Triple-space computing: Semantic Web Services based on persistent publication of information

D. Fensel

Digital Enterprise Research Institute (DERI)
DERI Innsbruck, Leopold-Franzens Universität Innsbruck, Austria
DERI Galway, National University of Ireland, Galway
dieter.fensel@deri.org

Abstract. This paper discusses possible routes to moving the web from a collection of human readable pieces of information connecting humans, to a web that connects computing devices based on machine-processable semantics of data and distributed computing. The current shortcomings of web service technology are analyzed and a new paradigm for fully enabled semantic web services is proposed which is called **triple-based** or triple-space computing.

1 Introduction

The web is a tremendous success story. Starting as an in-house solution for exchanging scientific information it has become, in slightly more than a decade, a world wide used media for information dissemination and access. In many respects, it has become the major means for publishing and accessing information. Its scalability and the comfort and speed in disseminating information is unprecedented. However, it is solely a web for humans. Computers do not have access to the provided information and in return do not provide any support in processing this information. Two complementary trends are about to change this transformation of the web, from being for humans only, into a web that interweaves computers to provide support for human interactions at a much higher level than is available with current web technology.

- The semantic web is about adding machine-processable semantics to data. The computer can "understand" the information and therefore process it on behalf of the human user (cf. [Fensel, 2003]).
- Web services try to employ the web as a global infrastructure for distributed computation, for integrating various applications, and for the automatization of business processes (cf. [Alonso et al., 2003]). The web will not only be the place where human readable information is published but the place where global computing is realized.

Eventually, semantic web services promise the combination of semantic web with web service technology. A fully mechanized web for computer interaction would become a new infrastructure on which humans organize their cooperations and business relationships (cf. [Fensel & Bussler, 2002]).

These trends promise to provide the holy grail of computer science. The semantic web promises to make information understandable to a computer and web services promise

to provide smooth and painless integration of disparate applications. Web services offer a new level of automatization in eWork and eCommerce, where fully open and flexible cooperation can be achieved, on the fly, with low programming costs. However, the current implementations of web service technology are still far from reaching these goals. There are a couple of obvious reasons for this. Integrating heterogeneous and dynamically changing applications is a tremendous task. Currently, a bizarre flow of standards and pseudo-standards are published to achieve this goal. We are still far away from the point where we can ensure that there is emerging consensus around some proposals and from deciding on whether these proposals deliver what they are promising. Also many of the existing proposals cover the required functionality at a very low level, only. Spoken in layman term's, remote procedure calls over HTTP may not be the right level of functionality to align business processes in a smooth and scalable fashion. Established standards in the pre-web eCommerce area such as EDI/ EDIFACT[1] provides a much higher level of support for mechanizing business transactions.

These obstacles may eventually be overcome, however these may also be an indication of a deeper problem around web services. Actually as we show in this paper, web services do not have much in common with the web. They are based on message exchange rather than on addressable and persistent publication, which is a key principle underlying the web. Thus, they have to deal with all the issues around message exchange and how to implement reference-, time-, and space-decoupled interactions. Actually web services are not sufficiently advanced to use the web as a means of information publishing and access.

Investigating true web services, that are based on the web paradigm for exchanging information, is at the core of this paper. We will investigate the potential of tuple- or space-based computing and the necessity to combine it with semantics. We will call this proposal triple-based computing and we show how this naturally fits into the vision of a semantic web. In a nutshell, realizing services on top of the semantic web may be a more realistic pathway to achieving semantic web services rather than trying to enrich web services with semantic annotations.

The contents of this paper are organized as follows. Section 2 discusses the web, the semantic web, web services, and eventually semantic web services as the web of machines that may evolve from the web primarily for humans. Section 3 analyzes web services and questions whether they actually have much to do with the web. Section 4 provides a vision on what actual web services could look like. They would use the web as a global tuplespace and the required extension of this tuplespace into a triplespace naturally adds semantics and the semantic web to them. Section 5 concludes the paper.

2 Elements and directions in achieving semantic web services

We split the discussion of elements and directions in achieving semantic web services into two parts. Section 2.1 discusses the major elements the future web may be based on. We start the discussion with the web itself and continues with discussing the

1. http://www.unece.org/trade/untdid/welcome.htm

semantic web, web services, and their combination in semantic web services. Section 2.2 follows up the discussion by investigating two different paths that may eventually lead from the current web to semantic web services.

2.1 Four dimensions describing the future of the web

Figure 1 illustrates four major stages in the development of the web: (1) the web as a collection of information accessible to the human reader; (2) the semantic web that adds machine-processable semantics and mechanized information processing; (3) web services that employ the web as a platform for distributed computing; and (4) semantic web services that combine both in providing mechanized service discovery, parametrization, composition, and execution. We will briefly elaborate on all four stages.

The **World Wide Web** is a big and impressive success story, both in terms of the amount of available information and of the growth rate of human users. It has started to penetrate most areas of our daily lives and business. This success is based on it's simplicity. The restrictiveness of HTTP and HTML allowed software developers, information providers, and users to gain easy access to this new media, helping it to reach a critical mass. However, this simplicity may hamper the further development of the Web. Or in other words: What we see currently is the very first version of the web and the next version will probably be even bigger and much more powerful compared to what we have now. It started as an in-house solution for a small group of users. Soon, it established itself as a world-wide communication media for hundreds of millions of people. In a small number of years it will interweave one billion people and will penetrate many more types of devices than just computers.

It is also clear that the current state of web technology is generating serious obstacles for it's further growth. The bottlenecks of current web technology create problems in searching information, problems in extracting information, problems in maintaining

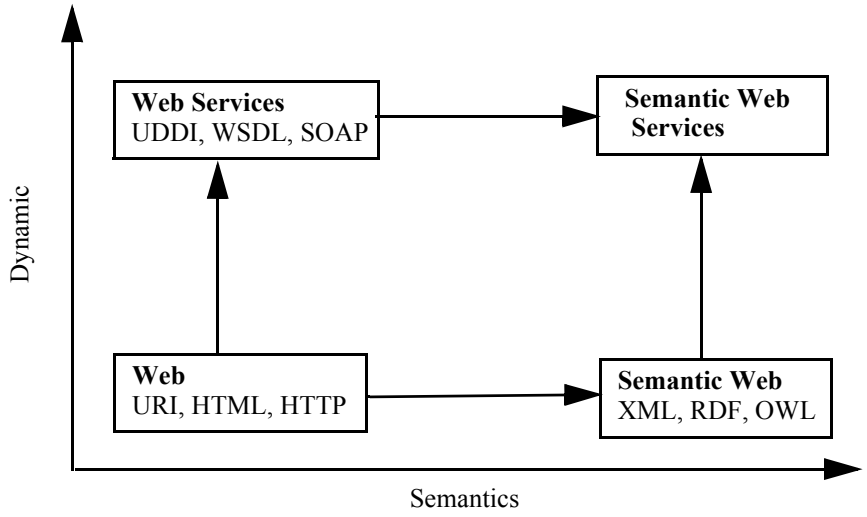

Figure 1 The four major stages in the development of the Web.

information, and problems in generating information. All these problems are caused by the simplicity of current web technology. Computers are used as devices to post and render information. However, they do not have any access to the actual content and therefore can provide only very limited support in accessing and processing this information. In consequence, the main burden in accessing, extracting, interpreting, and processing information is left to the human user.

Tim Berners-Lee created the vision of a **Semantic Web** that provides automated information access based on machine-processable semantics of data and heuristics that make use of these meta data. The explicit representation of the semantics of data accompanied with domain theories (i.e., ontologies) will enable a web that provides a qualitatively new level of service. New recommendations[2] such as XML, RDF, and OWL allow the adding of machine-processable semantics to the information present on the web. The semantic web will weave together an incredibly large network of human knowledge, with complementary machine processability. Various automated services will support the user in achieving goals via accessing and providing information in a machine-understandable form. This process may ultimately lead to extremely knowledgeable systems with various specialized reasoning services that may support us in nearly all aspects of our daily life, becoming as necessary for us as access to electric power.

The current web is mainly a collection of information but does not yet provide support in processing this information, i.e., in using the computer as a computational device. Recent efforts around UDDI[3], WSDL[4], and SOAP[5] have tried to lift the web to a new level of service. Software applications can be accessed and executed via the web based on the idea of **Web services**. Web services can significantly increase the web architecture's potential, by providing a way of automating program communication. Therefore, they are the focus of much interest from various software development companies. Web services connect computers and devices with each other using the Internet to exchange data and combine data in new ways. The key to web services is on-the-fly software composition through the use of loosely coupled, reusable software components. This has fundamental implications in both technical and business terms. Software can be delivered and paid for as fluid streams of services as opposed to packaged products. It is possible to achieve automatic, ad hoc interoperability between systems to accomplish organizational tasks. Examples include both business application, such as automated procurement and supply chain management, and non-commercial applications, which include military applications. Web services can be completely decentralized and distributed over the Internet and accessed by a wide variety of communications devices. Organizations can be released from the burden of complex, slow and expensive software integration and instead focus on the value of their offerings and mission critical tasks. The dynamic enterprise and dynamic value chains would become achievable and may be even mandatory for competitive advantage.

---

2. http://www.w3c.org/
3. http://www.uddi.org/
4. http://www.w3.org/TR/wsdl
5. http://www.w3.org/TR/soap12-part1/

Still, more work needs to be done before the web service infrastructure can make this vision come true. Current web service technology provides limited support in mechanizing service recognition, service configuration and combination (i.e., realizing complex workflows and business logics with web services), service comparison and automated negotiation. In a business environment, the vision of flexible and autonomous web service translates into automatic cooperation between enterprise services. Any enterprise requiring a business interaction with another enterprise can automatically discover and select the appropriate optimal web services relying on selection policies. This can be achieved by adding machine-processable semantics to the description of web services based on semantic web technology. **Semantic web services** can be invoked automatically and payment processes can be initiated.[6] Any necessary mediation would be applied based on data and process ontologies and the automatic translation and semantic interoperation. An example would be supply chain relationships where an enterprise manufacturing short-lived goods must frequently seek suppliers as well as buyers dynamically. Instead of employees constantly searching for suppliers and buyers, the web service infrastructure does it automatically within the defined constraints. Other applications areas for this technology are Enterprise-Application Integration (EAI), eWork, and Knowledge Management.

2.2 Two ways to heaven

When taking a closer look at Figure 1 it turns out that *two potential paths* in achieving semantic web services are implicitly present there. You can move to semantic web services via the web service track or via the semantic web track (see Figure 2).[7]

Projects such as DERI[8] and DIP[9] follow the first path. The current web service stack is

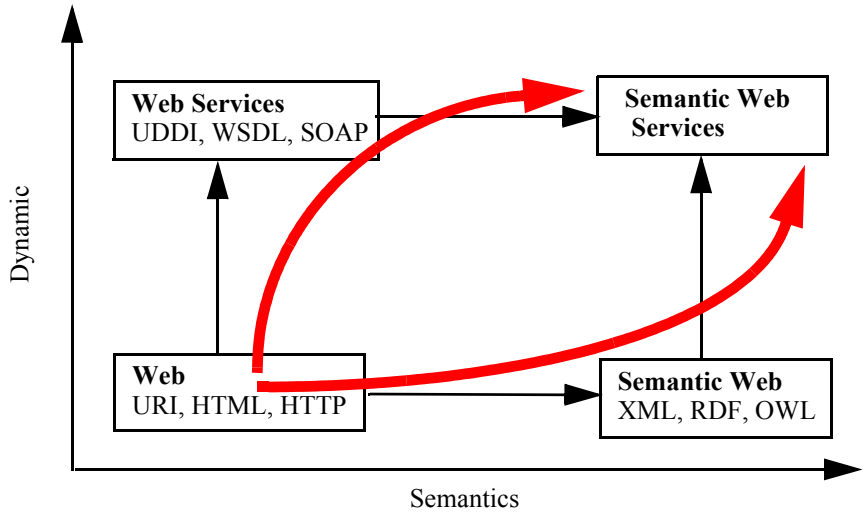

Figure 2 The two major footpaths in developing the Web.

6. See for initiatives in this area OWL-S (http://www.daml.org/services/), IRS-II (http://kmi.open.ac.uk/projects/irs/), and WSMO (http://www.wsmo.org/).
7. This section results from personal communication with Tim Berners-Lee and Eric Miller.

taken as a starting point and semantic annotations are designed to complement these elements. Semantics should be added to WSDL interface descriptions and choreography and orchestration elements. A strong mediation service is developed to cope with all the various miss-matches in data, protocol and process specifications.

In fact, this is not the only possible road to semantic web services. Alternatively one could directly focus on further developing the semantic web. By putting more and more ontologies and semantically annotated data on the web, services will evolve naturally that make use of these descriptions. In practical terms, one could ontologize existing standards such as EDI and EDIFACT and invite new business partners to make use of these public descriptions for implementing their trading relationships. Instead of deploying standards for service descriptions (and there is already a frightening number of pseudo standards in the arena) one could provide more and more reusable formalized descriptions on the web of services that can be exploited to achieve their functionality. This idea will be discussed further in the next section when we discuss the severe shortcomings of the current web service infrastructure.

3 Are web services really web services? - No!

Besides their name, *web* services do not have much to do with the web. Let's illustrate this briefly by assuming a time machine would bring us back to the pre-web time. What was a very common way, back then, of accessing a research paper? One was posting an email kindly asking for the paper and a friendly colleague posting it as an attachment. Dissemination of information was based on message exchange. The communication overhead in publishing and accessing information was high and dissemination was therefore quite limited and slow. Then the web came into being and changed the situation significantly. The author had to publish the paper once by putting it on his web page. After this, he could forget about it and focus on writing new papers. New services such as citeseer[10] even ensure durability of this publication beyond the life time of a web page (i.e., they disable the delete operation on the information space). All the potential readers could get instant access to the paper without requiring a two-stage message-exchange process. This tremendously scaled and speeded up the dissemination process of information. When comparing web services with this essential web principle it becomes quite obvious that web services are **not** about the web.

Web services require close coupling of applications they integrate. Applications communicate via message exchange requiring strong coupling in terms of reference and time. The communication has to be directed to the web service addressed and the communication must be synchronous. If both parties do not implement and jointly agree on the specific way this mechanism is implemented, then the applications must support asynchronous communication. The web is strongly based on the opposite principles. Information is published in a persistent and widely accessible manner.[11] Any other application can access this information at any point in time without having to request

8. http://www.deri.org/
9. http://dip.semanticweb.org/
10. http://citeseer.ist.psu.edu/cs
11. For privacy issues, protected sub-fragments of the web can be defined.

the publishing process to directly refer to it as a receiver of it's information. It is true that web services uses the internet as a transport media (relying on protocols such as FTP, SMTP, or HTTP), however that is all they have in common with the web.

Given this obvious evidence it is surprising that many more authors already have not complained about the erroneous naming of web services, that could be likened to the situation in the emperor's new clothes. Actually, the criticisms of the REST community (cf. [Fielding, 2000]) back up this argument and the position of this paper. Their two major criticisms around web services are about improper usage of URIs and messing up the state-less architecture of the web (cf. [Fielding & Taylor, 2002], [zur Muehlen et al., 2004]).

When sending and receiving SOAP messages, the content of the information is hidden in the body and *not addressed as an explicit web resource* with it's own URI. Therefore, all web machinery involving caching or security checks is disabled since its use would require the parsing and understanding of all possible XML dialects that can be used to write a SOAP message. Referring to the content via an explicit URI in an HTTP request would allow the content of a message to be treated like any other web resource.

The web service stack can be used to model state-full resources. However, one of the basic design principle of the web and REST architectures is not to provide state-full protocols and resources explicitly. Thus, application integration and servers for this architecture are easy to build. Every HTTP request for a URI should retrieve the same contents independently of what has happened before in other sessions or in a history of the current session. This allows thin servers to be used, that do not need to store, manage and retrieve the earlier session history, for the current session.[12] When a stateful conversation is required this should be explicitly modelled by different URIs. In consequence, *there should not be one URI for a web service and hidden ways to model and exchange state information* but each potential state of a web service should be explicitly addressable by a different URI. This conforms to the web and REST's way of modelling a stateful conversation for a state-less protocol and adhers to their architecture.

These criticisms of the REST community reinforces this paper's arguments. Web services do not rely on the central principles of the web: publication of information based on a global and persistent URI, instead, stateful conversations based on the hidden content of messages are established. The next section explores what web services would look like that are fully based on the web and it's underlying principles that made it such a success.

4 Triple-spaced computing

This section will discuss what a service paradigm that conforms with the basic principles of the web could look like. We start by discussing tuple-spaced computing as a paradigm to exchange data between applications. Then we introduce the concept of semantic self-description of information, which naturally lead us into a discussion of the

12. Actually cookies are a work-around of this principle, however they break when a client is run on different machines.

triple space.

4.1 Tuple-spaced computing

Tuple-based computing has been introduced in parallel programming languages, such as Linda, to implement communication between parallel processes (cf. [Gerlernter, 1992]). Instead of sending messages backward and forward a simple means of communication is provided. Processes can write, delete[13], and read tuples from a global persistent space.[14] A tuple is a set of ordered typed fields, each of which either contains a value or is undefined and a tuplespace is an abstract space containing all tuples and visible to all processes. The API for this is extremely simple and all complexity in message processing disappears (actually it is hidden in the middleware that implements the tuplespace). This tuplespace is similar to a blackboard in expert systems, where rules do not send messages to all other rules when they derive a fact. Rather, this is published by adding it to the publicly-visible board.

Tuple or space-based computing has one very strong advantage: It de-couples three orthogonal dimensions involved in information exchange (cf. Figure 3): reference, time, and space.

- Processes communicating with each other do not need to explicitly know each other. They exchange information by writing and reading tuples from the tuplespace, however, they do not need to set up an explicit connection channel, i.e., *reference-wise the processes are completely de-coupled*.
- Communication can be completely asynchronous since the tuplespace guarantees persistent storage of data, i.e., *time-wise the processes are completely de-coupled*.
- The processes can run in completely different computational environments as long as both can access the same tuplespace, i.e., *space-wise the processes are completely de-coupled*.

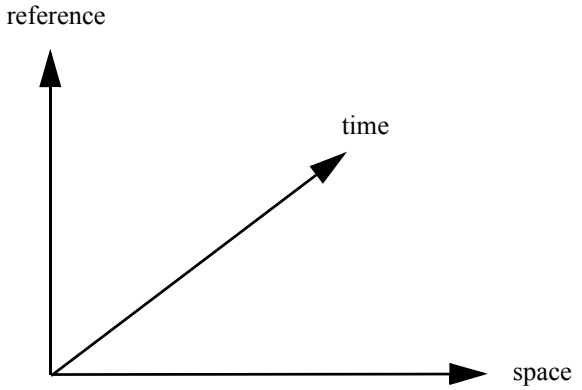

Figure 3 Three separate dimensions of cooperation, taken from [Angerer, 2002].

13. Actually, deleting tuples may not really be necessary in an exponentially growing space such as the web.
14. *Global* in the *local* framework of an application that is decomposed by parallel processes.

This strong decoupling in all three relevant dimensions has obvious design advantages for defining reusable, distributed, heterogeneous, and quickly changing applications like those promised by web service technology. Also, complex APIs of current web service technology are replaced by simple read and write operations in a tuplespace. Notice that a service paradigm based on the tuple paradigm also revisits the web paradigm: information is persistently written to a global place where other processes can smoothly access it without starting a cascade of message exchanges.

Johanson and Fox [Johanson & Fox, 2004] describe the application of tuplespaces for coordination in interactive work spaces, focussing on providing software infrastructure for the dynamic interaction of heterogeneous and ad hoc collections of new and legacy devices, applications, and operating systems. The reasons why they refer to tuplespaces as the underlying communications model resembles all the requirements for web services that should enable fully flexible and open eWork and eCommerce. The following is a list of some of the requirements mentioned by Johanson and Fox [Johanson & Fox, 2004]: limited temporal decoupling, referential decoupling, extensibility, expressiveness, simple and portable APIs, easy debugging, scalability, and failure tolerance and recovery.

In side remarks [Johanson & Fox, 2004] also report shortcomings of current tuplespace models. They lack the means to name spaces, semantics, and structure in describing the information content of the tuples. The tuplespace provides a flat and simple data model that does not provide nesting, therefore, tuples with the same number of fields and field order, but different semantics, cannot be distinguished. Instead of following their ad-hoc repairs we propose a simple and promising solution for this. We propose to refine the tuplespace into a *triple space, where <subject, predicate, object> describe content and semantics of information*. The object can become a subject in a new triple thus defining a *graph structure* capturing structural information.

Fortunately with RDF[15] (cf. [Klyne & Carroll, 2004]) this space already exists and provides a natural link from the space-based computing paradigm into the semantic web. Notice that the semantic web is not made unnecessary based on the tuple-spaced paradigm. The global space can help to overcome heterogeneity in communication and cooperation, however, it does not provide any answer to data and information heterogeneity. In fact, this aspect is what the semantic web is all about.

4.2 Triple-spaced computing

The web and the tuplespace have many things in common. They are both global information spaces for persistent publication. Therefore, they share many of the same underlying principles. They differ in their application context. The web is a world wide information space for the human reader and the tuplespace is a local space for parallel processes in an application. Thus, the web adds some features that are currently lacking in the tuplespace.

First, with URIs the web provides a well-defined *reference mechanism* that has world-wide scalability to address chunks of information. Tuplespaces lack this mechanism since they were designed mostly for closed and local environments. Johanson and Fox

15. http://www.w3.org/RDF/

[Johanson & Fox, 2004] already reported this as a bottleneck when applied in their setting of heterogeneity and dynamic change.

Second, the namespace mechanism of the web allows different applications to use the same vocabulary without blurring their communications. Namespaces help to keep the intended information coverage of identifiers separate even if they are named equally. Namespaces provides a well-defined *separation mechanism* that scales on a world-wide scale to distinguish chunks of information.

Third, the web is an information space for humans and the tuplespace is an information space for computers, however, the *semantic* web is for machines too. It provides standards to represent machine-processable semantics of data. We already mentioned RDF that provides nested triples as a data model to represent data and their formal semantics on the web. This enables applications to publish and to access information in a machine processable manner. RDF Schema [Brickley & Guha, 2004] defines classes, properties, domain and range restrictions, and hierarchies of classes and properties on top of RDF. Thus, a richer data model than nested triples can be used to model and retrieve information. This gets even further extended by OWL [McGuinness & van Harmelen, 2004], a data modeling language based on description logic.

Therefore, the semantic web has the true potential to become the global space for application integration, like the tuplespace became a means for the local integration of parallel processes. It provides the means for global integration with the inherent complexity stemming from information heterogeneity and dynamic changes. As with tuplespace, it makes problems with protocol and process heterogeneity transparent, by it's uniform and simple means for accessing and retrieving information. Complex message exchange is replaced by simple read and write operations in a global space.

Having said this, it is also clear that this is not the end but just the beginning of an exercise. No application can quickly check the entire semantic web to find an interesting triple. Conversely, no application would simply publish a triple and then wait forever until another application picks it up. Clever middleware is required that provides a virtual global triplespace without requesting each application either to download or to search through the entire semantic web.[16] The triplespace needs to be divided up to provide security and privacy features as well as scalability. However, none of these requirements are really new. They apply to any application that deals with the web on a global scale.

5 Conclusions

Johanson and Fox [Johanson & Fox, 2004] expect ubiquitous computing as the "killer app" for tuple-space based computing because of the model's portability, extensibility, flexibility, and ability to deal with heterogeneous environments. Actually, truly web-service enabled eWork and eCommerce shares many, if not all of the features of ubiquitous computing. In fact, we think that a tuplespace-based communication model is close in spirit to the web and may help to bring web services to their full potential. It requires moving from a message-oriented communications model into a web where

16. See for example the work of the company Tecco, http://www.tecco.at/en/index.html.

information is published (broadcast) based on a global and persistent URI.

The tuplespace helps to overcome many problems around heterogeneity in information distribution and information access. Since applications are decoupled in reference, time, and space, many issues in protocol and process alignment disappear because they are provided by the underlying middleware that implements the tuplespace. Still, the tuplespace does not contribute anything to the solution of data and information heterogeneity. In fact, there are already ad hoc proposals to add semantics to the data represented in it. Alternatively, this paper proposed a straightforward approach using semantic web technology to provide a well established mechanism for that. It will transfer the tuplespace into an RDF-based triplespace. This triplespace provides the web with the means to exchange data between applications based on machine-processable semantics. Therefore, this triplespace may become the web for machines as the web, based on HTML, became the web for humans.

Acknowledgement.
The paper is simply a synthesis of discussions I had with Tim Berners-Lee and Eva Kühn. It reflects only the private opinion of the author and not the official policy of DERI.

References

[Alonso et al., 2003] G. Alonso, F. Casati, H. Kuno, and V. Machiraju: *Web Services*, Springer, 2003.

[Angerer, 2002] B. Angerer: Space Based Computing: J2EE bekommt Konkurrenz aus dem eigenen Lager, *Datacom*, no 4, 2002.

[Brickley & Guha, 2004] D. Brickley and R.V. Guha (eds.): RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, February 2004, http://www.w3c.org/TR/rdf-schema/

[Fensel, 2003] D. Fensel: *Ontologies: Silverbullet for Knowledge Management and Electronic Commerce*, 2nd edition, Springer, 2003.

[Fensel & Bussler, 2002] D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, *Electronic Commerce Research and Applications*, 1(2), 2002.

[Fielding, 2000] R. T. Fielding: Architectural styles and the design of network-based software architectures, PhD Thesis, University of California, Irvine, 2000.

[Fielding & Taylor, 2002] R. T. Fielding and R. N. Taylor: Principled Design of the Modern Web Architecture, *ACM Transactions on Internet Technology (TOIT)*, 2(2), May 2002:115-150.

[Gerlernter, 1992] D. Gerlernter: *Mirrorworlds*, Oxford University Press, 1992.

[Johanson & Fox, 2004] B. Johanson and A. Fox: Extending Tuplespaces for Coordination in Interactive Workspaces, *Journal of Systems and Software*, 69(3), January 2004:243-266.

[Klyne & Carroll, 2004] G. Klyne and J. J. Carroll (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, February 2004, http://www.w3.org/TR/rdf-concepts/

[McGuinness & van Harmelen, 2004] D L. McGuinness and F. van Harmelen (eds.): OWL Web Ontology Language: Overview, W3C Recommendation, February 2004, http://www.w3c.org/TR/owl-features/

[zur Muehlen et al., 2004] M. zur Muehlen, J. V. Nickerson, and K. D. Swenson: *Developing Web Services Choreography Standards - The Case of REST vs. SOAP*, Decision Support Systems, 37, 2004.