stack. In *Proceedings of XML Europe 2004*, Amsterdam, Netherlands, April 2004.

[20] 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.

[21] 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.

[22] Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Thomas Nicol, Jonathan Robie, Mike Champion, and Steven Byrne. Document Object Model (DOM) Level 3 Core Specification. World Wide Web Consortium, Recommendation REC-DOM-Level-3-Core-20040407, April 2004.

[23] Håkon Wium Lie. CSS3 Module: Generated Content for Paged Media. World Wide Web Consortium, Working Draft WD-css3-gcpm-20070504, May 2007.

[24] Håkon Wium Lie and Melinda Grant. CSS3 Module: Paged Media. World Wide Web Consortium, Working Draft WD-css3-page-20061010, October 2006.

[25] Eric A. Meyer and Bert Bos. CSS3 Introduction. World Wide Web Consortium, Working Draft WD-css3-roadmap-20010523, May 2001.

[26] David E. Millard and Martin Ross. Web 2.0: Hypertext by Any Other Name? In Uffe Kock Wiil, Peter J. Nürnberg, and Jessica Rubart, editors, *Proceedings of the Seventeenth ACM Conference on Hypertext and Hypermedia*, pages 27–30, Odense, Denmark, August 2006. ACM Press.

[27] Nilo Mitra and Yves Lafon. SOAP Version 1.2 Part 0: Primer (Second Edition). World Wide Web Consortium, Recommendation REC-soap12-part0-20070427, April 2007.

[28] Yuri Rubinsky and Murray Maloney. *SGML on the Web: Small Steps Beyond HTML*. Prentice-Hall, Upper Saddle River, New Jersey, February 1997.

[29] Andrew L. Russell. 'Rough Consensus and Running Code' and the Internet-OSI Standards War. *IEEE Annals of the History of Computing*, 28(3):48–61, 2006.

[30] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101, March 2006.

[31] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. OWL Web Ontology Language Guide. World Wide Web Consortium, Recommendation REC-owl-guide-20040210, February 2004.

[32] 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.

[33] Anne van Kesteren. The XMLHttpRequest Object. World Wide Web Consortium, Working Draft WD-XMLHttpRequest-20071026, October 2007.

[34] 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.