The Rise of the Web for Agents

Ruben Verborgh, Erik Mannens, Rik Van de Walle
iMinds – Multimedia Lab – Ghent University
Gaston Crommenlaan 8 bus 201
B-9050 Ledeberg-Ghent, Belgium
{ruben.verborgh, erik.mannens, rik.vandewalle}@ugent.be

***Abstract*—Autonomous intelligent agents are advanced pieces of software that can consume Web data and services without being preprogrammed for a specific domain. In this paper, we look at the current state of the Web for agents and illustrate how the current diversity in formats and differences between static data and dynamic services limit the possibilities of such agents. We then explain how solutions that strive to provide a united interface to static and dynamic resources provide an answer to this problem. The relevance of current developments in research on semantic descriptions is highlighted. At every point in the discussion, we connect the technology to its impact on communication. Finally, we argue that a strong cooperation between resource providers and developers will be necessary to make the Web for agents emerge.**

Keywords-Software agents; Semantic Web; Web services

I. Introduction: the Web for agents

A. Imagining intelligent agents

Artificial intelligence has always been a dream of mankind [1], and the Web is bringing new opportunities to create automated, intelligent behavior. After all, the World Wide Web is arguably the largest collective work of knowledge ever produced. A significant amount thereof is publicly available, giving any human access to massive amounts of information. Any *human* indeed, but what about machines?

While today, the Web houses many search engines [2], their functionality largely comes down to *keyword search*: to answer a certain question, a search engine can help us find documents with related keywords, but we have to read and interpret those documents ourselves. Although very convenient, it does not mark the endpoint of our desires: it would be so much easier if a machine could directly answer our question.

It is not hard to imagine an *intelligent agent* [3] that looks up facts online—and we could also contemplate on all sorts of other tasks such an agent might take over from us: order groceries, plan a holiday, submit tax returns, *etc.* However easy it is to imagine, it turns out very hard to implement such a universal agent. With the release of Apple's Siri on the iPhone, we have seen a glimpse of the potential of agent technology, but even Siri is still programmed specifically for certain domains [4]. A truly universal agent must be able to use the Web to do things it has not been preprogrammed for.

In this paper, we investigate exactly how far away we are from such a universal intelligent agent. We take a look at current possibilities for machine agents, and identify missing links that need to be resolved before we all can witness the rise of the Web for agents.

B. The Web's role in communication history

To understand the significance and impact of the Web for agents, we have to view it in a richer historical perspective. Technology has always played an important role throughout the evolution of human communication, to the extent that several technological advances have contributed to the development of new models. Figure 1 illustrates the evolution through the four communication models detailed below.

- **one to one**—The invention of *writing* [5] made it possible to communicate complex messages from one person to one other person at the same time, without the requirement for these individuals to meet. Traditional writing only recently lost its dominant position in interpersonal communication to electronic media.
- **one to many**—Conveying the same message to larger audiences involved manual copying until the invention of the *printing press* [6]. Even in today's technological society, printed works remain an important means of spreading knowledge.
- **many to many**—The *World Wide Web* [7] made a radical change in the communication model by enabling bidirectional interactions. The scalable nature of the Web makes it indeed possible for more people than ever before to engage in worldwide communication.
- **between humans and machines**—The *Semantic Web* [8] aims to introduce another group of actors in the model: machines, which can range from software programs to any kind of electronic device. However, the degree of autonomy of such automated agents is currently limited.

Therefore, in the next section, we address the agent concept as a communication question, and elaborate on the current barriers between humans and machines on the Web. In Section III, we look at necessary changes in the future. Finally, we conclude in Section IV with a summary of what agents need.

II. Machine-accessible resources today

On today's Web, many resources are already machines-accessible in different ways and with varying degrees of automated interpretability. In most cases, clients need to be programmed for a specific purpose and tailored to a specific resource implementation. We will mention machine-accessible data embedded in HTML documents (both standardized and non-standardized), machine-accessible data in separate documents, and on-demand resources generated by Web services, and discuss their advantages and disadvantages.

 ISBN: 978-1-61208-248-6

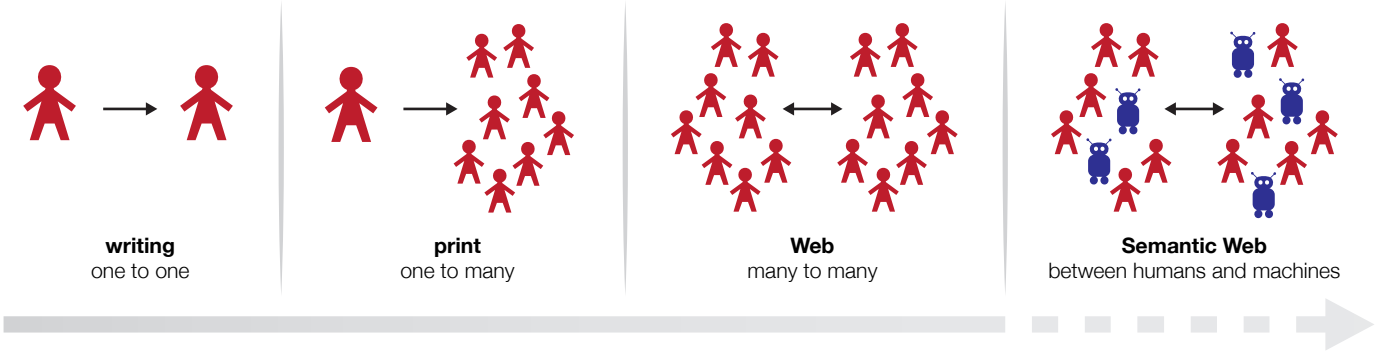

Fig. 1. Technology made communication evolve from a one-to-one to a many-to-many model. The future could bring human-machine intercommunication.

A. HTML with annotations

The Semantic Web's most common data format, RDF (Resource Description Framework, [9]), has a well-known counterpart that can be embedded in HTML, called RDFa [10] (derived "from RDF in *a*ttributes"). The recent 1.1 update supersedes the 1.0 version, which exclusively focused on XHTML. RDFa has become somewhat of a success story of the Semantic Web that also gained visibility outside of the scientific community. A prominent example of this is GoodRelations [11], an ontology to describe products, which was adopted by Google to enhance product search results [12]. Google thereby provided a clear incentive for product providers to enhance their HTML representations with machine-accessible information.

A second major incentive to use RDFa came from Facebook, who based the initial version of the Open Graph protocol on RDFa [13]. This protocol enables HTML pages to become objects people can talk about and share on Facebook and other social networks [14]. Given the importance of social media marketing nowadays, many website owners chose to provide RDFa metadata in the Open Graph vocabulary. This latter property is also the downside of Open Graph: the documentation seems to imply that the vocabulary choice is fixed. Instead of reusing existing ontologies [15], a new one was created, without providing links to define meaning in terms of other ontologies.

Another example of non-reuse was the introduction of Schema.org [16] by Google, Bing, and Yahoo!, widely considered an answer to Facebook's Open Graph protocol. Schema.org leaves the RDFa path by annotating human-readable text with HTML5 microdata [17], again with a specifically designed vocabulary. The Semantic Web community reacted quickly and released an RDFS schema for Schema.org [18], which eventually resulted in an official OWL version of the schema. The Schema.org annotation method has also been made compatible with RDFa Lite [19].

By now, the major issue with having similar but different vocabularies should be apparent: in how many vocabularies do we need to annotate a single HTML page to be understandable by all consumers? If any major RDFa consumer can impose a vocabulary on webmasters, annotating Web pages becomes a never-ending task. Only recently, one year after the launch of Schema.org, Twitter announced another annotation mechanism called Twitter Cards [20], which has a considerable overlap with the Open Graph vocabulary—and, consequently, with Schema.org. This implies that the *same* semantic content has to be expressed differently *three* times to be interpretable by the major Web traffic sources Google, Facebook, and Twitter. This is an alarming observation, since RDF ontologies were precisely created to enable interchangeable data, which is, after all, supposed to be the added value of semantic technologies.

From a communicational standpoint of view, we could say that, while common (and even standard) languages are used, every agent refuses to communicate in a lingo different from its own. It remains an open question whether any of the consumers of annotated HTML will support formats or vocabularies endorsed by others. As its Structured Data Testing Tool [21] shows, Google is able to extract RDFa and other data marked up with different vocabularies (including Open Graph metadata), but it is unsure whether this data will be used and if so, whether it will carry the same importance as Schema.org metadata. For the moment, only limited search result enhancements are performed, even if the annotations are written in the Schema.org vocabulary. However, the provided metadata fields are sufficiently powerful to enable a broad automated understanding of basic content properties, so increased usage can be expected in the future. To verify this, the aforementioned tool can generate a preview of how additional semantic annotations currently affect search results.

The diversity also forms a burden for implementors of intelligent agents, who cannot rely on one standardized vocabulary. For example, HTML itself has a single way of specifying a page title, namely the `title` tag, whereas each of the three aforementioned annotation mechanisms have *different* means of expressing the same thing. Fortunately, independent implementors are not bound by the business-driven decisions that prompted Facebook, Google, and Twitter to each favor or even exclusively support a different technique. If we assume the availability of an ontology that brings together similar terms from the different vocabularies, Semantic Web reasoning techniques can translate content from one vocabulary to another [22]. Therefore, support for one format could be sufficient for agents to support all of them.

 ISBN: 978-1-61208-248-6

B. Machine-targeted document formats

In addition to embedding machine-processable data in human-targeted representation, it can also be expressed directly in a machine-targeted format. One such format is of course RDF. We can distinguish two cases: either the RDF version is a machine-friendly alternative representation of a human-targeted document, or either the RDF version is the unique representation of the data. The first case is not unlike RDFa, where an HTML document always accompanies the embedded RDF data. However, isolating machine data from human data has the benefit of separating concerns, since only one of the data streams is needed at a time. Thanks to the Linked Data movement and principles, many datasets have already been made available as RDF [23].

At the moment, RDF is however not the default choice for many Web developers to expose data in a machine-processable way. While a few years ago XML was a common structured format, the JavaScript Object Notation (JSON, [24]) has become increasingly popular and is now ubiquitous, mainly thanks to its simplicity and native compatibility with the dominant Web client language JavaScript. JSON allows to represent complex hierarchical data efficiently, but unfortunately does not have any inherent semantics associated with data fields. As a result, clients have to be programmed to understand a specific type of JSON information. Furthermore, since JSON lacks native identification support (such as URIs in RDF), it if difficult to identify individual resources or to make circular references.

The JavaScript Object Notation for Linking Data (JSON-LD, [25]) provides a solution to these problems by bridging between JSON and RDF. It adds an `id` property to JSON fragments to enable identification of resources, and a `context` property to identify the semantics of data fields. Communication-wise, JSON-LD and RDF have equal expressive power, but the latter has the benefit of native JavaScript support, providing maximal parsing speed and familiarity for developers without prior Semantic Web knowledge. In that sense, JSON-LD is a hybrid language: its semantic grounds in RDF make it interpretable by automated clients, whereas its JSON format offers an accessible interface for developers of such clients.

C. Web services

As stated in the introduction, the envisioned tasks of agents are not limited to information retrieval. Far more possibilities—and a greater complexity—reside in performing so-called *world-changing* actions, which alter the state of digital or real-world objects. Many Web services [26] offer such actions, for example posting comments, ordering books, or reserving tickets. Preprogrammed clients do not need to know what task a service performs, because the author of the client is the one who interprets the service's functionality. The situation is different for an autonomous agent: if it wants to complete a certain task, it has to find a service that offers the desired functionality and it then needs to figure out how to invoke that service. Similarly to the role of machine-interpretable metadata in HTML documents, clients need a service's metadata to understand its capabilities and modalities.

Currently, most Web service documentation is only written in human language for developers. Many frameworks for discovery and machine-processable description of invocation modalities have been created, notably UDDI [27] and WSDL [28]. However, these frameworks do not reach beyond the technical aspects of a service, and therefore serve as assisting technologies for application developers instead of as metadata for autonomous agents.

The crucial difference with static data is the world-changing aspect: when *retrieving* information resources, no harm can be done because the application state doesn't change. Since a service can have *side-effects*, it can potentially be dangerous to issue requests without understanding what is going to happen. While the irrelevance of data can be determined afterwards, upon which that data can safely be discarded, the determination of a service's results after its invocation comes too late if that action cannot be rolled back. The write aspect is indeed as important as the read aspect, but currently underdeveloped [29]. Therefore, autonomous intelligent agents on today's Web are scarce, and those that exist are limited to a specific, pre-programmed domain, such as is the case with Apple's Siri.

III. Machine-accessible resources for the future

We will now have a look at techniques to make resources machine-accessible that are currently under development or research, of which we believe they will play a major role on the future Web for agents. Some of these techniques are already in use today, while others are still in the research stage or awaiting adoption.

A. The uniform interface

A key part in our vision is the unification of static resources and services, blurring the distinction between them, and providing a uniform interface to access all resources. This makes agent development considerably simpler since it needs to be programmed only against a single interface, instead of requiring different bindings for every service. Such a uniform interface is featured in the work of Roy T. Fielding, whose doctoral dissertation details the architectural style that underpins the original design of the Web's HyperText Transfer Protocol (HTTP, [30]), and is called REpresentational State Transfer (REST, [31]). The REST architectural style defines several *constraints* on the communication between clients and servers. These constraints introduce certain desirable properties in the systems that obey them, including reliability and scalability. Although HTTP provides the necessary interfaces to build applications that function according to the REST architectural constraints, not many of today's Web applications actually implement all of them.

According to Fielding, four constraints contribute to the uniform interface: *identification or resources*, *manipulation of resources through representations*, *self-descriptive messages*, and *hypermedia as the engine of application state*. We will now go through each of these constraints, and indicate why they are important for autonomous agents.

 ISBN: 978-1-61208-248-6

1) Identification of resources: On the Web, resource identification is achieved through the use of Uniform Resource Locators (URLs, [32]). A resource is defined as a temporarily varying membership function, of which the *mapping definition* (the function itself) remains constant, but the *mapped entity* can vary in time. For example, the resource identified by the URL `http://magazine.example.org/issues/current` could be defined as *"the latest issue of Example magazine"*, but could, depending on the month, be the January or February issue. This resource would be separate from *"the January issue of Example magazine"*—even if the current month *is* January and the mapped entities are thus the same—because the mapping definition is different.

The benefit for agents is clearly simplification, because everything is a resource and is identifiable by a URL. This means there is no need (or space) for a "service", since everything is modelled in terms of resources. For example, instead of a service that returns tomorrow's weather forecast for a specified city, the server provides a resource that gives the same thing and has a distinct URL. That way, this resource is indistinguishable from what we have previously called "static data"—and there is no apparent reason why it should not be. The reason we often *do* see a distinction is because the details of the underlying server implementation are inadvertently surfacing (*e.g.*, the forecasts are retrieved by a specific script, which is wrapped as a service rather than modelled as a set of resources with identifiers).

2) Manipulation of resources through representations: In HTTP communications, resources themselves are not exchanged or manipulated, but rather representations thereof. A single resource might have multiple representations in different formats, which allows clients and servers to perform *content negotiation* to mutually decide on a representation they both understand. For example, a browser would indicate (through `Accept` headers) a preference for HTML in English or Spanish (which can be set by the user). Another client might have a preference for plaintext or JSON. The server will try to accommodate those preferences to the extent possible.

From a communication viewpoint, this is very beneficial for agents. They can access, work with, and communicate about the *same* resources as those on the human Web. In fact, the Web for humans and the Web for agents are the *same* Web: only the representations are different, since machines are yet incapable of understanding human language. For the time being, agents can employ content negotiation to ask for RDF or JSON-LD content, and perhaps even indicate their preference for a specific vocabulary. Should the preferred version not be available, the server can choose to serve a best-effort representation such as HTML with RDFa.

3) Self-descriptive messages: One of the aspects of self-descriptiveness of messages is *statelessness*: every message should contain all metadata necessary to understand its meaning. For example, to get from the first to the second page of a listing, a client does not send a "next" message, but rather a request for the second page resource. This ensures that no other message is required to understand the request.

Another aspect of self-descriptiveness is the use of *standard methods*. HTTP provides only a few generic methods, such as GET, PUT, POST, and DELETE. This small number of methods shifts the focus from sending messages to objects (as is the case in object-oriented programming) to retrieving and manipulating representations of resources. For example, instead of sending the `findWeather` message with *"Boston"* as an argument, we perform a standard GET request on the *"current weather in Boston"* resource. This again considerably simplifies agent development, plus it offers *guarantees* for several methods: GET guarantees that it will preserve resource state, while DELETE does not. Both methods guarantee that they can be executed multiple times without causing additional effects from the first execution, which POST cannot.

Some Web applications today violate statelessness and/or do not respect the guarantees of the standard methods. This can be problematic for agents, as they need to correctly assess the consequences of requests they make. Even on the human Web, problems can arise if a method that should not induce side-effects suddenly does [33].

4) Hypermedia as the engine of application state: The last constraint necessary to achieve a uniform interface is known as the hypermedia or HATEOAS (Hypermedia As The Engine Of Application State) constraint. Due to lack of time, Fielding's dissertation does not elaborate on this constraint, but he did so in later blog posts [34]. Basically, the hypermedia constraint says that a representation of a resource should contain the necessary *controls* to chose possible next steps or actions. For example, in the hypermedia format HTML such controls are links, forms, buttons, *etc*.

If we look at the human Web, we see that the hypermedia constraint is well-implemented: people never have to manually type a URL in the address bar to change a page within the same website. The situation is different for machine-targeted resources: often, the developer has to configure an agent to use or construct specific URLs. This limits the capabilities of agents, since they need hypermedia controls as much as humans do. With RDF content, such controls are implicitly defined if URLs (as a special case of URIs) are used as resource identifiers, since these identifiers then serve as links that can be followed to look up more information about the resource (known as *dereferencing*). However, this only concerns static resources, as the RDF standard currently does not define semantics other than retrieval.

If we look at the act of communication, the hypermedia constraint makes messages provide the *context* necessary to gain more insight in the meaning of the resource, both for humans and agents. Note that, although the *message* should be self-descriptive, the *representation* carried within that message can have controls to other resources—this is exactly the purpose of hyperlinks. Although it is possible to add controls to machine-targeted representations at a later stage [35], this often involves remodeling the Web application in a resource-oriented way. The hypermedia constraint is, however, a requirement for agents that want to autonomously discover and consume resources.

 ISBN: 978-1-61208-248-6

B. Semantic description of functionality

While the uniform interface thus creates the necessary environment for agents, the RDF content type does not provide sufficient semantics to perform all possible tasks. This is because RDF is intended for static documents in the first place. Concretely, if an RDF document contains a URL of a certain resource, then an agent can predict what will happen if a GET request is issued: the server will return a representation of the identified resource. The HTTP specification [30] also describes the effect of other methods: PUT will place the entity supplied with the request at the specified URL, and DELETE will remove the identified resource (given the agent has sufficient permissions to issue such requests).

However, as the HTTP specification states, *"[t]he actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI."* This indicates that HTTP provides no means for an agent to predict the effect of issuing a POST request to a resource. This is a major issue, since the POST method is needed often: the specification mentions examples such as posting messages, handling form data, and annotating resources. The issue does not occur on the human Web, because the POST form is usually accompanied by textual and visual clues, and because we understand the context in which the form is presented to us. Agents do not have similar clues at their disposal, since RDF only describes resources statically. Because of this, agents require a description of the dynamic aspects of Web resources in order to understand the effect of methods such as POST.

Earlier description techniques, such as OWL-S [36] and WSMO [37], were designed with the Web service model in mind. Web services employ a so-called *overloaded* form of POST, the semantics of which do not correspond to those in the HTTP specification. Therefore, these techniques are not the right match for Web applications with a REST architecture.

Several description techniques that specifically target REST are currently the subject of research. One approach to combine the REST principles and RDF is to start from the SPARQL query language [38], since it supports update operations from version 1.1 onwards [39]. This can indeed be a way for RDF-aware agents to perform state-changing operations on resources. However, SPARQL is a technology specifically for machines and is therefore not suited for environments where other representations are also important, as in the Web for agents that we envision. Linked Open Services (LOS, [40]) expose functionality on the Web using a combination of HTTP, RDF, and SPARQL in a way that is more friendly towards different representation formats, although the hypermedia constraint is currently not addressed by this technique.

RESTdesc [41] is a semantic format specifically designed to describe the effects of POST requests in hypermedia-driven Web applications. It aims to complement static RDF descriptions of resources with a dynamic view in an RDF-like language. Its purpose is to support autonomous agents in the execution of non-preprogrammed actions on dynamic resources on the Web.

From a communication perspective, finding a way of letting machines consume dynamic resources with the same ease as static resources comes down to semantically expressing what the effects of manipulating a specific resource are. Eventually, this should enable agents to choose specific resource actions based on the functional goals they want to achieve. That way, a user could instruct an agent to perform a task, which the agent then can break down in different resource manipulations. Furthermore, such a resource-oriented approach integrates well with existing machine-readable data on the Web, whereof Linked Data is the most prominent example. Agents can then use this data seamlessly to achieve their goals [42].

IV. Conclusion: what the Web for agents needs

In this paper, we have zoomed in on the obstacles for agents on today's Web. Many competing machine-processable annotation techniques exist, as well as a large variety of RDF documents. World-changing actions are performed through Web services, which are separate from other documents on the Web. In an ideal Web for agents, there is a uniform interface to all resources, both static and dynamic, removing the current distinction between documents and services. At the same time, agents will need a mechanism to understand the effects of performing state-changing operations on resources.

As long as machines are not able to understand human language, semantic technologies will remain important for agents to derive meaning from resources on the Web. We believe, however, that this is best achieved in a transparent way such as with the resource and representations model, which exposes the *same* resources for both human and machine Web clients. Or in the words of the famous Scientific American article: *"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."* [8]

If we want to take the step to the next communication model, in which machine agents have the same capabilities on the Web as humans, we must be willing to work on the current issues and start building Web applications with all aspects of the uniform interface. We have only seen the tip of the iceberg of what is possible with agent technology, and opportunities will only increase as the number of devices that join the Web grows on a daily basis. However, it will take a strong cooperation between Web resource providers and agent developers to make the rise of the Web for agents happen in the not-too-distant future.

Acknowledgments

The described research activities were funded by Ghent University, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

 ISBN: 978-1-61208-248-6

REFERENCES

[1] B. G. Buchanan, "A (very) brief history of artificial intelligence," AI *Magazine*, vol. 26, no. 4, pp. 53–60, 2005.

[2] A. Halavais, *Search Engine Society*, ser. Digital media and society series. Cambridge: Polity, 2008.

[3] J. Hendler, "Agents and the Semantic Web," IEEE *Intelligent Systems*, vol. 16, no. 2, pp. 30–37, Mar–Apr 2001.

[4] J. Aron, "How innovative is Apple's new voice assistant, Siri?" *The New Scientist*, vol. 212, no. 2836, p. 24, 2011.

[5] H. Haarmann, *Geschichte der Schrift*, ser. Wissen in der Beck'schen Reihe. C. H. Beck, 2002, vol. 2198.

[6] T. Carter and L. Goodrich, *The invention of printing in China and its spread westward*. New York: Ronald Press, 1955.

[7] T. Berners-Lee and M. Fischetti, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. San Fransisco: Harper, 2000.

[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[9] G. Klyne and J. J. Carrol. (2004, Feb.) Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[10] B. Adida, M. Birbeck, S. McCarron, and I. Herman. (2012, Jun.) RDFa core 1.1. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/2012/REC-rdfa-core-20120607/

[11] M. Hepp, "GoodRelations: An ontology for describing products and services offers on the Web," *Knowledge Engineering: Practice and Patterns*, pp. 329–346, 2008.

[12] E. Franzon. (2012, Nov.) Google recommends using RDFa and the GoodRelations vocabulary. [Online]. Available: http://semanticweb.com/google-recommends-using-rdfa-and-the-goodrelations-vocabulary_b909

[13] I. Facebook. (2010) The Open Graph protocol. [Online]. Available: http://opengraphprotocol.org/

[14] M. Zuckerberg. (2010, Apr.) Building the social Web together. [Online]. Available: https://blog.facebook.com/blog.php?post=383404517130

[15] E. Simperl, "Reusing ontologies on the Semantic Web: A feasibility study," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 905–925, 2009.

[16] Google, Inc., Yahoo, Inc., and Microsoft Corporation. (2011, Jun.) Schema.org. Specification. [Online]. Available: http://schema.org/docs/schemas.html

[17] Web Hypertext Application Technology Working Group. HTML – microdata. [Online]. Available: http://www.whatwg.org/specs/web-apps/current-work/multipage/microdata.html

[18] M. Hausenblas and R. Cyganiak. (2011, Jun.) What is schema.rdfs.org? [Online]. Available: http://schema.rdfs.org/

[19] M. Sporny. (2012, Jun.) RDFa lite 1.1. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/2012/REC-rdfa-lite-20120607/

[20] A. Roomann-Kurrik. (2012, Jun.) Twitter Cards. [Online]. Available: https://dev.twitter.com/blog/twitter-cards

[21] Google. Structured data testing tool. [Online]. Available: http://www.google.com/webmasters/tools/richsnippets

[22] A. Hogan, J. Pan, A. Polleres, and Y. Ren, "Scalable OWL 2 reasoning for Linked Data," in *Reasoning Web. Semantic Technologies for the Web of Data*, ser. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, 2011, vol. 6848, pp. 250–325.

[23] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data – The Story So Far," *International Journal On Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[24] D. Crockford. (2006, Jul.) The application/json media type for JavaScript Object Notation (JSON). IETF Request for Comments. [Online]. Available: http://www.ietf.org/rfc/rfc4627

[25] M. Lanthaler and C. Gütl, "On using JSON-LD to create evolvable RESTful services," in *Proceedings of the Third International Workshop on RESTful Design*, Apr. 2012, pp. 25–32.

[26] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web services architecture," IBM *Systems Journal*, vol. 41, no. 2, pp. 170–177, Apr. 2002.

[27] T. Bellwood, S. Capell, L. Clement, J. Colgrave, M. J. Dovey, D. Feygin, A. Hately, R. Kochman, P. Macias, M. Novotny, M. Paolucci, C. von Riegen, T. Rogers, K. Sycara, P. Wenzel, and Z. Wu. (2004, Oct.) UDDI version 3.0.2. OASIS. [Online]. Available: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

[28] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. (2001, Mar.) Web Services Description Language (WSDL) 1.1. W3C Note. [Online]. Available: http://www.w3.org/TR/wsdl

[29] S. Coppens, R. Verborgh, M. Vander Sande, D. Van Deursen, E. Mannens, and R. Van de Walle, "A truly Read-Write Web for machines as the next-generation Web?" in *Proceedings of the SW2012 workshop*, Nov. 2012. [Online]. Available: http://stko.geog.ucsb.edu/sw2022/sw2022_paper3.pdf

[30] R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. (1999, Jun.) Hypertext Transfer Protocol – HTTP/1.1. IETF Request for Comments. [Online]. Available: http://www.ietf.org/rfc/rfc2616

[31] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 115–150, May 2002.

[32] T. Berners-Lee, L. Masinter, and M. McCahill. (1994, Dec.) Uniform Resource Locators (URL). IETF Request for Comments. [Online]. Available: http://www.ietf.org/rfc/rfc1738

[33] R. Verborgh. (2012, Jul.) GET doesn't change the world. [Online]. Available: http://ruben.verborgh.org/blog/2012/07/19/get-doesnt-change-the-world/

[34] R. T. Fielding. (2008, Oct.) REST APIs must be hypertext-driven. Untangled – Musings of Roy T. Fielding. [Online]. Available: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

[35] O. Liskin, L. Singer, and K. Schneider, "Teaching old services new tricks: adding HATEOAS support as an afterthought," in *Proceedings of the Second International Workshop on RESTful Design*. ACM, 2011, pp. 3–10.

[36] D. Martin, M. Burstein, J. Hobbs, and O. Lassila. (2004, Nov.) OWL-S: Semantic Markup for Web Services. W3C Member Submission. [Online]. Available: http://www.w3.org/Submission/OWL-S/

[37] H. Lausen, A. Polleres, and D. Roman. (2005, Jun.) Web Service Modeling Ontology (WSMO). W3C Member Submission. [Online]. Available: http://www.w3.org/Submission/WSMO/

[38] E. Wilde and M. Hausenblas, "RESTful SPARQL? you name it! – aligning SPARQL with REST and resource orientation," in *Proceedings of the 4th Workshop on Emerging Web Services Technology*. ACM, 2009, pp. 39–43.

[39] E. Prud'hommeaux and A. Seaborne. (2008, Jan.) SPARQL Query Language for RDF. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[40] R. Krummenacher, B. Norton, and A. Marte, "Towards Linked Open Services and Processes," in *Proceedings of the Third future internet conference on Future Internet*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 68–77.

[41] R. Verborgh, T. Steiner, D. Van Deursen, S. Coppens, J. Gabarró Vallés, and R. Van de Walle, "Functional descriptions as the bridge between hypermedia APIs and the Semantic Web," in *Proceedings of the Third International Workshop on RESTful Design*. ACM, Apr. 2012.

[42] J. Domingue, C. Pedrinaci, M. Maleshkova, B. Norton, and R. Krummenacher, "Fostering a relationship between Linked Data and the Internet of Services," in *The Future Internet*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6656, pp. 351–364. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20898-0_25

 ISBN: 978-1-61208-248-6