Towards a tuplespace-based middleware for the Semantic Web

Robert Tolksdorf, Elena Paslaru Bontas, Lyndon J. B. Nixon
Freie Universität Berlin, Institut für Informatik
Takustr. 9 14195 Berlin, Germany tolk, paslaru, nixon@inf.fu-berlin.de

Abstract

The realization of the Semantic Web needs a set of specialized middleware as its infrastructure. In this paper we describe the principles of tuplespace computing, explain why tuplespaces are a suitable middleware for the Semantic Web, envision "Semantic Web Spaces" [1] *and outline how our tuplespace platform XMLSpaces can be extended to support Semantic Web technologies, like RDF(S) and OWL.*

1. Introduction

For the realization of real world Semantic Web-based systems, one needs powerful middleware technologies to cope with the requirements of reliability, scalability, self-organization and co-ordination which are inherent in the open distributed nature of the Web.

The Semantic Web[2] envisions a distributed network of machine-understandable knowledge. As it is built upon the existing Web infrastructure it inherits the Web architectural model, which has been formally described as REST (Representational State Transfer)[18]. The fundamental principle of the REST architecture is that resources are *stateless* and *published* based on a *global and persistent URI*. The Semantic Web extends this world of URIs from addressable resources to any abstract concepts represented within a computer system. It is built upon open standards for the representation of Web-based knowledge in a machine-readable manner (RDF, OWL) as well as the realization of semantically enriched Web Services[17, 33, 36, 40].

Tuplespaces have application to the Web in that they realize global places where information can be *published* and *persistently stored*. They have advantages over the standard client-server model in cases of parallel processing of published information from heterogeneous sources. Tuplespaces have been used to explore application to open distributed systems, multi-user and workflow co-ordination, XML middleware and self-organization[7, 11, 12]. All of these areas are relevant to Web-based systems.

1 This work is partially supported by the EU Network of Excellence KnowledgeWeb (FP6-507482)

In this paper we describe the evolution of tuplespace computing, explain how tuplespaces are a suitable middleware for the Semantic Web, suggest how Semantic Web Spaces could be envisioned and describe its realization as an extension of the XMLSpaces platform[12].

2. Tuplespace Computing

The aim of this section is to introduce the notion of tuplespace computing and present several extensions to the classical Linda-based systems that motivate the use of semantically enriched tuplespaces in the context of the Semantic Web. Since the Web and the Semantic Web are confronted with the same problems of heterogeneity, dynamics or scalability as open distributed systems, the different flavors of Linda-based systems can give an impression about how tuplespaces can be integrated with the Semantic Web.

2.1. The coordination language Linda

The coordination language Linda[19] has its origins in parallel computing and was developed as a means to inject the capability of concurrent programming into sequential programming languages. It consists of coordination operations (*the coordination primitives*) and a shared data space (*the tuplespace*) which contains data (*the tuples*).

The tuplespace is a shared data space which acts as an associative memory for a group of agents. The coordination primitives are a small yet elegant set of operations that permit agents to emit a tuple into the tuplespace (operation *out*) or associatively retrieve tuples from the tuplespace either removing those tuples from the space (operation *in*) or not (operation *rd*). A tuple is an ordered list of typed field. Retrieval is governed by matching tuples against a template - a tuple which contains both literals and typed variables. A match occurs when the template and the tuples are of the same length, the field types are the same and the value of constant fields are identical. For example the tuple *("N70241", EUR, 22.14)* will match the template *("N70241", ?currency, ?amount)* as long as the variables share the same field types. Both retrieval operations (i.e. *in* and *rd*) are blocking: they return only when a matching tuple is found[2]. Linda combines synchronization and communication in a simple model with a high level of abstraction.

2.2. Linda extensions

Aside from the theoretical foundation of Linda by a variety of formal models, attention has been paid to applying Linda to different areas aside from parallel computing. We distinguish two categories of extensions of the original approach (based on [38]). Approaches proposing **new types of tuplespaces** aim to overcome the technical problems of dynamic, open, distributed systems (e.g. heterogeneity, scalability, fault-tolerance, coordination of multiuser access), by proposing distribution strategies and various tuplespace structures such as multiple spaces, hierarchical spaces and naming spaces[11, 44, 30, 7, 35]. A second research direction extends the primitive set of field types (usually those of the host implementation language) with **new types of tuples** to support more complex data structures as tuple field values, introduce flexibility through rules, event models and logic-based approaches, and define **new coordination primitives** to support additional operations as well as **new matching mechanisms** [12, 42, 32, 41].

3. From tuplespaces to Semantic Web Spaces

The Linda model designed for parallel programming has been extended into a very flexible concept that has successfully been applied to various technical domains, which face similar problems as the emerging Semantic Web. [38] mentions the following features of Linda as attractive for programming open distributed applications:

- It uncouples interacting processes both in space and in time.
- It permits associative addressing: data is accessed by its content, not by knowing its reference.
- It supports asynchrony and concurrency as an intrinsic part of the tuplespace abstraction.
- It separates the coordination implementation from characteristics of the host platform or programming language.

Semantic Web-based systems need to access knowledge stores distributed on the Web to acquire and infer knowledge to realize specific tasks. In consequence these knowledge stores must handle parallel access from multiple, heterogeneous systems and must coordinate responses with other systems (e.g. that resolve ontological mismatches). Additionally Web Services would better communicate using the traditional publication-based communication paradigm of the Web in order to spatially and temporally decouple messages that are to be exchanged to accomplish a common activity. From this basis Semantic Web Services[28], an application of Semantic Web technologies to Web Services, could become a reality[15]. Multi-Agent Systems or Grid Computing, areas which have been identified as being complementary to Semantic Web and Semantic Web Services[24, 49, 20, 4, 21, 8, 14], could also benefit from a tuplespace-based middleware.

Applying tuplespaces to the open global environment of the Web raises however new requirements, some of which have already been mentioned in other work[15, 26]:

1. A reference mechanism. The Web uses URIs as a global mechanism to uniquely address resources.
2. Richer tuple typing than just the core data types. Richer typing can support validation and correct interaction with tuplespaces.
3. Tuple nesting. Data models such as XML and RDF/XML permit the nesting of elements within a single document. Likewise Web-based information should be able to explicitly show where one unit is contained within another.
4. A separation mechanism. On the Web, vocabularies containing the same terms can be kept separate using the namespaces mechanism.

After an initial proposal for a Semantic Web Space [46], we analyzed these requirements further on the basis of a Semantic Web-enhanced traffic management use case[47] in the domain of Multi-Agent Systems, as only a platform meeting the requirements arising from concrete use cases can offer a powerful middleware solution for the problems of data and process heterogeneity on the Semantic Web.

3.1. New types of tuplespaces

Just as the approach in XMLSpaces[45] represents the content and structure of an XML-enabled tuplespace as a single XML tree, the structure of a Semantic Web Space is represented explicitly by means of an *ontology* (model of the tuplespace, see Figure 1). The ontology describes the typical components of the tuplespace, such as sub-spaces, supported tuple types and matching templates, and coordinates the information access. It can easily be extended with new types of tuplespaces, rules describing new primitives and access policies, and metadata about spaces and allows the usage of automatic reasoning in the management of tuples and accessing agents. The ontological model of the tuplespace references the concrete spaces and triples published by different parties, which are syntactically RDF documents and (sets of) RDF statements, respectively. Representing the entire tuplespace as an ontology model offers a means to reference tuples and tuplespaces, which can be identified and addressed using URIs – by definition assigned to named Semantic Web resources.

Besides providing a mechanism to address and identify spaces and their contents, Semantic Web Spaces must reflect the open and distributed nature of the Web, requiring a

2 Besides, the retrieval operations of the original Linda model do not return the complete result set, but simply the first matching tuple.

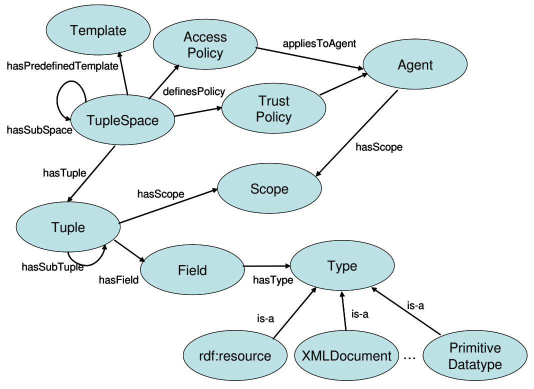

Figure 1. Model of a tuplespace

distributed approach to tuplespaces which can support stability and scalability at a very high magnitude. A decentralized architecture of Semantic Web Spaces involves supporting the virtual integration of distributed storage of tuples into a single view on the tuplespaces.

Two possible distribution strategies are named spaces that exist in parallel (*multiple spaces*) and *nested spaces*. In the former case the co-ordination primitives must specify explicitly which tuplespace they wish to operate upon. In the latter, operations need to be extended to emit and retrieve complete tuplespaces: The view on tuplespaces includes those tuples found in child tuplespaces while those in parent tuplespaces would be hidden. Multiple tuplespaces strictly partition the global tuplespace and shield the views on the space from each other. Similarly, nested tuplespaces make information available in exactly one space at a time. The hierarchical structure indicates some inclusion relation. This is, however, not true when nested spaces are viewed as first class objects. In that case, the tuples contained are *not* considered and do not appear in their "parent"-space. We conclude that both options are *unsuited for our purposes*.

Merrick and Wood have put forward a further alternative for structuring tuplespaces in their Scopes-approach[31]: A tuple can be part of multiple so-called scopes. A scope is a partial view on the complete tuplespace. That view contains sets of tuples on which the usual tuplespace operations can be applied. Every tuple is identified by a name. Formally, a scope is seen as a set of sets of names and can therefore be constructed from names. It is written $[ab, b, de]$ as an abbreviation for $\{\{a, b\}, \{b\}, \{d, e\}\}$. From a name x, a scope $[x]$ can be constructed. Scopes are defined to *match* whenever they have an element – which is a set of names – in common. So $[ab]$ matches the above scope, while $[d, e]$ does not, since the scope $[d, e]$ is not equivalent to $[de]$. Aside from constructing scopes, a set of operations is defined for combining or intersecting scopes and the sets they contain.

For Semantic Web Spaces we reinterpret the concept of scopes as a structuring mechanism of tuplespaces to devise a structure of *context* in which a client operates on a set of RDF statements. This allows for the integrated view of the contained data (as tuples) to be contextualized to who is accessing the tuplespace and under which circumstances this access occurs (thus protecting data that may not be intended to be universally accessible).

As in common Linda-like systems Semantic Web Spaces may use different persistent storage paradigms, including for example replication and distribution of the RDF data on several physical systems such as [3, 22].

3.2. New types of tuples

Semantically enriched tuplespaces require *new types of tuples* with defined semantics. Semantic Web Spaces focuses on tuples containing information in standard representation languages of the Semantic Web (i.e. RDF(S)[23], OWL[37], SWRL[25]).

RDF statements are the core model of Semantic Web knowledge representation. They are represented by triples with the structure *(subject, predicate, object)*. To support richer typing, tuples in Semantic Web Spaces are stored as an ordered list of typed field values. These types can be URIs identifying RDF classes. The triple form is also extended with an ID field, providing Linda primitive-based access to the tuple identifier. Thus RDF tuples are typed *(?rdfs:Resource, ?rdf:Property, ?rdfs:Resource,?rdf:ID)* [3].

By re-using the object of one statement as the subject of another, the tuplespace can build a semantic graph structure of the contained knowledge. There are three particular modeling constraints in RDF that need to be handled specifically in the tuplespace platform: 1). **blank nodes**, 2). **containers and collections**, and 3). **reified statements**.

Blank nodes are nodes in the RDF graph which are not identified by global identifiers. We propose a class in the tuplespace ontology representing the class of BlankNodes, which carry an internal tuplespace-unique value for representing each instance, playing the role of "blank node identifier". An application receiving an instance of BlankNode should know that the identifier is only to be considered as local to the query. To support operations where tuples contain blank nodes, we permit subspaces, and not just single tuples, to be added to, matched in or removed from the Semantic Web Space.

Containers and collections are special RDF objects which represent a set of resources. We propose that the rdfs:Container typed resources (rdf:Bag, rdf:Alt, rdf:Seq) are represented in a tuple by a resizable array datatype, and that the members of a collection (rdf:List) are represented by a closed array datatype. In the tuplespace the container or collection can be referenced by an URI, and accessed ei-

3 The object of a RDF triple may be a new resource or a value, a literal. The ID field should be understood not as the ID of the RDF statement, but of the tuple making that statement

ther as the entire array (and processed further at the client) or through the RDF membership properties.

Reification is a means to reference RDF statements. For this purpose, RDF uses its own vocabulary (rdf:Statement, rdf:subject, rdf:predicate, rdf:object) to assign URIs to specific statements so that they can be reused as a subject or object of another statement. A reified tuple is then built up of a set of tuples linking a rdf:Statement instance to the reified tuples subject, predicate and object. The rdf:Statement instance can then be used as the subject or object of another tuple. To simplify operations, a tuple can be included in a tuple field and typed as rdf:Statement in order to represent a reification. We do not consider the tuple identifier as reification as it refers to a tuple in a tuple space and not the RDF statement which is encoded in the tuple.

OWL support in Semantic Web Spaces is a straightforward task if we consider the syntactical compatibility of RDF(S) and OWL. OWL restrictions are represented using blank nodes. owl:unionOf, owl:oneOf, owl:intersectionOf and owl:AllDifferent require Collections. The semantic differences between OWL and RDF Schema play a role only for the matching algorithms, which can use reasoners with specific capabilities.

3.3. New co-ordination primitives

Moreover, transferring the original Linda to the world of Semantic Web requires further work on the conceptional level of the Linda co-ordination model.

While common Linda systems deal with *data* represented as tuples, Semantic Web Spaces are intended to manage also *information* with formally defined semantics represented in triple form (i.e. *(subject, predicate, object)*). These are called RDF tuples. Such tuples also carry a truth assignment which makes them different from being merely interpreted as data. Changing the scope of the tuplespaces from data to information affects the Linda operations involved. Hence the classical semantics of Linda must be re-evaluated for tuplespaces of truth-assigned content.

According to this, we consider that there are two views on the tuplespace in Semantic Web Spaces. *The information view* interprets the data from the space according to the semantics of the information it encodes. In this view, Semantic Web Spaces define additional primitives with their own semantics. *claim* inserts a RDF tuple into the tuplespace if and only if it is satisfiable according to the defining RDF Schema. *retract* removes a RDF statement from the tuplespace. However, following the reasoning that an asserted truth can not be un-asserted, a *retract* operation does not imply a deletion of the corresponding tuple from the space. Rather, the statement itself is 'lost'—as in, it can not be retrieved as a result of matching operations— but the reference to the statement is retained [4] *endorse* and *excerpt* read a single matching tuple or all matching tuples respectively from the information view (i.e. only those which were claimed and found satisfiable).

The *data view* of Semantic Web Spaces preserves the operations *out*, *in* and *rd* of Linda. It is also extended with operations which provide simpler type checking for RDF tuples, without considering RDF semantics or extended matching (see below). These operations *outr*, *inr* and *rdr* handle only values of type *(subject, predicate, object)*.

Matching relations are also extended to work on RDF typing and are able to take into account defined RDF(S) semantics, for example to match a sub-relation in a tuple for a relation in a template (see Sections 3.4 and 4.2).

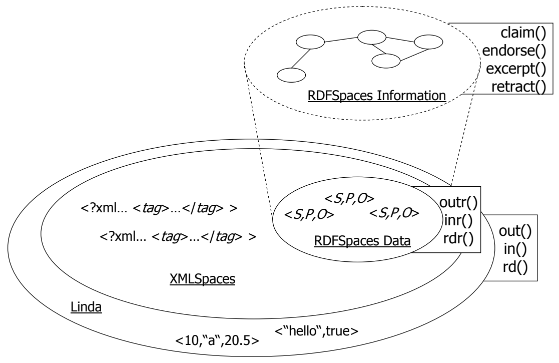

Figure 2. Views in Semantic Web Spaces

Figure 2 shows the structure of Semantic Web Spaces. We illustrate the RDFSpace – further work could extend this for OWLSpace, RulesSpace and so on. It shows both a data view, encompassing simple datatype tuples, XMLSpaces tuples (containing XML documents[12, 45]) and Semantic Web tuples such as RDF triples, and an information view, where RDF tuples are handled according to the semantics of the information that they contain and reasoning features are embedded (RDFS, OWL Lite, OWL DL etc.).

3.4. Semantic matching

Semantic Web Services, or any application built upon the use of Semantic Web Spaces, are made possible as a result of appropriate matching mechanisms. Matching through templates is the fundamental interaction paradigm of tu-

4 Deleting a tuple from a tuplespace is still possible at the data level.

plespaces. New matching procedures are required to support the RDF-based paradigm of the Semantic Web Space which includes matching both on values and on types. As well as resource matching (equivalence of simple datatypes but also of complex datatypes (arrays, lists) and of URIs), matching procedures must also take into account ontological information and different levels of precision matching made available to allow requesters to choose the bounds in which a template may be determined as valid for a tuple.

For example, a query on who owns any type of animal using the template *(?, ex:owns, ?ex:Animal)* means match on tuples whose subject (of any type) has the predicate *ex:owns* and the object of type Animal. Given a tuple *(X, ex:owns, Z)* where Z is of type Dog and the RDF Schema information that Dog is a subclass of Animal, we expect this tuple to match. However, matching will only succeed if the matcher is aware of the RDF Schema information, and the procedure selected allows matching on subclasses. The matching functionality for clients will need to support simple templates as well as queries in an expressive RDF query language which could be mapped into tuplespace templates. Relationships between templates and nesting of templates need to be expressible (e.g. find tuples in which resource R is an object in one statement and the subject in another, or find tuples where resource R is the object of a statement whose subject is the result of another template).

Blank nodes are returned within Subspaces. This avoids the problem that a blank node identifier within a query result is local to the query, i.e. it can not be used by the client to retrieve further tuples containing this blank node. Rather, where the matched tuple has a blank node as subject or object, the connecting tuples in the RDF graph are returned along with it, so that the client can access a RDF subgraph with all tuples related to the selected blank nodes.

Reified statements could be handled by two different levels of matching, one which considers the reification in terms of RDF statements of subject, property and object, and the other which retrieves a nested tuple from a tuple field.

OWL primitives in the data view are handled in the same way as RDF(S), while a pre-defined set of templates operating on typical OWL syntax constructs, could be used to make this matching task easier. The semantical interpretation of OWL tuples (i.e. in the information view) is handled by embedding available DL reasoners.

4. Implementation of Semantic Web Spaces

We are developing “RDFSpaces”, a first version of Semantic Web Spaces, on the basis of XMLSpaces [12, 45]. Since XMLSpaces already implements much of the functionality of a distributed tuplespace, we extended it to support the management of RDF triples and a core set of RDF(S)-specific matching templates.

XMLSpaces is a distributed coordination platform – implemented in Java and refined as XMLSpaces.NET in C# on the .NET platform – that extends the Linda coordination language with the ability to carry XML documents in tuple fields. XMLSpaces defines an extensible set of matching relations, including those given by XML query languages, as a hierarchy of matching relations on tuples and an open set of matching amongst data, documents and objects. The complete tuplespace is represented in XML form as a tuple tree containing tuples and nested spaces. This allows a more flexible management of the coordination medium. XMLSpaces supports the open distribution of resources in order to support the co-ordination of wide area applications and the integration of multiple servers into a single logical data space. Its distribution strategy can be configured at startup, and current implementations include both full and partial replication.

4.1. Implementing new types of tuples

While richer data typing has been abstracted to the XML document level, XMLSpaces permits the use of objects from the implementation programming language as tuple values. Object serialization and matching is object class specific, where the object class and its properties and relationships are defined within the programming code. A similar approach is taken for the use of conceptual instances, where class definitions are resolved from an ontology and used for serialization and matching. The Semantic Web Space has a view on the tuplespace not as a XML DOM but as a RDF graph (e.g. in a Java implementation a switch from JAXP to Jena). The reading and writing of tuples is supported by the transformation to and from raw RDF (as XML serialization, N3 triples, etc.) within the platform to and from sets of tuples. Namespace and URI support are already required by the XMLSpaces platform in allowing well-formed and valid XML to be the value of tuple fields[5].

RDFSpaces introduces a new type of tuples containing four ordered fields. The field types are constrained to being valid RDF types (rdfs:Resource as subject and object, rdf:Property as predicate, rdf:ID for the identifier). The implementation is extended to include RDF Classes and BlankNodes as field types and as well as programming language datatypes for Containers and Collections.

4.2. Implementing semantic matching

Experience with matching relations has already been had through XMLSpaces, which required a variety of matching relations on XML documents: matching on a shared schema, on a minimal subset of the schema, on equal contents, on equality of attributes in elements, or on a query expression formalized in a common XML query language.

5 URIs are datatypes in the XML Schema and XML Namespaces are supported in the DOM Level 2 object model.

Some of these matching procedures needed the interpretation of external schema to determine a match. Likewise Semantic Web Spaces should offer matching procedures with different levels of precision and ontological reasoning, which might require the interpretation of an ontology.

RDFSpaces currently implements a core set of matching functions for RDF and RDF(S), which rely exclusively on the RDF(S) data model and can be easily extended if application-specific matchings are required. While classical Linda implementations - including XMLSpaces - allow a sequential retrieval of the space content (i.e. they retrieve the first tuple matching a query), Semantic Web Spaces adds a retrieval operation that retrieves the complete set of matching tuples corresponding to a given query/template. For this operation, we follow the *copy-collect* primitive defined by Rowstron and Wood [39]. It works within scopes [31]) - a scope is created by the system into which all matching tuples are copied. A reference to this scope is passed to the client who is given alone the right to access the scope. The client can then make normal destructive reads in that scope to remove all of the tuples. When the scope is empty the system destroys the context.

Rather than support a complex RDF query structure, Semantic Web Spaces provide simple template-based matching. Clients can use matchings to build local sub-graphs of relevant information and use RDF query engines for more complex processing of those sub-graphs.

5. Related and future work

There have been numerous extensions to the original Linda (see [38] for a complete discussion), such as JavaSpaces [41], TSpaces by IBM [48] and our work on service-based [43], Web-based [10] and self-organising [9] Linda.

None of these extensions have attempted to deal with semantics as in our approach. Little work currently exists in this field. sTuples [27] attempts to combine Semantic Web and tuplespace technology by extending JavaSpaces to permit a tuple field to contain an OWL individual. Triple Spaces [16] have been proposed as a communication mechanism for the WSMX Semantic Web Service platform [6] while we see our Semantic Web Space as a generic middleware for the Semantic Web. In both approaches it is unclear what consideration has been made of how storing RDF/OWL into a tuple space affects the fundamental issues of tuple and tuplespace representation, Linda operations and matching. These issues are specifically handled in our Semantic Web Space. A minimal architecture for Triple Spaces is also proposed [5], in which however many of the powerful and beneficial aspects of Linda and the tuplespace model have been removed.

Most work seeking to offer a middleware for activity on the Semantic Web to date has focused on peer-to-peer (P2P) networks. The P2P approach introduces robustness and flexibility at the expense of efficiency[29]. Hybrid approaches like Edutella[34] and PEPSINT[13] use a super-peer as mediator for the query translation from peer to peer based on a global ontology, becoming more efficient at the price of loss of autonomy[1]. In comparison to query reformulation in P2P networks, tuplespaces support a single template for retrieval and carry out matches based on resolving internally defined matching rules. The communication overhead is lower than in P2P networks as tuplespace synchronization is only necessary by addition or deletion of tuples and not non-destructive retrieval (read operations).

Our work will be taken forward in that we evaluate the mentioned Semantic Web Space implementation for RDF data (RDFSpaces) and seek additional extensions with a focus on scalability issues. Based on our earlier work on tuplespaces for interactive applications and for XML document management, we expect positive results from performance tests, which will be an important contribution for Semantic Web applications, an area in which questions of scalability and efficiency are still open. Such a semantics-aware space will be of value to a wide range of application domains, such as data integration, multi-agent communication and Semantic Web Services, as a powerful and flexible basis for enabling asynchronous large-scale collaboration on shared tasks on distributed, heterogeneous data.

References

[1] S. Bergamaschi, F. Guerra, and M. Vincini. A peer-to-peer information system for the semantic web. In *Proceedings of the AP2PC*, 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 5 2001.

[3] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *The Semantic Web - ISWC2002*, 2002.

[4] P. Buhler and J. M. Vidal. Semantic Web Services as Agent Behaviors. In *Agentcities: Challenges in Open Agent Environments*, pages 25–31. Springer-Verlag, 2003.

[5] C. Bussler. A minimal triple space computing architecture. In *Proceedings of the WIW 2005 Workshop on WSMO Implementations*, volume 134. CEUR Workshop Proceedings, 2005.

[6] C. Bussler, E. Kilgarriff, R. Krummenacher, F. Martin-Recuerda, I. Toma, and B. Sapkota. D21. v0.1 WSMX Triple-Space Computing. http://wsmo.org/TR/d21/v0.1, June 2005.

[7] G. Cabri, L. Leonardi, and F. Zambonelli. MARS: a programmable coordination architecture for mobile agents. *IEEE Internet Computing*, 4(4):26–35, 2000.

[8] M. Cannataro and D. Talia. The Knowledge Grid. *CACM*, 46(1):89–93, 2003.

[9] A. Charles, R. Menezes, and R. Tolksdorf. On the Implementation of SwarmLinda. In *Proceedings of the ACM Southeastern Conference 04*, pages 296–297, 2004.

[10] P. Ciancarini, A. Knoche, D. Rossi, and R. Tolksdorf. Redesigning the Web: From Passive Pages to Coordinated Agents in PageSpaces. In *ISADS97*, pages 377–384, 1997.

[11] P. Ciancarini, A. Knoche, R. Tolksdorf, and F. Vitali. PageSpace: An Architecture to Coordinate Distributed Applications on the Web. *Computer Networks and ISDN Systems*, 28(7–11):941–952, 1996.

[12] P. Ciancarini, R. Tolksdorf, and F. Zambonelli. Coordination Middleware for XML-centric Applications. In *Proceedings of ACM SAC 2002*, pages 335–343, 2002.

[13] I. F. Cruz, H. Xiao, and F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. In *Third International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*.

[14] D. De Roure and J. Hendler. E-Science: the Grid and the Semantic Web. *IEEE Intelligent Systems*, 19(1):65–71, 2004.

[15] D. Fensel. Triple-based Computing - WSMO Working Draft. http://www.wsmo.org/2004/tp-computing/, June 2004.

[16] D. Fensel. Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In *INTELLCOMM*, pages 43–53, 2004.

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

[18] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California Irvine, 2000.

[19] D. Gelernter and N. Carriero. Coordination languages and their significance. *Commun. ACM*, 35(2):97–107, 1992.

[20] N. Gibbins, S. Harris, and N. Shadbolt. Agent-based Semantic Web Services. In *Proceedings of the 12th International Conference on World Wide Web WWW03*, 2003.

[21] C. Goble and D. D. Roure. The Semantic Grid: Myth Busting and Bridge Building. In *ECAI-2004*, 2004.

[22] S. Harris and N. Gibbins. 3store:Efficient Bulk RDF Storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems*, 2003.

[23] P. Hayes and B. McBride. RDF Semantics. *Available at http://www.w3.org/TR/rdf-mt/*, 2004.

[24] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2), 2001.

[25] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. 2004.

[26] B. Johanson and A. Fox. Extending tuplespaces for coordination in interactive workspaces. *Journal of Systems and Software*, 69(3):243–266, 2004.

[27] D. Khushraj, O. Lassila, and T. W. Finin. sTuples: Semantic Tuple Spaces. In *MobiQuitous*, pages 268–277, 2004.

[28] R. Lara, H. Lausen, S. Arroyo, J. de Bruijn, and D. Fensel. Semantic Web Services: description requirements and current technologies. In *Proceedings of the ICEC'03*, Pittsburgh, PA, 2003.

[29] A. Lopatenko and B. Matthews. P2P data integration for Current Research Information Systems. In *Proceedings of the CRIS2004 Conference*, 2004.

[30] R. Menezes and R. Tolksdorf. Adaptiveness in Linda-based Coordination Models. In *Proceedings of ESOA 2003*, 2004.

[31] I. Merrick and A. Wood. Coordination with scopes. In *Proceedings of SAC'00*, pages 210–217. ACM Press, 2000.

[32] N. Minsky, Y. Minsky, and V. Ungureanu. Making Tuple Spaces Safer for Heterogeneous Distributed Systems. In *Proceedings of SAC'00*, COMO, Italy, March 2000.

[33] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: A Framework and Infrastructure for Semantic Web Services. http://www.cs.unibo.it/ gaspari/www/iswc03.pdf, 2003.

[34] W. Nejdl, B. Wolf, S. Staab, and J. Tane. EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network, 2001.

[35] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.

[36] OWL Services Coalition. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf, November 2003.

[37] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. *Available at http://www.w3.org/TR/owl-absyn/*, 2004.

[38] D. Rossi, G. Cabri, and E. Denti. Tuple-based technologies for coordination. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 4, pages 83–109. Springer Verlag, 2001. ISBN 3540416137.

[39] A. I. T. Rowstron and A. M. Wood. Solving the Linda multiple rd problem using the copy-collect primitive. *Sci. Comput. Program.*, 31(2-3):335–358, 1998.

[40] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding Semantics to Web Services Standards. In *Proceedings of the International Conference on Web Services (ICWS'03), 2003.*

[41] Sun Microsystems, Inc. *JavaSpace Specification, Revision 0.4*, 1997.

[42] G. Sutcliffe and J. Pinakis. Prolog-D-Linda: An Embedding of Linda in SICStus Prolog. Technical Report 91/7, The University of Western Australia, Department of Computer Science, 1991.

[43] R. Tolksdorf. Laura: A coordination language for open distributed systems. Technical Report 1992/35, Technische Universität Berlin, Fachbereich 20 Informatik, 1992.

[44] R. Tolksdorf. Laura—A service-based coordination language. *Science of Computer Programming*, 31(2–3):359–381, July 1998.

[45] R. Tolksdorf, F. Liebsch, and D. M. Nguyen. XMLSpaces.NET: An Extensible Tuplespace as XML Middleware. In *Proceedings of .NET Technologies'2004*, 2004.

[46] R. Tolksdorf, L. Nixon, D. M. Nguyen, F. Liebsch, and E. Paslaru Bontas. Semantic Web Spaces. Technical Report TR-B-04-11, Free University of Berlin, July 2004.

[47] R. Tolksdorf, L. Nixon, E. Paslaru Bontas, D. M. Nguyen, and F. Liebsch. Enabling real world Semantic Web applications through a coordination middleware. In *Proceedings of ESWC'05*. Springer Verlag, 2005.

[48] P. Wyckoff, S. McLaughry, T. Lehman, and D. Ford. T spaces. *IBM Systems Journal*, 37(3):454–474, 1998.

[49] Y. Zou, T. Finin, L. Ding, H. Chen, and R. Pan. Using Semantic Web technology in Multi-Agent systems: a Case Study in the TAGA Trading Agent Environment. *Proceeding of the 5th International Conference on Electronic Commerce*, September 2003.