*The Knowledge Engineering Review*, Vol. 23:2, 181–212. © 2007, Cambridge University Press
doi:10.1017/S0269888907001221 Printed in the United Kingdom

# Tuplespace-based computing for the Semantic Web: a survey of the state-of-the-art

LYNDON J. B. NIXON[1], ELENA SIMPERL[2], RETO KRUMMENACHER[2] and FRANCISCO MARTIN-RECUERDA[3]

[1]*Freie Universität Berlin, Berlin, Germany;*
*e-mail: nixon@inf.fu-berlin.de;*
[2]*Digital Enterprise Research Institute (DERI), Universität Innsbruck, Innsbruck, Austria;*
*e-mail: elena.simperl,reto.krummenacher@deri.at;*
[3]*Information Management Group (IMG), University of Manchester, Kilburn Building, Manchester, UK;*
*e-mail: fmartin-recuerda@cs.man.ac.uk*

**Abstract**

Semantic technologies promise to solve many challenging problems of the present Web applications. As they achieve a feasible level of maturity, they become increasingly accepted in various business settings at enterprise level. By contrast, their usability in open environments such as the Web—with respect to issues such as *scalability*, *dynamism* and *openness*—still requires additional investigation. In particular, Semantic Web services have inherited the Web service communication model, which is primarily based on synchronous message exchange technology such as remote procedure call (RPC), thus being incompatible with the REST (REpresentational State Transfer) architectural model of the Web. Recent advances in the field of middleware propose *'semantic tuplespace computing'* as an instrument for coping with this situation. Arguing that truly Web-compliant Web service communication should be based, analogously to the conventional Web, on shared access to persistently published data instead of message passing, space-based middleware introduces a *coordination infrastructure* by means of which services can exchange information in a time- and reference-decoupled manner. In this article, we introduce the most important approaches in this newly emerging field. Our objective is to analyze and compare the solutions proposed so far, thus giving an account of the current state-of-the-art, and identifying new directions of research and development.

## 1 Introduction

The World Wide Web has caused a fundamental shift in the way we access information and services. However, the current Web is aimed mostly at people—information can be found in pages written in natural language and the functionality of many existing Web services is described in human-readable form. The drawbacks of this situation become evident for instance when searching for precise information in a Website. In this case, the most sophisticated information extraction and page-indexing techniques provided by current search engines still require exhaustive manual post-processing. Furthermore, current Web service implementations, including recent approaches in the areas of Web service discovery and composition, quickly reach their boundaries when complementary Web services have to be combined for joint usage, even in a semi-automatic manner.

The Web of the next generation, introduced by Tim Berners-Lee under the name '*Semantic Web*' (Berners-Lee *et al.*, 2001), aims at dealing with such situations by augmenting the current

Web information space with formalized knowledge and structured data, which can be processed by semantics-aware Web services (Lara *et al*., 2003). The Semantic Web promises to solve many challenging and cost-intensive problems of the current generation of Web applications. Ontologies, formalized by the use of Web-suitable representation languages, are shared and reused across platforms in order to allow these to handle the large amounts of heterogeneous data on the Web. Semantic Web services use ontologies and reasoning so as to enable a more flexible or even fully automated cooperation between disparate Web applications.

Current research achievements towards the realization of these revolutionary ideas provide the core building blocks for developing semantic applications in closed environments such as companies' Intranets. This applies for both the knowledge representation and the Web services field, as stated in a study of the Gartner Group on emerging technologies published in July 2006[1]. According to their analysis, the *corporate Semantic Web* is estimated to be a technology trigger, while Web services for organization-internal usage have already reached a feasible level of maturity. In contrast, Web-related aspects—including major concerns such as *scalability*, *dynamism* and *openness*—need further investigation before these technologies can be successfully used on the Web. Recent advances in the area of middleware propose '*semantic tuplespace computing*' as an instrument for coping with this situation.

Arguing that truly Web-compliant Web service communication should be based, analogously to the conventional Web, on shared access to persistently published data instead of message passing, approaches in the aforementioned field introduce a *communication/coordination infrastructure* by means of which services can exchange information in a time- and reference-decoupled manner through a shared virtual data space. This space will be 'semantics-aware', that is, it will also provide functionality for the coordination of and interaction with semantic data such as an appropriate coordination model and implicit reasoning and semantic querying support. Spaces 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 concurrent processing of published information from heterogeneous sources. This has been successfully demonstrated by a wide range of space-based systems applied to solve communication and coordination issues in areas such as open distributed systems, workflow execution, XML middleware and self-organization (Ciancarini *et al*., 1996; Cabri *et al*., 2000; Ciancarini *et al*., 2003). All of these areas are relevant to Web-based, and implicitly to Semantic Web–based, systems.

In this article, we provide an overview of this newly emerging research field. Our objective is to compare and classify the solutions proposed so far, thus giving an account of the current state-of-the-art, and identifying new directions of research and development. Section 2 introduces the notion of *tuplespace computing* and presents several extensions to classical Linda-based systems that motivate the use of semantically enriched tuplespaces in the context of the Semantic Web and Semantic Web services. Sections 3–6 elaborate on the most representative approaches applying tuplespaces as a middleware for application integration and inter-process communication on the Semantic Web. Building upon the results of this analysis, we introduce a framework for structuring and classifying this field, which we term *semantic tuplespace computing*, and align the enumerated systems to the associated criteria. In doing so, we are able to determine the commonalities and differences among the studied approaches, thus identifying the core concepts of this young research field and directions for further research and development (Section 7). Section 8 concludes the article with a discussion of the results of our comparative study and its implications.

## 2 Background knowledge

Coordination in computer systems refers to models, formalisms and mechanisms for describing concurrent and distributed computation (Papadopoulos & Arbab, 1998). Although parallel computing

[1] `http://www.gartner.com/.`

can make much more computation power available and produce significant performance improvements, it raises the problem of coordinating the activities of a large number of concurrently active processes. This has led to the design and implementation of coordination models that seek to formally define how dependencies between parallel activities shall be handled.

In Gelernter & Carriero (1992), coordination is defined as '*the process of building programs by gluing together active pieces*'. Here, the focus is given to the integration of heterogeneous components communicating through different concurrent processes as to produce a virtually unified application, which operates like a single system, abstracted from its distribution, parallelism and internal heterogeneities.

The first coordination language was Linda. Linda has its origins in parallel computing, and was developed as a means to inject the capability of concurrent programming into sequential programming languages (Gelernter, 1985). 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 that acts as an associative memory for a group of agents or clients. 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 preserving the retrieved tuples in the space (operation rd). A tuple is an ordered list of typed *fields*. Retrieval is governed by an associative matching technique: tuples are matched against a tuple *template* specifying the values or the types of a subset of the tuple fields. A match occurs when the template and the tuple are of the same length, the field types are the same, and the values of constant fields are identical. Both retrieval operations are blocking, that is, they return a result only when a matching tuple is found. In this way, Linda combines synchronization and communication in an extremely simple model with a high level of abstraction.

The following features of Linda have been mentioned as attractive for programming open distributed applications (Rossi *et al*., 2001):

- It decouples the interacting processes both in reference and in time. In other words, the producer of a tuple and the consumer of that tuple do not need to know one another's address or exist concurrently.
- Linda permits associative addressing. This means that data are accessed in terms of what kind of data are requested, rather than which specific data are referenced.
- Linda 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.

Linda-based tuplespace systems have been realized in a number of platform implementations. We mention here only some of the principal platforms that have found usage in various coordination projects.

Sun's JavaSpaces is a precursor to the Jini technology, providing a communication middleware for Java (Freeman *et al*., 1999). It realizes an object-oriented version of the Linda model in which tuples encapsulate Java objects, and matching takes into account Java typing, code mobility, reflection and embedded security. It supports distributed caching, publish-subscribe, transactions and load balancing.

IBM's TSpaces is a small footprint Java implementation of a tuplespace server (Wyckoff *et al*., 1998). While comparable to JavaSpaces, TSpaces adds more advanced operator and database functionality so that it can support more complex applications. Besides the extended set of coordination primitives, new operators can be added dynamically. A data management layer provides features similar to relational database systems; it uses indexes for efficient data retrieval and supports queries richer than Linda template matching.

More recently, GigaSpaces, an extension of JavaSpaces, was established to support a tuplespace-based architecture where clients communicate over a Grid network [GigaSpaces(TM)

Technologies Ltd., 2002]. It aims at combining and integrating distributed caching (Data Grid), content-based distributed messaging (Message Grid) and parallel processing (Processing Grid).

These platforms have tended to extend the simple core Linda model in order to increase the expressiveness of the coordination language. For the fulfilment of the tasks for which these platforms are intended, Linda has proven—in its original form—to be insufficient. The aforementioned platforms have, however, also demonstrated that Linda can be extended in particular ways to support new functionalities beyond those it was originally conceived for. We focus on the requirements raised by coordination over open distributed systems, particularly the Web as the most significant representative of such systems at present.

### *2.1 Coordination in open distributed systems and the Web*

Linda was originally conceived for enabling parallel processing in centralized and closed computer network environments. However, the growing importance of open networks combined with the need for concurrency that tuplespaces provide requires the coordination model of Linda to be rethought.

For a Linda implementation, the key difference between open and closed systems is the dynamism that is implied by the former. In an open system clients can join and leave at will, while in a closed system the participating clients are fixed at start-up and do not change. Hence, it is more complex to optimize the coordination between clients as the system cannot know in advance which clients will use the space or which characteristics the active clients have. For example, clients may not necessarily agree in advance on shared models and types for the data placed in tuples. It may become the task of the middleware to (partially) resolve such heterogeneities. For open distributed Linda systems, these twin issues of dynamism and heterogeneity need to be taken into account and are furthermore tightly related to the problem of scalability.

Perhaps, the ultimate open distributed system at present is the World Wide Web. The increased usage of the Web as communication channel for computer systems has led to a growth in the research of Web-style middleware, culminating in the present efforts in the field of Web services.

The Web builds on the principle that data are stored on Web servers that are connected to the global network and accessible by a unique address, known as URL. The principles of interaction have been formalized in the REST model (REpresentational State Transfer) (Fielding, 2000). The standard Web interaction model runs over HTTP, with the commands GET, POST, PUT and DELETE. The parameters for the interaction are passed in the URL or as part of the HTTP message body. Generally, the interaction involves the publication of some data at some URL and then the reading of that data by other clients who know the URL of the published data.

Web services provide standardized descriptions and interfaces as the bases for the execution of programmatic components across the Web. They communicate as a general rule by message exchange, for example, using Simple Object Access Protocol (SOAP) over HTTP. A client wishing to interact with a service forms a message conforming to the description and interface supported by that service and sends it by an appropriate protocol to the service endpoint (e.g. to an URL over HTTP).

However, this message-based interaction pattern requires point-to-point communication and hence breaks if one of the parties becomes unavailable. Furthermore, once a message is delivered, there is not necessarily any retrievable representation of that message available to the communicating partners. Thus, if a set of messages fails, it is not guaranteed that the related messages can be traced back and interpreted.

For any situation that is more complex than a single client executing a single operation on a single Web service, coordination becomes necessary. Typically, a sequence of operations (potentially belonging to different Web services) must be executed in a particular order for a certain goal to be resolvable. The required sequence is usually referred to as *conversation*. The problem is how Web services can make the clients aware of which conversations they support to achieve a particular goal. This is further complicated when interactions involve several services, each of which

**Table 1** Example Web services for a publication-based approach

| |
|---|
| BaseformService: Returns the lemmatized (base) form of the input word<br>http://wortschatz.uni-leipzig.de:8100/axis/services/Baseform?wsdl |
| SentencesService: Returns sample sentences containing the input word<br>http://wortschatz.uni-leipzig.de:8100/axis/services/Sentences?wsdl |
| SynonymsService: Thesaurus which returns synonyms of the input word<br>http://wortschatz.uni-leipzig.de:8100/axis/services/Synonyms?wsdl |
| RightCollocationFinderService: Returns linguistic collocations that occur to the right of the given input word<br>http://wortschatz.uni-leipzig.de:8100/axis/services/RightCollocationFinder?wsdl |
| LeftCollocationFinderService: Returns linguistic collocations that occur to the left of the given input word<br>http://wortschatz.uni-leipzig.de:8100/axis/services/LeftCollocationFinder?wsdl |
| toi-wetter: The weather service of T-Online<br>http://wetter.t-online.de/soap.php?wsdl |

supporting a differently specified conversation. Hence, coordination is typically implemented in a middleware stack, which takes over the responsibility for mediating between the interacting services.

In the Web services standard stack, a number of initiatives look at languages and models to express the coordination between services. WS-coordination, for example, is a meta-framework for implementing specific coordination protocols (Cabrera *et al.*, 2002). It supports the following features:

- Passing of unique identifiers between interacting parties to ensure message routing between conversations.
- Registering Web services that participate in a given conversation.
- Informing a protocol handler which role it plays in a conversation (such as master and slave).

The developed coordination protocols define a set of rules for the conversation between the coordinator and the conversation's participants. For WS-coordination to operate, all potential participants must agree on who will be the coordinator. Distributed coordination is also supported, that is, rather than having a single coordinator, each service can interact with its own coordinator, and the individual coordinators must coordinate among themselves. However, WS-coordination does not define how the coordination protocol itself is described. Furthermore, the Web services architecture does not include a dedicated component for carrying out the interactions between services using the coordination protocol. These components must exist somewhere outside of the Web services and be somehow discovered by all services that wish to participate in a conversation.

Web services could better communicate using the traditional publication-based communication paradigm of the Web, which allows the interactions between services to be decoupled in time and reference (Krummenacher *et al.*, 2005). Given that classical Web services fail to support the persistent availability of messages, as mentioned before, it seems in fact that Web services made a step back compared to the traditional Web. In the World Wide Web, humans publish information persistently for an undefined number of consumers without being occupied by consecutive user requests. Many Web services are implemented by the use of Web Services Description Language (WSDL) (Chinnici *et al.*, 2007) and SOAP (Gudgin *et al.*, 2007), while there is no clear technical motivation for this message-based approach from an information management perspective. In particular, services that provide (semi-)static knowledge could be better implemented using a publication-based approach such as tuplespaces. Some adequate examples are thesauri like the wortschatz-services depicted in Table 1, weather services with static information—in the sense that the weather forecast changes only once to a few times a day—and currency rate or other financial information services as for instance that provided by `www.xignite.com`[2].

[2] The services were discovered with seekda, the Web service Search Engine of aleph-webservices.com.

An additional advantage of the publication-based approach is the fact that the provided information can be further processed and integrated, when stored in the same space, and the implicit knowledge inferred—knowledge that otherwise might not be available from one single Web service. The same positive mashup effect is at the basis of the European Patient Summary proposal presented in Krummenacher *et al*. (2007). The health records of various general practitioners, specialists and hospitals are published to a shared space with the benefit that caregivers can provide faster and more adequate treatment.

In this way the achievement of a common goal depends less on the right service being available, or on knowing the actual endpoint of the service. As the number of Web services available increases, machines will need to coordinate their message exchanges in similar ways as humans attempt to: for example, which message has priority, which data are available now, what can I already act on, which responses still are to be delivered.

The persistent publication model behind tuplespaces seems well suited for coordination on the Web, particularly between Web services. Yet, applying tuplespaces to the open global environment of the Web raises new requirements, some of which have already been mentioned in previous works (Fensel, 2004; Johanson & Fox, 2004):

- *A reference mechanism*. The Web uses URLs as a global mechanism to uniquely address resources. This offers means to address particular spaces or tuples independently of their tuple fields and field data, that is, directly by use of their name.
- *A separation mechanism*. Distributed applications that have independent naming schemes may use the same names for their tuplespaces, semantics or structure. On the Web, vocabularies can be kept separate—even when using the same terms—using the namespaces mechanism.
- *The nesting of tuples*. Web data models such as XML permit the nesting of elements within a single document. Likewise, Web-based information should be able to explicitly show how units are interlinked.
- *Richer typing*. Tuple values are typed according to the core data types. However, this is not precise enough in a large-scale environment with dynamically changing information. Richer typing can support validation and correct interaction with tuplespaces.

### *2.2 Coordination on the Semantic Web*

Semantic Web services (Fensel & Bussler 2002; Lara *et al*. 2003; Sivashanmugam *et al*., 2003; Cabral *et al*., 2006) foresee the use of Semantic Web technologies in the description of services to enable a more flexible or even fully automated cooperation between disparate service-enabled applications. This means that negotiation between different coordination protocols could be handled automatically by the coordination component through its interpretation of the semantic descriptions of the conversations supported by each participating service. There are two major initiatives in the field of Semantic Web services, which each choose a different model for coordination. The OWL-S effort (Martin *et al*., 2004) defines an ontology for describing Web services, but leaves the distinction between conversational behaviour and internal composition of the service unclear. To address this lack of expressiveness for the description of complex real-world services, the OWL-S initiative has founded a Semantic Web Services Language committee whose latest specification for FLOWS, a First Order Logic Ontology for Web services, includes the conceptual notion of '*Channel*' as a repository and conduit for messages[3]. Using the operators of the logical language, it is possible to model conversational behaviour. The other main Semantic Web services effort, Web Service Modeling Framework [WSMF (Fensel & Bussler, 2002)], defines an ontology (WSMO) and a language (WSML) to express that ontology. Not only services themselves but also goals, mediators for handling data and process heterogeneity and ontologies used in the service descriptions are also part of the WSMO model, and can all be described semantically in WSML so that, for example, the ability of a service, or combination of services, to

[3] `http://www.daml.org/services/swsf/1.0/swsl`.

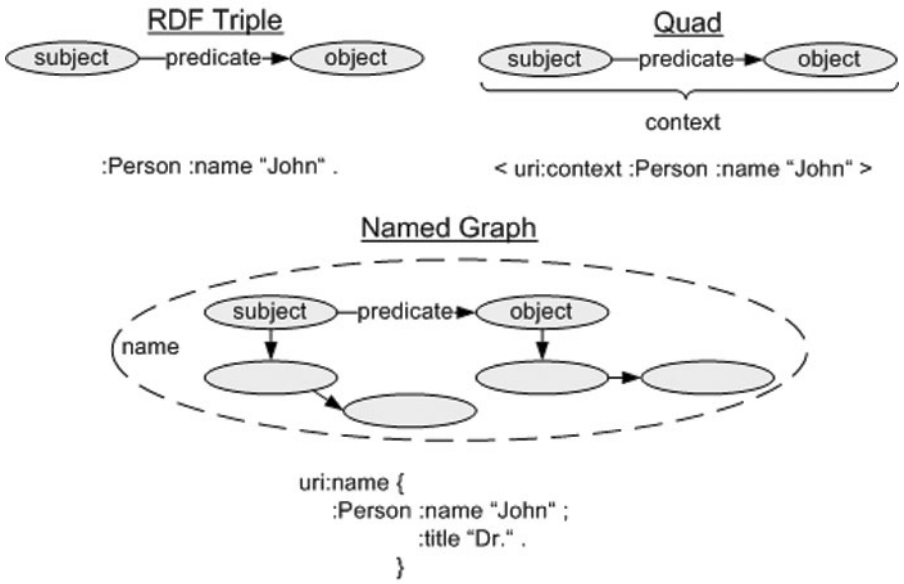


**Figure 1** Triple, Quad and Named Graph in RDF

achieve a stated goal can be reasoned over, and communication heterogeneity solved by describing the ontologies being used and making the mediators available that can mediate between two ontologies. The definition of models for the conversational behaviour of a WSMO service is still an ongoing work, with the latest specification using Abstract State Machines[4]. The WSMF initiative has also identified the potential use of tuplespace technology in Semantic Web service communication, in order to bring interactions closer to the '*persistently publish and read*' paradigm of the Web. Triple Space Computing (TSC) [as the tuplespace would coordinate the exchange of Resource Description Framework (RDF) triples (Klyne & Carroll, 2004)] has been integrated with the WSMF components into a proposal for a Semantically Empowered Service-oriented Architectures (SESAs) (Werthner *et al*., 2006).

The Semantic Web envisions greater autonomy for the services that exist on the Web, including discovery of and interaction with other services in order to achieve specific goals. Resolving goals requires sophisticated coordination. The Semantic Web brings to this scenario both increased automation of the communication—the human being taken out of the loop—and use of semantics rather than pure syntactic data in the data exchange. This is also reflected in the adopted matching algorithms. Traditional Linda-like associative addressing, in which tuples with the same number of fields and field types are matched, is no longer applicable. RDF data, the basic type of semantic data, are constructed by three resources [unique identifier (URI), literal or anonymous resource] of the form <subject, predicate, object> which in turn are composed to graphs with subjects and objects as nodes, and predicates as edges (cf. Figure 1). Tuplespaces for the Semantic Web must therefore integrate semantic matching by taking into account the meaning of resources (given by ontologies) and the relationship between the resources. The matching algorithms will thus have to make use of semantic query language-like constructs. In this respect we mention SPARQL [W3C Candidate Recommendation and currently the only language to be standardized in the near future (Prud'hommeaux & Seaborne, 2006)], OWL-QL for OWL (Fikes *et al*., 2005), or rule-based queries (e.g. datalog queries). In other words, the new matching algorithms are able to combine type and value matching, semantic querying rewriting and inference—all of these of course depending on the expressivity of the language formalism applied and on the computational costs involved by an increasing expressivity.

[4] `http://www.wsmo.org/TR/d14/v0.3/.`

In summary, a coordination model for the Semantic Web must be able to support autonomous activity of participants as well as semantic information as the data being coordinated.

A number of projects have arisen that apply space-based computing to respond to the need for coordination on the Semantic Web and Semantic Web services. We now turn to descriptions of these projects.

## 3 Triple Space Computing

TSC (Fensel, 2004) extends tuplespace computing with Web and Semantic Web technology. The work was elaborated in the scope of the TSC project[5]. In addition to the simple flat data model in which tuples with the same number of fields and field order but different semantics cannot be distinguished, Fensel (2004) proposes the use of RDF (Klyne & Carroll, 2004) to enhance the tuple model and to create a natural link from the space-based computing paradigm into the (Semantic) Web.

### *3.1 Conceptual model*

Tuples (now triples with the dimensions and semantics of RDF: <subject predicate object>) are assigned a URI and can be interlinked (just like any resource on the Web) to form graphs. Furthermore, the established namespace definition method known from XML (Bray *et al.*, 2006) and RDF can directly be used as a separation mechanism for distributed vocabularies. Moreover, the application of Semantic Web technology allows the space to install more sophisticated semantic matching algorithms than pure Linda template matching: not only the structure but also the meaning of data is taken into account.

Krummenacher *et al.* (2005) extends the work of Fensel (2004) with a concrete proposal of how to represent semantic tuples (triples) using quads (MacGregor & Ko, 2003)[6]. As mentioned above, all triples are uniquely identified by a URI and hence every triple can be distinguished from all other triples independent of its content. Retrieval of information is hence no longer done solely by templates, but also by use of the identifier.

Within the TSC project, Krummenacher *et al.* (2006) eventually suggest to only use an identifier per set of triples, that is, per RDF graph. This seems to contradict the idea of 'one resource one identifier' implied by the World Wide Web. The authors, however, argue that the use of having an identifier per triple does not justify the added complexity on storage and processing level and that the resources (knowledge) in the space are generally more complex than simple triples, that is, that graphs are required to model information. First of all, the graph URI can be used to directly address a given set of triples, which is of benefit when exchanging complex objects such as Web service descriptions or business orders (one call to the space instead of a possibly very large number of calls when addressing single triples).

Furthermore, the identifier is used as a pointer to add context information to semantic data that will not defer from one triple to another within a set that is written at the same time (e.g. the time of publication, the type of modification)[7]. The reduced granularity has a direct impact on the amount of meta-information triples, and the URI per RDF graph approach is clearly sufficiently fine-grained. Moreover, this approach has proven to be effective in Carroll *et al.* (2005a, b), where a graph is associated with a name (URI) in order to provide trust and security measures. Based on this, the data granularity adopted in TSC are chosen to reflect named graphs (RDF graph + URI) and not individual RDF triples (**Figure 1**).

[5] `http://tsc.deri.at`, funded by the Austrian Federal Ministry of Transport, Innovation and Technology under the FIT-IT Semantic Systems action line.

[6] quad = RDF triple + context.

[7] The annotation with meta-information of spaces and graphs is done by the use of a small triple space ontology that defines the properties to express vicinity of information, the publishing agent or the time of publication (Krummenacher *et al.*, 2006).

**Table 2** Examples of Semantic templates in TSC

| Template | Description |
|---|---|
| ?s a doap:Project; foaf:member ?o | Matches all triples where the subject is of type doap:Project and where the same subject has triples indicating the members |
| ?s ?p ?o. ?o a foaf:Person | Matches all triples where the object is of type foaf:Person |
| ?s foaf:name ?a; foaf:mbox ?b | Matches the triples that contain subjects for which the name and a mailbox (foaf:mbox) are indicated |

Although TSC inherits the space model from Linda, several adaptations were required to cope with the requirements of an open distributed system like the Web. In particular, the heterogeneity of data and the obvious scalability issues result in increased complexity of the tuple management mechanisms. To handle this problem, TSC defines a simple space structure: the overall information space consist of a disjoint set of (sub-)spaces, each identified by an own URI. Every space is seen to provide a particular communication medium, that is, TSC proposes the installation of new virtual spaces whenever there are new interaction contexts, new groups of agents or new security aspects in consideration. According to the authors, this approach is likely to improve at least local scalability by naturally restricting the amount of participants and hence the amount of data in a given space. More sophisticated tuplespace models such as space hierarchies or overlapping spaces are not considered in the current implementation.

### *3.2 Operations*

The interaction primitives defined for TSC were motivated by Linda, and subsequent implementations like TSpaces (Lehman *et al.*, 1999) or JavaSpaces (Freeman *et al.*, 1999). In short, the Application Programming Interface (API) provides operations for writing, reading and removing triples as RDF graphs, or named graphs (Carroll *et al.*, 2005a), in either a blocking or a non-blocking manner. Note that, in order to allow Web-like communication, the traditional template-based read and remove operations were enhanced with URI-based primitives that allow the extraction of information by identifier. In other words, the graph names, which are URIs, can be used to directly retrieve all the information attached to the given graph:

```
read (URI space, Transaction tx, URI name):NamedGraph
take (URI space, Transaction tx, URI name):NamedGraph
```

These TSC primitives emphasize again the sought convergence of space and Web technology: the data (triples) are shared in a space, but retrieved as resources by their unique name and not only by Linda-like associative matching. The transaction parameter 'tx' in the signatures allows the operation to be included in a transactional group of interaction calls, as further explained at the end of this section.

In addition to the publishing and retrieval operations, TSC adopted a simple form of a publish/subscribe mechanism. Users can subscribe to a template, which are graph pattern based expressions, as also used in SPARQL (Prud'hommeaux & Seaborne, 2006). In that way, the process flow of a user is decoupled from the information publication by other nodes and the space takes care of informing about the reception of relevant data. Semantic templates, as shown in Table 2, are of course also applied for all other retrieval operations mentioned above.

The graph patterns—acting as tuple templates—are expressed in N3 syntax (Berners-Lee, 2005). In the examples above, a '.' terminates a triple, a ';' introduces another property of the same subject, the ',' symbol introduces another object with the same predicate and subject, and the letter 'a' stands for rdf:type.

Any interaction with the triple space (TS) is at all times executed against a particular space. A concept such as the 'global space' or 'root space' is not defined in the TSC framework.

Lastly, it should be noted that all primitives in TSC provide transaction support in order to survey the execution of a group of tasks, that is, a set of operations of a process are only executed if all operations succeed or cancelled if one or more operations cannot be executed.

Transactions allow participants to concurrently access a TS without agreeing on explicit rules. A typical sequence of operations of a user in TSC could be:

(1) tx = createTransaction();
(2) namedGraph1 = read(space, tx, graphName);
(3) namedGraph2 = take(space, tx, template);
(4) graph3 = process(namedGraph1, namedGraph2);
(5) nameURI = write(space, tx, graph3);
(6) commitTransaction(tx).

This sequence of operations creates a transaction (1), reads a Named Graph identified by *graphName* from the TS (2), consumes another named graph retrieved by template (3), processes the received graphs and derives a new graph *graph3* (4) and writes the graph back to the TS (5). Finally the transaction is committed (6). By enclosing the read, take and write operations with a transaction, it is guaranteed that either all the operations are applied to *space*, or the operations do not have an effect at all. Further, it is guaranteed that the graph observed in (2) still exists and has not been modified by a concurrent operation of another participant.

The coordination layer (cf. next section) implements the transaction management, that is, the creation, committal and abortion of transactions and guarantees that concurrent operations are processed consistently.

### *3.3 Prototype*

The TS implementation is based on the CORSO (Coordinated Shared Objects) middleware (Kühn, 2001). CORSO provides a virtual shared memory space of Java objects, extending the traditional Linda model with transaction management and data replication mechanisms. TSs, as well as the named graphs, are mapped onto CORSO objects with a distinct OID (Object Identifier) in order to share them amongst the participating nodes that run a so-called TS '*kernel*' implementation. A kernel provides native or remote access to the shared objects and locally the manipulation functionality mentioned above. An outline of the proposed architecture is given in Figure 2. This architecture is a realization of a three-layer approach: (1) **access**, the implementation of the interaction primitives; (2) **semantic tuplespace**, the coordination and communication infrastructure that enables the sharing and exchange of semantic data and (3) **persistency**, the semantic data storage attached to the kernel in order to guarantee persistency. These three parts comprehend the minimal functionality that a TS kernel has to provide.

The operations layer implements the primitives defined by the TSC API and thus provides the access point for space users. The mediation support, used as a means to overcome data heterogeneity at runtime, is attached to the operation layer; the same applies to the security framework. Simple security measures are provided by the use of roles, users and permissions that can be defined with the help of the security management API. The security information and the mediation rules are stored in triple form and handled by the space infrastructure itself. A TS kernel implementation hosts not only user or application data but also the administrative or management data in an all-in-one solution. The operation layer provides, in other words, the necessary extensions to CORSO at the front end of the kernel implementation.

On one hand, the semantic data are stored in the shared object space while, on the other hand, it is written to a persistent data framework (most likely tailored to the needs of RDF data) bound to the kernel through the data access layer. The data access layer provides means to resolve

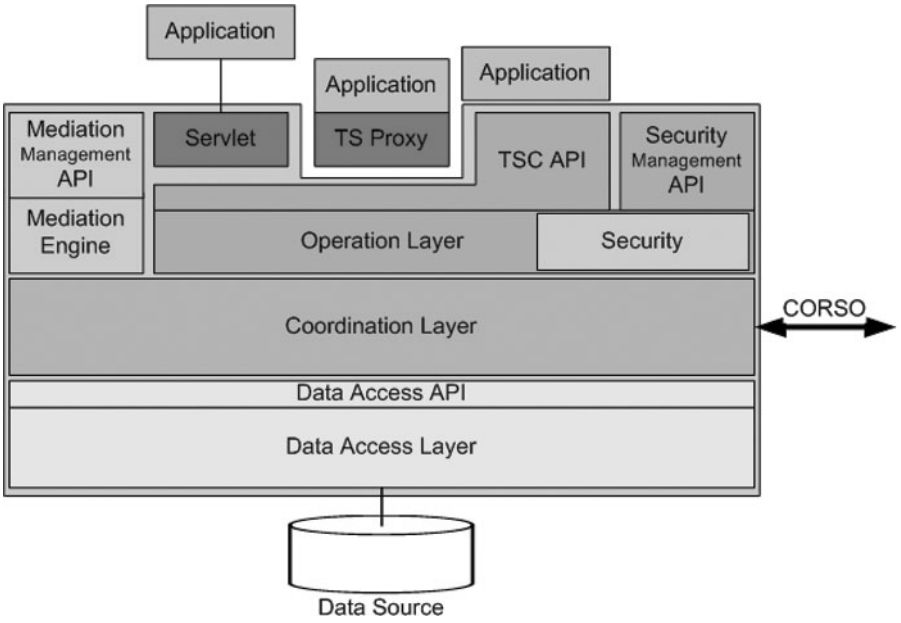


**Figure 2** TS kernel architecture outline

semantic templates and, most importantly, abstracts from the actual storage framework; this is the reason why the data access API was defined.

The prototype implementation of TSC makes use of the YARS (Yet Another RDF Store) storage framework (Harth & Decker, 2005) in order to ensure persistency and RDF querying. YARS is a highly scalable RDF store that allows the use of quads by help of its support for contextualized storage. A context in YARS is a particular area in the local or remote storage that is associated with a URI, relative for local access, absolute for remote access. Every space and graph is mapped to a particular context and hence also distinguishable at storage level. As the queries to YARS are expressed in N3QL (Berners-Lee, 2004), an N3-based (Berners-Lee, 2005) query language, the prototype data access layer is suitable to manage the semantic templates of TSC.

YARS maps the requests to tree-shaped datalog queries with one shared variable. A graph pattern is tree-shaped if its reference graph is acyclic; otherwise, the expressivity of the query engine would fall into full datalog, which could result in complex join implementations over multiple variables. Therefore, the current, and used, release of YARS does not support graph-shaped datalog queries.

In summary, spaces and graphs are, on one hand, mapped to CORSO objects for direct access of whole data objects and, on the other hand, stored in an RDF store for template (query) resolution over arbitrary graphs.

As part of the TSC project, the technologies introduced above are used as the storage and communication component within the Web Service Modelling Execution Environment [WSMX, (Mocan *et al*., 2006)], a reference implementation of a Semantic Execution Environment (SEE)[8] along the ideas of the WSMF (Fensel & Bussler, 2002). The project outcome is thus tailored to the needs of (Semantic) Web services.

## 4 Conceptual Spaces

*Conceptual Spaces*/*CSpaces* was born as an independent initiative to extend TSC (Fensel, 2004) with more sophisticated features and to study their applicability in different scenarios apart

[8] cf. OASIS Semantic Execution Environment (SEE) TC, `http://www.oasis-open.org/committees/semantic-ex/`.

**Table 3** Details about the structure of subspaces in CSpaces

| Subspace | Details |
|---|---|
| Domain theory | Stores a set of consistent logical theories that gives an explicit, partial account of a conceptualization |
| Meta-data | Provides an ontological description of the CSpace itself |
| Instance | Used to represent individuals and the values of their attributes in a domain theory |
| Trust and security | Described in terms of policy rules and reputation information (Suryanarayana & Taylor, 2004) |
| Mapping and transformation rules | Defines correspondences between common terms, relations and instances of two domain theories |
| Annotations | Defines links between concepts and instances (topics) specified in each domain theory with information resources (occurrences) |
| Subscriptions/advertisements | Stores queries that identify the information that is requested by information consumers and will be published by information producers |

from Web services: Personal and Distributed Knowledge Management, Enterprise Application Integration (EAI), Distributed Software Components and Ubiquitous Computing (Martín-Recuerda, 2005, 2006).

Just as theWeb has been characterized by an abstract model called REST, which is defined as a set of constraints (*client-server architecture, stateless, cache, uniform interface, layered system and code-on demand*), CSpaces is characterized around seven building blocks: *semantic data and schema model (knowledge container), organizational model, coordination model, semantic interoperability and consensus-making model, security and trust model, knowledge visualization model and architecture model*. Those building blocks are characterized as follows (see also Figure 3):

*Semantic data and schema model*. Defines a knowledge container, called a CSpace, in which data elements and their relations are described using a formal representation language that includes a set of modelling primitives enriched with rules in order to build a logical theory. These knowledge containers also store relations, called annotations, between data objects and related external objects like documents and Web pages. Also the relations with other knowledge containers are also included in order to facilitate interoperation among them. CSpaces can have associated access rights and maintain meta-data information about themselves that include unique identifier, creator, list of members, etc.

*Organizational model*. Promotes networks of knowledge containers (CSpace) connected by mappings and transformations rules that follow DAG (Directed Acyclic Graph) configuration. The leaves, called Individual CSpace, represent the knowledge containers created and maintained by a software agent or individual. The rest of knowledge containers defined on the upper levels of the hierarchy of each DAG are called Shared CSpaces and represent the consensual knowledge among their sons. Shared CSpaces act as semantic bridges between other Shared and Individual CSpaces facilitating knowledge access and sharing. Depending of the concrete application, Shared CSpaces can appear in three different flavours: *materialized view*, *virtual view* (Ullman, 1997) and *hybrid materialized-virtual view* (Hull & Zhou, 1996).

*Coordination model*. The coordination model is defined on top of mediated, semantic and persistent communication channels (Shared CSpaces) that represent at the same time consensual implementation of knowledge containers. Thus the concepts of knowledge repository and communication channel become one, and messages can be described in a more compact manner, because message content can refer to the ontological terms stored in the CSpace used for communication. The coordination model combines two metaphors: '*persistent publish and read*' (space-based computing) and '*publish and subscribe*'.

*Consensus-making model*. Encourages the use of collaborative approaches for creating Shared CSpaces by a group of individuals. One of the main problems of collaborative methods for building KBs is the need for the mechanism for solving disagreements between contributors. Following ideas of Kotis & Vouros (2003), a consensus-making model encourages contributors to participate in structured conversations where they can incorporate suggestions/positions that enable constructive criticism and avoid potential deadlocks.

*Security and trust model*. Relies on three relevant works in the area of distributed trust: PACE[9], PROTUNE[10] and POBLANO[11]. Similar to PACES's architecture style, four core services have been identified: Key manager, Trust manager, Execution manager and Storage manager. Unlike PACE, the security and trust model is a vertical layer that provides support to the rest of CSpaces models. Like PROTUNE, CSpaces security and trust model promotes the combination of policy-based and reputation-based trust management approaches. The reputation-based trust management has been enhanced with features proposed by POBLANO.

*Knowledge access model*. CSpaces promotes an infrastructure that facilitates users to deal with machine-processable semantics. An intensive use of knowledge access solutions, based on the graphical representation of knowledge bases and mapping rules, controlled natural language and natural language generation techniques, are the mechanisms proposed.

*Architecture model* (blue-storm). CSpaces proposes a distributed and decentralized hybrid architecture based on peer-to-peer (P2P) and client-server infrastructure for storing, reading and sharing information. A client-server P2P configuration drives a two-tiered system. The upper-tier is composed by well-connected and powerful servers, and the lower-tier, in contrast, consists of clients with limited computational resources, which are temporarily available.

Given space limitations and the need to be aligned with the other three system overviews, only the first three models are described in more detail in the following subsections.

### *4.1 Conceptual model*

This section partially covers relevant aspects of the semantic data model and the organizational model. A CSpace is a knowledge container defined as a set of tuples, where each tuple has a well-defined structure of seven fields

<guid, fm, type, subspace, sguid, vguid, mguid>

Ideally, fm is a first-order logic formula. However, limitations imposed by applications and/or members of the CSpace can restrict fm to less expressive formalisms like Description Logics, or even RDF triples. The field type identifies in which formal language fm has been defined [e.g. fol (Fitting, 1996), shiq-dl (Baader *et al*., 2003), dlp (Grosof *et al*., 2003)].

Unlike the Semantic Web, the current proposal of the CSpaces' data model does not commit to a specific formalism until an evaluation of use cases determines which languages are most appropriate. The field subspace defines a subset of the CSpace to which a tuple belongs. Currently, there are seven different types of subspaces defined for each CSpace: *domain theory*, *meta-data*, *instance*, *trust and security*, *mapping and transformation rules*, *annotations* and *subscriptions/advertisements*. The field guid is a global unique identifier for the logical formula[12], which can simplify reification and make the code more compact. The sguid field is the global unique identifier of the CSpace in which the tuple was created, which attaches provenance to a logical formula. The field vguid is a version global unique identifier for the logical formula, while the mguid field is the identifier of the member of the CSpace that stored the tuple. Given that each member of a CSpace has a

[9] http://www.isr.uci.edu/projects/pace/.
[10] http://rewerse.net/.
[11] http://www.jxta.org/.
[12] In accordance with W3C recommendations, the unique identifiers used in CSpaces follow the URI/IRI specification (Duerst & Suignard, 2005).

reputation score, this last identifier is expected for instance to be used to measure the degree of trustworthiness of each of the logical statements.

Each of the seven subspaces (cf. Table 3) can have a 'mirror' that stores an efficient representation (in terms of reasoning performance) of the data. Thus, each subspace has a raw and a reasoning side. All editing operations are done in the raw side that stores the imported and local data, schemas and mapping and transformation rules. The reasoning side provides an efficient representation optimized in order to maximize the reasoning performance. Periodically, the modifications on the raw side are transferred to the reasoning side. The strict separation between raw and reasoning side allows the implementation of additional features, such as approximate reasoning techniques (Groot *et al*., 2005) like language weakening and knowledge compilation to speed up reasoning performances and to debug methods for eliminating inconsistencies (Schlobach & Cornet, 2003).

Previous experiences of merging and mapping ontologies (Jos de Bruijn & Ehrig, 2004; Craig Schlenoff, 1999) and bottom-up generation of distributed KBs (Matteo Bonifacio, 2002) have influenced the organization model suggested by CSpaces. Jos de Bruijn & Ehrig (2004) and Craig Schlenoff (1999) promote the use of shared domain theories as shared conceptualizations and interlingua for data and application integration. In bottom-up approaches for knowledge management (Matteo Bonifacio, 2002), individuals independently build their own KBs. CO4 (Euzenat, 1995) suggests to build hierarchical structures of individual and shared KBs. Individual KBs are visualized on the bottom of the hierarchy and shared KBs on upper levels of the hierarchy. Martín-Recuerda (2006) envisions DAGs networks of Cspaces, connected by mapping and transformation rules. Two types of CSpaces have been defined: Individual and Shared CSpaces. The former is a knowledge container defined by an individual that reflects his/her own perception of a concrete domain. Shared CSpaces are CSpaces shared by several users that have reached an agreement on how to specify common domain theories, instances, annotations, etc. Shared CSpaces act as semantic bridges between several Shared and Individual CSpaces. A Shared CSpace can appear in three different flavours: materialized view, virtual view (Ullman, 1997) and hybrid materialized-virtual view (Hull & Zhou, 1996).

### *4.2 Operations*

This section partially covers relevant aspects of the coordination model of CSpaces. A detailed description can be founded in (Stijn Heymans & Scicluna, 2007). CSpaces also adopt and extend the coordination model from Linda. On one hand, Linda brings a simple and elegant metaphor for process coordination and introduces a clear distinction between process design/implementation and process coordination. On the other hand, Linda lacks in the following aspects:

- data representation capabilities;
- data retrieval capabilities;
- built-in transaction support;
- flow-decoupling from the client side.

The CSpaces semantic data and schema (introduced at the beginning of this section) provided a fixed tuple configuration that eliminates the difficulty of finding and retrieving tuples with unrestricted and unknown configurations. The use of rich and formal representation languages and the use of ontologies for describing the data are our main proposals for coping with the first problem. CSpaces aims to improve data retrieval capabilities by using a formal query language instead of template matching. This provides a more expressive means for data retrieval, allowing for more fine-grained means to look for information. Given that complex formulae can be inserted in the fm field of each tuple, template matching would not be very handy (citation) for retrieval purposes. The query language is however dependent largely on the representation formulism chosen to describe information in the space.

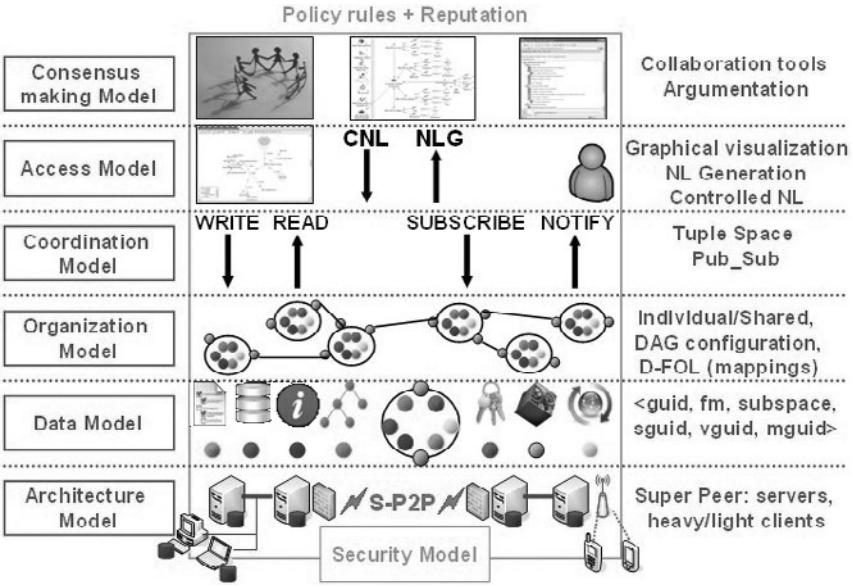


**Figure 3** CSpaces building blocks

Built-in transaction support simplifies the implementation of complex interactions between processes, because process designers do not have to implement lock mechanisms for the shared virtual semantic data space. CSpaces supports multiple retrieval and writing operations using a single primitive, and introduces new primitives for specifying the scope of a transaction.

Finally, flow-decoupling from the client side is overcome by support for the publish-subscribe model (Martín-Recuerda, 2006). Notifications contribute to the design of more flexible processes that can continue with other activities until the data requested are available in the CSpace. The interaction primitives of CSpaces were influenced by the TSpace API (Lehman *et al.*, 1999) and publish-subscribe extensions based on SIENA API[13].

### *4.3 Prototype*

The prototype described in Martín-Recuerda (2006) (see also Figure 4) has been adapted from a use case designed by members of the Engineering Informatics Group (EIG)[14] of Stanford University, and published in Jinxing Cheng & Law (2003). The aim of the use case is to integrate heterogeneous applications for project management, which teams of the construction industry use for dealing with large construction projects. The members of those teams can belong to different organizations and use diverse tools for the same purposes or for managing separate aspects of the project. Thus, large volumes of project information will be created from different geographically dispersed sources. In particular, Cheng *et al.* aimed to integrate the following tools: Primavera Project PlannerTM (P3)[15] and Microsoft ProjectTM[16] for scheduling and weather information from YAHOO[17] (substituted by National Weather service[18] in the CSpaces prototype).

The current intention for the prototyping of CSpaces is that Individual CSpaces will be stored in desktop computers (heavy-clients) and the shared CSpaces will be hosted by a server. Each computer will run Otter 3.3 (provides the reasoning space), Oracle 10g (providing persistent storage services for CSpaces) and an extended version of ActiveSpace (providing coordination

[13] http://www-serl.cs.colorado.edu/serl/siena/.
[14] http://eil.stanford.edu/.
[15] http://www.primavera.com/.
[16] http://office.microsoft.com/project/.
[17] http://weather.yahoo.com/.
[18] http://www.nws.noaa.gov/xml.

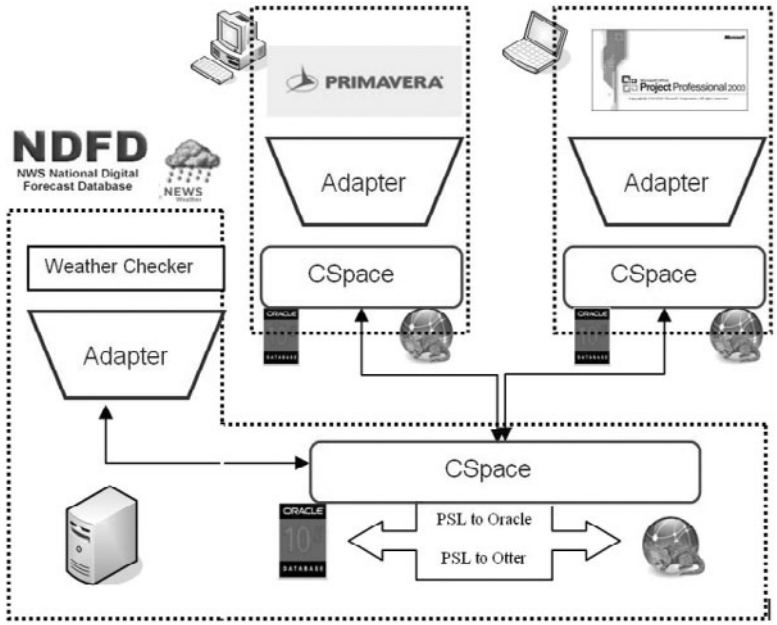


**Figure 4** CSpace prototype

services). ActiveSpace follows a JavaSpace-like abstraction for building SEDA (Staged Event Driven Architecture) style applications. SEDA is an architectural pattern for building massively scalable, distributed and concurrent systems.

Several aspects of CSpaces have been simplified or not considered in order to facilitate a prototypical implementation:

- The domain theories stored on the server and the heavy-clients are essentially the same. Only some information related with internal activities are hidden and are not shared in the Shared CSpace. Thus, mapping and transformation rules are not supported in the initial prototype.
- Current Semantic Web standards like OWL (McGuinness & van Harmelen, 2004) and RDF will not be considered in this example. Data will be represented using FOL.
- Only a materialized view approach for Shared CSpaces is considered.

## 5 Semantic Web Spaces

Semantic Web Spaces (Tolksdorf *et al.*, 2005a,b) has been proposed by the Freie Universität Berlin. It is envisaged as a middleware for the Semantic Web, enabling clients using Semantic Web data to access and process knowledge to coordinate their interdependent activities. As a result, one client is enabled to react to knowledge inserted into the space by inferring new facts and another client is freed to draw some conclusion that was waiting on the availability of these facts. The space as shared medium for knowledge enables the clear separation of the sequential knowledge processing of an individual client and the parallel coordination of knowledge between concurrently active clients.

It was originally envisaged as an extension of their XMLSpaces work, which is a tuplespace platform that extends the Linda coordination models to support the exchange of XML documents. In XMLSpaces, the tuple fields may contain XML documents and matching templates are based on XPath or XMLQuery expressions.

Semantic Web Spaces extends the Linda coordination model in a similar manner to support the exchange of RDF triples as tuples, with matching based on RDFS reasoning capabilities. Just as the Semantic Web is conceived as a set of layers in which RDF is built upon by the subsequent layers of ontologies, logic, proof and trust [cf. Figure 5 according to Berners-Lee *et al.* (2001)],

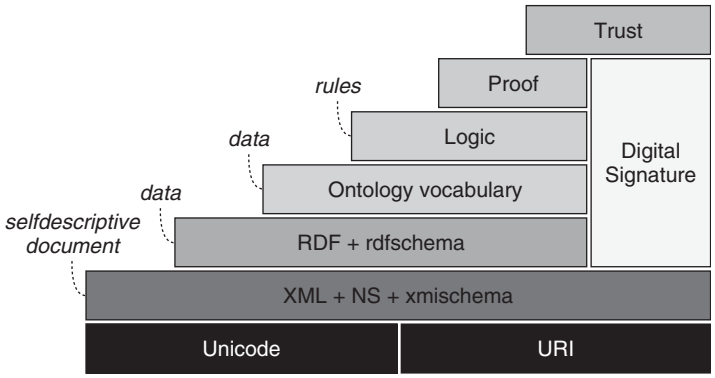


**Figure 5** The Semantic Web stack

this platform is seen as the first step towards further spaces for the Semantic Web. Extensions for OWL and rules, for instance, would permit the use of more expressive languages in the tuples as an extension of the RDF(S) concepts already used in the approach described in this article, while trust-enabled Semantic Web Spaces would include agent policies as tuples and execute matchmaking agents to determine if two agents (source and target) can trust one another before permitting an operation.

### *5.1 Conceptual model*

A conceptual model has been drawn up (Tolksdorf *et al*., 2006) in which the extensions to the traditional Linda coordination model necessary to support a tuplespace that manages Semantic Web information were considered.

The tuple model is based on the concept of RDFTuple. This is a tuple that contains four fields which take URIs as values. The first three fields reference Web resources corresponding to the subject, predicate and object of an RDF statement. The fourth field is an identifier for the tuple, generated with the help of a tuplespace ontology (see below). Each field is typed by an URI, while the ID field is defined as an RDF ID (Hayes & McBride, 2004). Special consideration is taken for representing blank nodes, containers/collections and reification.

Semantic Web–specific matchings using RDF-specific reasoners are added. In combination with available ontologies, RDFTuples introduced to the space can be checked for ontological conformance. Template matches can be made not only against the actual RDFTuples in the space but also against those which can be inferred. For instance, subClassOf and subPropertyOf statements allow matches to take place on the basis of subsumptive reasoning. This means that any variable typed with Class A in a template can be matched to a constant typed in Class B in the respective field of a tuple if A subsumes B.

While the original Linda considered a single tuplespace, extensions have introduced multiple, nested and hierarchical spaces. The distributed and replicated Semantic Web Spaces are virtually partitioned using contexts, drawing on the concept of scopes (Merrick & Wood, 2000). Clients may be allocated certain contexts, controlling their view upon the space to those tuples existing within their context. Contexts provide a simple form of access control, allowing clients to have private spaces as well as shared spaces with specific other clients. From the system perspective, they can be used to perform clustering (of RDFTuples which are related in some way) and hence to improve matching efficiency.

In addition, Semantic Web Spaces define an ontology for describing the space itself. Thus it creates a meta-space of RDFTuples, which explicitly represents the actual structure of the active Semantic Web Spaces. An instance of the Semantic Web Spaces ontology forms a queryable (and possibly editable) description of the space, including its permitted structure, supported tuple types and matching templates, and effective access and trust policies. The meta-model of the Semantic

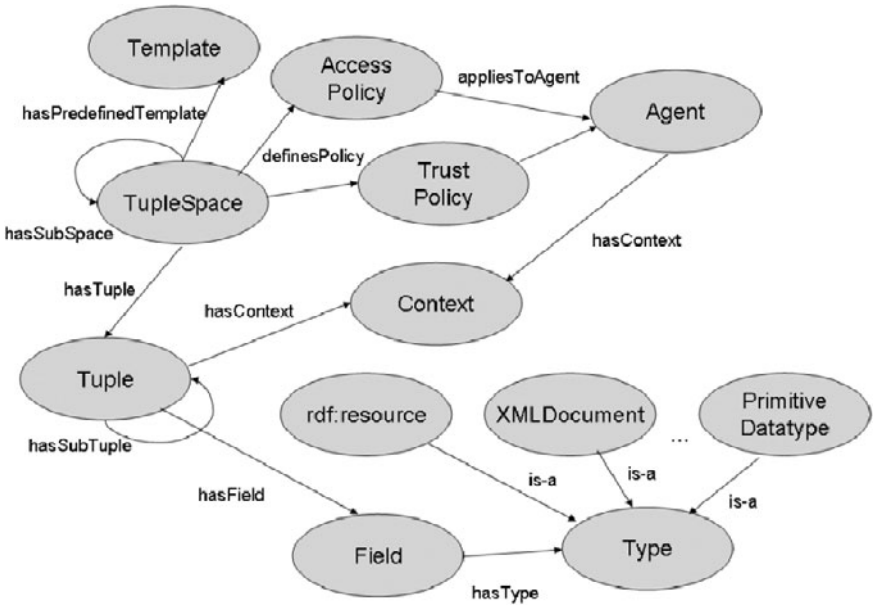


**Figure 6** Semantic Web Spaces ontology

Web Spaces contains all instances of tuples currently stored in the space (and hence provides for each the unique URI by which they can be referenced) and can store meta-information relating to each tuple such as its author, insertion time, number of reads or current context.

A part of the tuplespace ontology is shown in Figure 6.

### *5.2 Operations*

Previous works on defining the formal semantics of Linda operations (Busi *et al.*, 2000a,b; Busi & Zavattaro, 2000) did not take into consideration the consequences on those semantics when the data being coordinated can be seen as carrying meaning, as Semantic Web data would be interpretable as being collections of statements of knowledge. Hence Semantic Web Spaces defines new operations, which redefine the classical Linda operations for the coordination of RDF data, both syntactically and semantically.

Two views on the coordination of Semantic Web data are defined: the data view, where tuples contain syntactically valid RDF data without any formal meaning, and the information view, where RDFTuples are recognized as being special data structures that express formally defined knowledge about concepts. This imposes consistency and satisfiability constraints with respect to the RDF semantics (Hayes & McBride, 2004) and to the associated ontologies defined in RDFS (Brickley & Guha, 2004).

In the data view, a Linda-compliant variation of the traditional out in and read operations is defined (Table 4). They preserve the original Linda operation semantics while operating on the structure of RDF triples. In addition, as interactions with RDF often make use of graphs of RDF triples rather than individual triples, multiple tuple operations that output or read a set of RDF triples as a single request–response are also specified (outgr, rdgr and ingr).

Handling SemanticWeb knowledge in the information view requires, however, coordination primitives which take into account the truth value of the underlying tuples and the ontologies the tuples might refer to. For this purpose, the operations claim, endorse and retract are introduced.

The claim operation is similar to a Linda out, but permits the claim to be made only if the statement is consistent with a RDF schema that constrains the meaning of RDF classes and properties. The endorse and retract operations block and use triple pattern matching, with support for the

**Table 4** Coordination model in Semantic Web Spaces

| | Data view |
|---|---|
| outr: (Statement) → boolean | Insert a new RDF statement to the data view |
| outgr: (Model) → boolean | Insert a set of RDF statements extracted from a Jena model to the data view |
| rdr: (Triple or Node) → Statement | Read an RDF statement from the data view of the space using a three-fielded template of the form (s,p,o) or a Node containing a tuple identifier |
| rdgr: (Triple) → Model | As rdr but returns all matches as a Jena model |
| inr: (Triple or Node) → Statement | Destructively read an RDF statement from the data view of the space likewise with a template or tuple identifier |
| ingr: (Triple) → Model | As inr but destructively reads all matched RDF statements from the data view |
| Information view | |
| claim: (Statement) → boolean | Assert a RDF statement in the information view of the space if consistent with the RDF Schema |
| endorse: (Triple) → Subspace | Read a RDF statement from the information view of the space |
| excerpt: (Triple) → URI | Read all matching RDF statements by copying them into a context and returning an URI identifying it |
| retract: (Triple) → Subspace | Deny the truth value of an RDF statement in the information view of the space (retained in the data view) |

RDF semantics (e.g. subsumption-based on RDF schema information), to match and return an RDF statement in the space, which may also be an inferred statement (i.e. not explicitly claimed in the space). A 'subspace' is returned, that is, a set of triples, as matches may not be expressable in a single statement, as is the case when matching on an RDF collection or container. Additionally, Semantic Web Spaces does not return any statements with blank nodes without providing the statements also that share this blank node. Additionally, retract will remove the matched statements from the space, ignoring inferred statements. While a statement may occur multiple times in the data view, it can only exist (in its own space) once (as a statement is valid regardless of how many times it is stated). However, removal of a statement from the information view does not change the data view, as retraction is interpreted as a falsification of the statement rather than its deletion.

The excerpt operation in particular uses a 'context' in the information view to contain a set of copies of the matched tuples in a private partition of the space (the reference is only passed to the agent excerpting the triples). As a result, contexts explicitly contain inferred tuples, which are only implicit in the information view of the space. This allows agents to construct RDF models and continue interacting with those models without affecting the rest of the space (e.g. destructive reads).

### *5.3 Prototype*

Figure 7 shows the full architecture model for Semantic Web Spaces. From left to right, it can be divided into three major components:

- **The publication component** deals with I/O mechanisms of the space and the way it organizes the data.
- **The retrieval component** implements various tuple matching heuristics that are applied to return the results of the agent's queries.
- **The security component** guarantees the secure execution of the aforementioned activities.

From top to bottom, the architecture contains three layers: the first two layers correspond to the information and data view that was mentioned in the previous section, while the third layer

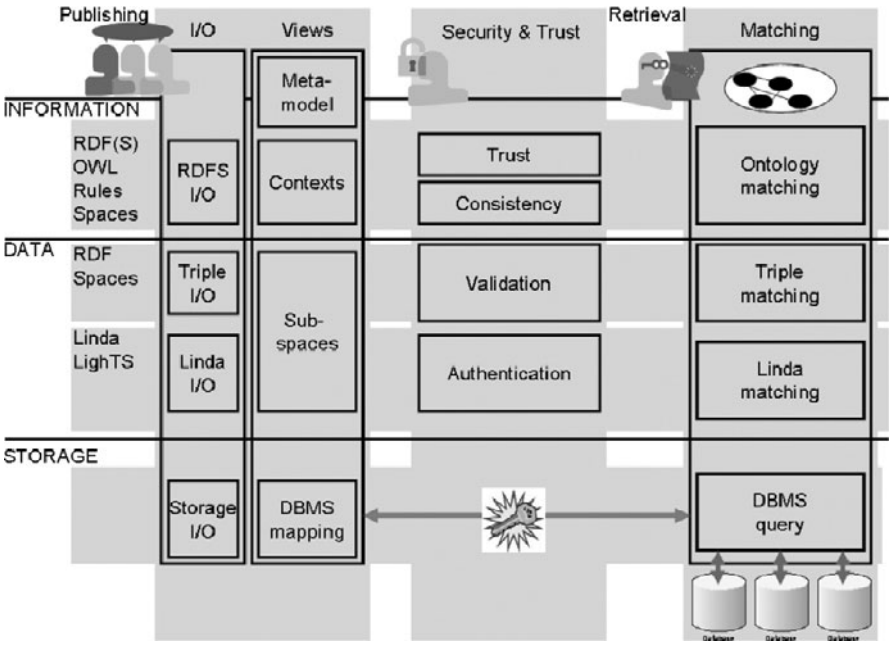


**Figure 7** Architecture of a Semantic Web Space

handles the persistent storage of the tuplespace information. Accordingly, from bottom to top, the tuplespace system is concerned with raw data, syntactic virtual data (Linda tuples) and semantic virtual data (RDF tuples).

The publication component describes how the platform communicates with other systems, whether they are Semantic Web clients or storage managers (interfaces to the back-end storage). For each of the three architectural layers the component implements an API, which defines the format of the messages that can be sent to it, the semantics (meaning) of those messages by which the developer can know which response to expect from a certain message and the format of that response. Further on, the component includes methods that seek to optimize the operations on the space through the management of tuples being added, searched for or removed from the space. Subspaces and contexts are forms of partitioning the tuplespace into defined sections, based on data or knowledge content of tuples. The RDMS (Relational Database Management System) mapping is not just a syntactic mapping to the API of the back-end storage; it also optimizes these operations. The meta-model, that is, the tuplespace ontology, organizes the meta-data for the tuplespace so that one can express the structure of the tuplespace and the tuples it contains. The system can access this model to query where certain tuples may be found or allow agents to optimize their messages to the system.

The retrieval component implements the way external templates are matched against the tuplespace data. It is related to query processing (e.g. query resolution and query rewriting) on the tuplespace or on the physical data stores, and can imply the usage of reasoning services in order to enhance the results of pure syntactic data retrieval.

The third component of Semantic Web Spaces is concerned with security and trust. It is accessible to the publication and the retrieval operations. Any operation by an agent on the space may be proofed by security and trust components, whether it is a retrieval operation (which requires matching, e.g. at storage level) or just a tuple addition (which only requires access to the tuplespace through the tuplespace API).

Semantic Web Spaces takes a lightweight approach to implementation to allow for more flexibility in the selection of solutions for different aspects of the implementation and to aim at a minimal system footprint and simplicity/flexibility of code in order to try to maximize efficiency

and the possibility of later code optimization. It uses the LighTS tuplespace framework (Picco *et al*., 2005) and extends it for handling the tuples that contain semantic information.

The prototype implements the front- and back-end of the architecture above, but does not yet include support for security and trust. The data view is a Jena RDF model in which all triples are reified (and hence referenceable and duplicable) and there is no inference[19]. The information view is a Jena RDF model with inference, so that it reflects the data view extended by the triples inferrable from its content according to the available RDF schema(s). The TS ontology is a Jena RDF model with the meta-model of the space. To save on accesses, the ontology is not updated after every operation on the space, but temporary models log changes and updates are made after a certain amount of operations. Retrieval is by SPARQL (Prud'hommeaux & Seaborne, 2006) queries generated from the triple patterns of the retrieval operations, which are applied to the appropriate Jena model. Persistent storage is implemented by using the Jena2 support for RDBMS.

The prototypical implementation has demonstrated that an approach based on Linda and tuplespaces can be realized to coordinate between agents communicating on the Semantic Web (i.e. sharing semantic data). A first evaluation (Nixon *et al*., 2007) has shown that challenges remain in implementing the system in a highly scalable manner. A further open issue is related to the appropriateness of Semantic Web frameworks such as Jena in the context of coordination systems, since such components introduce an overhead in the processing of Semantic Web information that is not required, for example, at the data level of Semantic Web Spaces.

## 6 sTuples

sTuples (Khushraj *et al*., 2004) has been developed as part of the pervasive computing work at the Nokia Research Center. Given the particular characteristics of pervasive environments, that is, the heterogeneity and dynamics of multiple clients in the environment, the Semantic Web was seen as a solution to semantic interoperability issues, while tuplespaces were seen as a satisfactory middleware able to provide data persistence, as well as temporal and spatial decoupling and synchronization.

### *6.1 Conceptual model*

sTuples consists of three key extensions to the Linda model:

- Semantic tuples extend the data tuple.
- Tuple template matching is enhanced by using a semantic match on top of object-based matching.
- Specialized agents reside on the space and perform user-centric services such as tuple recommendation, task execution and notification.

A semantic tuple is a JavaSpace object tuple that contains a field of type DAML + OIL Individual (McGuinness *et al*., 2002). This field contains either a set of statements about an instance of a service, or some data or an URL from which such a set of statements can be retrieved. Semantic tuples can be either data tuples or service tuples, depending on whether they contain the semantic information provided by a service/agent or are advertising an available service (such as controlling a light or the volume of a television set). Both categories can be further refined in an ontology of semantic tuple types.

A semantic tuple manager is in charge of managing all interactions in the space concerning semantic tuples (i.e. insertion, reading and removal). When a semantic tuple is added to the space, the DAML + OIL statements it contains are extracted and asserted in the space's own knowledge base. The system checks that the statements are valid, and that the knowledge base remains consistent. Likewise, when a semantic tuple is removed from the space, the statements that it contains are retracted from the knowledge base.

[19] `http://jena.sourceforge.net`.

A semantic tuple matcher carries out the matching of templates to semantic tuples. Reasoning capabilities are provided by RACER, a Description Logics reasoner (Haarslev & Möller, 2001). A semantic tuple template, unlike the usual Linda approach of actual and wildcard values, is a semantic tuple whose DAML + OIL individual-typed field draws upon a dedicated 'TupleTemplate' ontology. A set of statements using this ontology can be interpreted by the matcher as a semantic query upon the statements in the space's local knowledge base. However, due to the increased complexity of different DL-based queries, the matcher performs its matching through a series of steps of increasing complexity.

(1) The statements are validated against the TupleTemplate ontology so that invalid queries are immediately rejected.
(2) The candidate semantic tuples are selected by matching their tuple type (e.g. Light-Service as a subclass of ServiceTuple) against the value of the hasTupleCategory property in the query.
(3) RACER reasons over the set of candidate tuples so that inferable facts can be available (e.g. all classes that an individual belongs to through subsumption).
(4) The tuple template contains different TupleFields, which express desired or undesired field types and values. An exact match occurs when a semantic tuple is found, which contains all desired tuple fields (in terms of the expressed type and possibly value) and does not contain any undesired tuple fields.
(5) If there were no matches and subsumption matching was requested in the tuple template, then the subsumption of field types is also taken into consideration in searching for a match.
(6) If there were no matches and plugged-in matching was requested in the tuple template, then plugged-in tuples will be matched too.
(7) Otherwise, there are no matches and no tuple is returned.

Matching tuples will be weighted based on the degree to which they match the template. The matched tuple with the highest weight is selected to be returned to the client.

### *6.2 Agents*

Specialized agents reside in the space and offer added functionality to the user by abstracting typical user functionality needs and hence simplifying client interactions. In general, clients continue to interact with the Service Managers, which mediates between the clients and the available services in the network. New services register themselves in the system by passing a Service Tuple instance to the manager containing a service identifier, the DAML + OIL instance describing the service, a free text description, a service icon, a limit of the number of threads the service can support, a lease (specifying the duration the service remains active) and a location dependency indicator. Likewise, data from clients or services are passed as Data Tuple instances to the manager and contain a unique identifier for the tuple producer, a DAML + OIL instance containing the data to be shared and a list of subscribers to that object. The Service Managers are arranged in a tree-like hierarchy, and each has its own space and specialized agents.

The Tuple Recommender Agent allows a client to register its interests with a Service Manager using a pre-defined preferences ontology. The agent can monitor the space for any services or data that match the interests of the client. If no matches are found at the time of the request, a notification request is registered with the space for a specified time period for any matching tuples that may be added to the space.

The task execution agent acts as a proxy for the user. The client registers tasks with the manager, using a dedicated task ontology. Matching service tuples are retrieved and subscribed to, and commands are sent to the service as specified in the task ontology instance (e.g. switching a light on or off). The service response can also be captured (if specified in the task ontology instance) to be returned to the client or passed to another service (in the case of composite tasks).

A publish-subscribe agent dynamically delivers data to users that have subscribed to it. A data tuple is written to the space that is meant to be shared by multiple clients. A client requests data tuples of a particular type by using the tuple template ontology. The agent will find a matching data tuple and add the requester to the tuple field containing the list of subscribed users.

### *6.3 Prototype*

sTuples was built as an extension of Sun's JavaSpaces, which provides a centralized server and already extends the classical tuplespace model with field and tuple typing (based on Java's object-oriented model), Java objects as tuple contents, object-based polymorphic matching, transactional security and a publish-subscribe mechanism. A generic SemanticTuple interface extends the JavaSpaces Entry interface. RACER is used as a reasoner, which also supports the publish-subscribe mechanism.

It is also integrated with the Vigil framework for realizing 'Smart Home' scenarios in which mobile clients access home devices such as lights and consumer electronics over low-bandwidth wireless networks. Vigil provides distributed trust, access control and authentication services in the pervasive computing environment.

## 7 Comparative study

In this section we perform a comparative study of the semantic tuplespace proposals presented in the previous sections on the basis of a pre-defined classification framework. The framework has been compiled by the authors of the article according to their expertise in the two fields of research, which are relevant for the state-of-the-art survey—Linda-based coordination systems and the Semantic Web—complemented by a comprehensive literature survey (Omicini *et al.*, 2001; Johanson & Fox, 2004; Wells *et al.*, 2004). The classification dimensions can be divided into two categories. The first dimension aims at situating the analyzed approach within the general field of coordination middleware. The second dimension is concerned with aspects that are typical for the Web, the Semantic Web and semantics-aware information systems. In the following, we elaborate on each classification dimension.

### *7.1 Part I—Linda-based coordination systems*

*Type of the approach:* Whether the proposal is an extension of the original **tuplespace technology** or if it rather applies this technology in a particular application setting, thus being a **tuplespace-based technology**.

*Scope of the approach:* A list of possibly overlapping application areas the proposal is targeted at. Examples in this category include **the Web**, **distributed systems**, **mobile systems**, **knowledge management**.

*Communication/coordination model:* Information about the way agents interact with the tuplespace. Examples of possible communication models are **Linda**, **publish-subscribe**, **message queues**.

*Extensions of the communication/coordination model:* The types of extensions from the original communication model, which are available in the surveyed system. Examples for the Linda communication model include **copy-collect**, **tuple expiration**, **notification**, **ordering**, **clustering** or various heuristics to perform **template matching**.

*Tuplespace model:* An account of the way tuples are organized in tuplespaces. We encounter several types of tuplespaces in the Linda literature, for example, **plain** tuplespaces, **nested** tuplespaces, tuplespaces organized **hierarchically**, **scopes**.

*Tuple model:* A description of the way data are organized as tuples. This includes the number of allowed fields and the data types that are permitted as tuple field values: primitive data types such as integer, strings or floats, arbitrary Java objects, URIs, DOM objects, XML documents.

Further on, tuples may contain further tuples, a model that is referred to as **nested tuples**.
*Distribution:* Whether the tuplespace data are distributed over multiple Linda kernels or implemented on a single server.
*Replication:* Whether the data can be replicated over multiple distributed spaces or it is stored in one space at a point in time.
*Implementation:* Information about the implementation platform(s), the development architecture, the implementation language(s), etc.

### *7.2 Part II—Semantic Web:*

*Resource identification:* Whether and to which extent the semantic tuplespace system provides support for the identification of resources. From a Web point of view, the use of URIs would be the most obvious approach.
*Addressability of spaces:* The means applied to address multiple spaces if available. On the Web, domains are addressed by unique identifiers; a particular space can hence be seen as a particular domain. Note that in this context Internet domains, and hence spaces, are treated just like any other Web resource.
*SemanticWeb languages:* The languages that are supported by the semantic tuplespace system at tuple model level. Examples include typical Semantic Web knowledge representation languages such as RDF(S), OWL, SWRL.
*Reasoning:* To which extent and in which form the system provides support or makes use of reasoning services.
*Validity and consistency:* Support for validity and consistency checking of the semantic data stored and managed by the system at global and local scale.
*Semantic matching:* Support for reasoning-enabled matching on asserted or inferred tuple data. The most obvious example in this category includes the usage of subsumption reasoning on application-specific type systems.
*Semantic clustering:* Support for distribution strategies on a semantic basis. In this case the meaning of the data, which are structured using formal ontologies, can be used to derive new rules for clustering and distributing the tuplespace among multiple kernels.

Table 5 summarizes the results of the comparison according to the classification framework above.

In the remainder of this section, we discuss the most important aspects of this comparative study.

With respect to the underlying tuple and tuplespace models, the lowest common denominator of the analyzed semantic tuplespace platforms is the support of new tuple types encapsulating the data formalized using Semantic Web languages. Except for the sTuples proposal, which focuses on DAML + OIL, the remaining three tuplespace approaches foresee a certain level of support for RDF data. Semantic Web Spaces and CSpaces provide first ideas with respect to RDFS, OWL and SWRL support. Usually, tuples are extended with an identification mechanism. CSpaces, TSC and Semantic Web Spaces also suggest a means to attach provenance information to individual (or sets of) tuples; however, they resort to slightly different interpretations of the provenance concept. In CSpaces, the meta-data are an integral part of the tuple model, while the other approaches extend this concept by proposing a meta-model for capturing various types of information about a space, its contents and the agents interacting with it. CSpaces and TSC introduce versioning information to the classical tuple notion, and the unique identifier of the creator of the tuple for trust purposes. Versioning is out of the central focus of Semantic Web Spaces and sTuples. As a conclusion, CSpaces relies on a more complex data model that addresses many of the requirements specified by sTuples, TSC and Semantic Web Spaces. The other approaches take into account these requirements; however, they do not embed all the corresponding information at the data model level. Semantic Web Spaces, for instance, rely on a very simple

**Table 5** Comparison of the approaches

| | TSC | CSpaces | SWS | sTuples |
|---|---|---|---|---|
| Linda-based coordination systems | | | | |
| Type of approach | Tuplespace technology | Tuplespace-based technology | Tuplespace technology | Tuplespace-based technology |
| Scope of approach | Semantic Web service | Distributed knowledge management | Semantic Web | Mobile systems |
| Communication/ coordination model | Linda, publish subscribe | Publish subscribe | Linda | Publish subscribe |
| Extensions of the communication/ coordination model | Notifications, transactions | Multiple read and write transactions | Data vs. semantic level | – |
| Tuplespace model | Multiple independent spaces | DAG configuration of interconnected spaces | Disjoint, nested spaces and temporary contexts | Centralized space |
| Tuple model | Named graphs | Fixed tuple format of arbitrary data units associated with IDs, versioning and meta-data information and containing references to spaces and interpreting semantics | Nested tuples, triples | ServiceTuples and DataTuples containing DAML + OIL instances |
| Distribution | Super peer architecture on top of CORSO and JavaSpaces | Super peer architecture | Self-organized, distributed architecture, ontological description of the space | Centralized architecture based on JavaSpaces |

**Table 5** *Continued*

| | TSC | CSpaces | SWS | sTuples |
|---|---|---|---|---|
| Replication | Eager replication provided through CORSO | – | – | – |
| Implementation | Java | Java | Java | Java |
| Semantic Web | | | | |
| Resource identification | URI of named graphs | Data units identified by global system IDs | URIs defined in the tuplespace ontology | Object IDs provided through JavaSpaces |
| Addressability of spaces | URIs | Global system IDs | URIs defined through the tuplespace ontology | Not applicable |
| Semantic Web languages | RDF | Language-independent, up to FOL | RDF(S), extendable to OWL, SWRL | DAML + OIL |
| Reasoning | – | Used for query answering, query rewriting and consistency checking | Used to define semantic matching methods and to validate the contents of spaces | Used to define semantic matching methods |
| Validity and consistency | – | Consistency checking and syntactic validation services | Against ontologies and using conventional validators for RDF | – |
| Semantic matching | Graph pattern templates and N3QL resolution | Matching based on query engines | Subsumption-based matching | Subsumption-based matching |
| Semantic clustering | – | – | Extendable to self-organization supported by tuplespace ontology | – |

data model and handle provenance and identification issues with the help of a tuplespace ontology, while sTuples uses the object identification mechanism delivered with JavaSpaces. TSC uses quads as primary data unit, hence encapsulating data and identification information in one object.

Further on, Semantic Web Spaces, CSpaces and TSC define an additional layer on top of the data model. In the case of Semantic Web Spaces, this is called information view and visualizes tuples as RDF graphs. For CSpaces, tuples are logically grouped in knowledge containers that store a logical theory, the relations (mappings) with other logical theories (other CSpaces), relations with real-world objects (annotations), security and trust information, and a meta-data characterization of the CSpace itself. Such meta-data are considered with minor variations in each of the aforementioned approaches, which recommend the usage of ontologies as a basis for its modelling and formalization. However, only Semantic Web Spaces and TSC currently provide specifications of such a tuplespace ontology.

The tuplespace model is explicitly taken into consideration in three of the presented approaches (TSC, Semantic Web Spaces and CSpaces). TSC and CSpaces explicitly include the notion of several independent tuplespaces, although in the case of CSpaces, the spaces that have domain dependencies are interconnected by mapping rules. Semantic Web Spaces, on the other hand, mention one global space that can be partitioned using the notion of contexts.

The communication/coordination model underlying the four approaches is based on Linda or the publish-subscribe paradigm. In addition, all proposals take into account the advantage of Semantic Web technologies to improve the matching abilities of retrieval operations. However, only Semantic Web Spaces distinguish between operations at data syntax level and semantic level. The former group of operations guarantees backward compatibility with classical Linda applications. sTuples and CSpaces use publish-subscribe as the underlying communication paradigm. This model has been already considered in commercial implementations of tuplespace computing like TSpaces and JavaSpaces to one of the major limitations of Linda-based computing: the flow-coupling of consumer applications. In addition, the combination of tuplespace computing and publish-subscribe improves the latter, avoiding the event-storm problem. CSpaces introduces atomic multiple read and write operations to deal with the particularities of information retrieval on the Semantic Web, as information items published and consumed by semantic applications tend to contain multiple RDF triples, which hence need to be written or read from the space within one operation. Atomicity of operations is also the main motivation for extending the Linda communication model into transactions, as it is the case in TSC. Further on, similar to CSpaces, which are based on publish-subscribe, TSC uses notifications as a means to decouple the flow of semantic applications accessing the space.

From an architectural point of view, the envisioned systems are designed to rely on decentralized models. While sTuples resorts to the centralized architecture underlying JavaSpaces, the remaining approaches foresee an architectural model supporting decentralization while converging in terms of requirements like scalability. TSC follows REST principles, and CSpaces induce decentralization and self-organization by means of P2P ideas, while Semantic Web Spaces aim at tackling these issues using intelligent distribution strategies [e.g. self-organization on swarm intelligence principles (Tolksdorf & Menezes, 2003)]. The four surveyed systems have been prototypically implemented in Java.

Aspects such as distribution and replication remain to be further investigated in the field of semantic tuplespace computing. Though the need for adequate strategies is acknowledged in the context of the TSC, CSpaces and SemanticWeb Spaces, all of them being targeted at open, dynamic environments such as the Web, the proposals consider them only marginally. Further research is also needed with respect to the question of how semantic technologies can enrich the traditional techniques employed to solve these issues.

## 8 Conclusions and outlook

The survey that has been conducted indicates the increased interest in taking Linda-like approaches to communication and coordination as a means to propel the simplification of

large-scale, open and distributed systems. All four approaches intend to demonstrate the benefits of using a shared information space as an alternative to traditional coordination middleware, often based on the message-based paradigm. The application areas of the surveyed systems are diverse; the reasons for deploying tuplespace technology remain, however, the same, the simplification of interaction patterns and implementation of large-scale heterogeneous complex systems.

Existing Linda tuplespace implementations have already provided evidence that the original Linda concept does not fully comply with the increased complexity of current large-scale application environments. This can be already observed, for example, in the scope of systems such as JavaSpaces or TSpaces. Some important technological updates introduced by these implementations include notifications through publish-subscribe support, support for transactions, layered or nested spaces instead of single servers, and more expressive tuple fields and matching algorithms. The semantic tuplespace implementations surveyed here include additionally the application of Semantic Web languages like RDF, RDFS or OWL, as well as the enhancement of the available template matching algorithms with Semantic Web–specific query languages and reasoning.

Some initial specifications and prototypes of these implementations have been evaluated and compared in the previous chapter, and we now highlight several open issues in systems and research challenges for the future. We conclude with a brief summary of the potential for this emerging research area and of the findings of this first analysis of the start-of-the-art.

### *8.1 Open issues and research challenges*

The analysis makes clear that aspects like data distribution and replication still require further investigation, and that semantics-aware tuples and template matching are not enough to realize a feasible Web-scale semantic middleware. The four approaches acknowledge the need for enhanced replication and caching algorithms for a successful application of semantic tuplespaces in open, distributed systems; however, they do not yet consider these issues to a satisfactory extent.

We therefore expect future research to concentrate on such mechanisms in order to address the significant challenges of distribution and scalability, which are the key problems in dynamic large-scale systems like the World Wide Web or pervasive computing environments. Semantic clustering of data, organization of spaces according to the internal structures of data and the joint usage of local and global spaces are possible starting points for the future extensions to existing semantic tuplespaces. Some of these ideas were already materialized in the presented projects or at least marginally considered, but not yet implemented. We thus expect that upcoming work will take up these ideas, and that solutions for the mentioned distribution and scalability issues will be developed around them.

The strength of semantic tuplespaces is the natural and integrated link to the languages and tools of the Semantic Web. The presented approaches make first steps into this direction; however, it is also obvious that further work is required to fully enable the Semantic Web within tuplespace environments. Moreover, the future of the Semantic Web is seen in the integration of rule languages with the currently available W3C recommendations RDF(S) and OWL. Such complex knowledge representation formalisms and the associated sophisticated reasoning services they enable are still missing from the surveyed approaches.

Support for more expressive knowledge representation languages would form the basis for more powerful means to formally define meta-information about spaces for the usage of such meta-models to configure and optimize the management of the middleware. TSC and Semantic Web Spaces already propose tuplespace ontologies to capture and manage the available application data and the space hierarchies. Semantic middleware management is seen to be one of the major advantages of semantic tuplespaces compared to traditional space frameworks, as it provides advanced means for data and infrastructure handling, which directly influence and likely improve the necessary distribution and scalability measures.

### 8.2 Research area potential

The application fields and use cases of the presented semantic tuplespaces already reveal some of the many expected areas of interest for this novel technology. In particular, the sTuples project shows that semantic tuplespaces are not only to be applied to the Semantic Web or Semantic Web services but also to many other areas—if not all—where distributed, heterogeneous and autonomous agents interact in open large-scale systems. In the particular case of sTuples, the space was installed to improve a smart home installation, that is, a pervasive computing application, where many collaborative agents share information in order to improve the user experience.

Besides pervasive computing, and consequently mobile and ubiquitous computing for the same reasons, it also makes sense to look at the enhancement of Grid architectures. In particular in the course of the Semantic Grid, the convergence of the Grid and Semantic Web services, it will be interesting to apply semantic tuplespaces in order to ease the interaction of computation and information services (Shafiq *et al.*, 2006).

In summary, the presented application fields are very diverse, ranging from sharing information on the Semantic Web, or collaboration in knowledge management and pervasive computing to a full-fledged communication and coordination paradigm for Semantic Web services or the Semantic Grid. There, the potential application areas are just as diverse: EAI, eHealth, digital multimedia systems and recommender systems, to only mention a few.

### 8.3 Summary of findings

This article introduced the novel field of research that is interested in the combination of semantics and tuplespace computing. The emergence of Semantic Web technology in the area of distributed computing—which consequently allows heterogeneous and autonomous agents to interact and collaborate without direct human intervention—reactivated the ideas of tuplespaces in the large. The Linda language introduces the decoupling of processes in various dimensions and hence obvious advantages for large and complex open systems, in particular, as shown in this work, in combination with the Semantic Web:

- Decoupling of agents in time and reference inherited from Linda.
- Simplified interaction patterns for complex, distributed systems.
- Ease of implementation through unified and simple interaction primitives.
- Enhanced search algorithms by means of Semantic Web technology.
- Data and process mediation to counteract the increasing heterogeneity problems.
- Possibility for integrated knowledge management through reasoning support.

We have investigated and analyzed four current semantic tuplespace projects, and identified some open issues and future research challenges on the way to making the vision of the semantic tuplespace paradigm a reality.

First specifications and prototypes of the considered systems indicate a core agreement on some principles to follow for developing semantic tuplespaces, while differences in approach will provide a basis for scientific evaluation and comparison as prototypes are completed and become stable. Already, it has been identified that in particular more work needs to be done to ensure Web scalability and data accessibility (distribution algorithms) in the large, suggesting further research in innovative areas like distributed semantic querying, semantic clustering and self-organizing systems.

We expect that with further development and publication in this area of semantic tuplespace computing, which has already started (Martín-Recuerda, 2006; Fensel *et al.*, 2007; Nixon *et al.*, 2007), we can count on ongoing integration of this research into the Semantic Web architecture as a coordination middleware, particularly for machine-to-machine communication.

## References


Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. 2003 *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge, UK: Cambridge University Press.

Berners-Lee, T. 2004 *N3QL - RDF Data Query Language, W3C Design Issues*. http://www.w3.org/DesignIssues/N3QL.html.

Berners-Lee, T. 2005 *Primer: Getting into RDF & Semantic Web using N3, Website v1.61*. http://www.w3.org/2000/10/swap/Primer.html.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001 The semantic web. *Scientific American* **284**(5), 34–43.

Bray, T., Hollander, D., Layman, A. & Tobin, R. 2006 *Namespaces in XML 1.0*, 2nd edn., W3C Recommendation.

Brickley, D. & Guha, R. 2004 *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Recommendation.

Busi, N., Gorrieri, R. & Zavattaro, G. 2000a Comparing three semantics for Linda-like languages. *Theoretical Computer Science* **240**(1), 49–90.

Busi, N., Gorrieri, R. & Zavattaro, G. 2000b On the expressiveness of Linda coordination primitives. *Information and Computation* **156**(1–2), 90–121.

Busi, N. & Zavattaro, G. 2000 On the expressiveness of event notification in data-driven coordination languages. *Lecture Notes in Computer Science* **1782**, 41–55.

Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V. & Pedrinaci, C. 2006 IRS-III: A broker for semantic web services based applications. In *Proceedings of the 5th International Semantic Web Conference*. Athens, GA, USA: Springer Verlag, pp. 201–214.

Cabrera, F., Copeland, G., Freund, T., Klein, J., Langworthy, D., Orchard, D., Shewchuk, J. & Storey, T. 2002 *Web Services Coordination (WS-Coordination)*, http://msdn.microsoft.com/ws/2002/08/WSCoor.

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

Carroll, J., Bizer, C., Hayes, P. & Stickler, P. 2005a Named graphs. *Journal of Web Semantics* **3**(4), 247–267.

Carroll, J., Bizer, C., Hayes, P. & Stickler, P. 2005b Named graphs, provenance and trust. In *Proceedings of the 14th International World Wide Web Conference*. Chiba, Japan: ACM Press, pp. 613–622.

Chinnici, R., Moreau, J.-J., Ryman, A. & Weerawarana, S. 2007 *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C Recommendation.

Ciancarini, P., Knoche, A., Tolksdorf, R. & Vitali, F. 1996 PageSpace: an architecture to coordinate distributed applications on the web. *Computer Networks and ISDN Systems* **28**(7–11), 941–952.

Ciancarini, P., Tolksdorf, R. & Zambonelli, F. 2003 Coordination middleware for XML-centric applications. *Knowledge Engineering Review* **17**(4), 389–405.

de Bruijn, J., Martin-Recuerda, F., Manov, D. & Ehrig, M. 2004 *State-of-the-art survey on Ontology Merging and Aligning V1, Project Deliverable d4.2.1, 2004*. SEKT project IST-2003-506826. http://sekt.semanticweb.org/.

Euzenat, J. 1995 Building consensual knowledge bases: context and architecture. In Mars, N. (ed.), *Building and Sharing Large Knowledge Bases*. Amsterdam: IOS Press, pp. 143–155.

Fensel, D. 2004 Triple-space computing: semantic web services based on persistent publication of information. In *Proceedings of the IFIP International Conference on Intelligence in Communication Systems INTELLCOMM 2004*. Bangkok, Thailand: Springer-Verlag, pp. 43–53.

Fensel, D. & Bussler, C. 2002 The web service modeling framework WSMF. *Electronic Commerce Research and Applications* **1**(2), 113–137.

Fensel, D., Krummenacher, R., Shafiq, O., Kuehn, E., Riemer, J., Ding, Y. & Draxler, B. 2007 TSC—Triple Space Computing. *e&i Elektrotechnik und Informationstechnik* **124**(1/2), 31–38.

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

Fikes, R., Hayes, P. & Horrocks, I. 2005 OWL-QL A Language for deductive query answering on the semantic web. *Journal of Web Semantics* **2**(1), 19–29.

Fitting, M. 1996 *First-Order Logic and Automated Theorem Proving*, New York: Springer-Verlag New York, Inc.

Freeman, E., Arnold, K. & Hupfer, S. 1999 *JavaSpaces Principles, Patterns, and Practice*. The Jini Technology Series. Essex, UK: Addison-Wesley Longman Ltd.

Gelernter, D. 1985 Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* **7**(1), 80–112.

Gelernter, D. & Carriero, N. 1992 Coordination languages and their significance. *Communications of the ACM* **35**(2), 97–107.

GigaSpaces(TM) Technologies Ltd. 2002 GigaSpaces Platform—White Paper. http://www.gigaspaces.com/.

Groot, P., Hitzler, P., Horrocks, I., Motik, B., Stuckenschmidt, J. P. H., Turi, D. & Wache, H. 2005 D2.1.2 Methods for Approximate Reasoning. Knowledge Web Project Deliverable. http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.1.2.pdf

Grosof, B., Horrocks, I., Volz, R. & Decker, S. 2003 Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th International World Wide Web Conference*. Budapest, Hungary: ACM Press, pp. 48–57.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J-J., Nielsen, H., Karmarkar, A. & Lafon, Y. 2007 *SOAP Version 1.2 Part 1: Messaging Framework* (2nd ed.), W3C Recommendation.

Haarslev, V. & Möller, R. 2001 Description of the RACER system and its applications. In McGuinness, D. L., Goble, C., Möller, R. and Patel-Schneider, P.F (eds), *Proceedings of the International Workshop on Description Logics*, Aachen, Germany: CEUR Workshop Proceedings, pp. 131–141.

Harth, A. & Decker, S. 2005 Optimized index structures for querying RDF from the web. In *Proceedings of the 3rd Latin American Web Congress*. Buenos Aires, Argentina: IEEE Computer Society, pp. 71–80.

Hayes, P. & McBride, B. 2004 *RDF Semantics*, W3C Recommendation.

Hull, R. & Zhou, G. 1996 A framework for supporting data integration using the materialized and virtual approaches. In *Proceedings of the ACM SIGMOD '96*, Montreal, Canada, pp. 481–492.

Cheng, J., Gruninger, M., Sriram, R. D. & Law, K. H. 2003 Process specification language for project information exchange. *International Journal of Information Technology in Architecture, Engineering and Construction*, **1**(4), 307–328.

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

Khushraj, D., Lassila, O. & Finin, T. W. 2004 sTuples: semantic tuple spaces. In *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, USA: IEEE Press, pp. 268–277.

Klyne, G. & Carroll, J. J. 2004 *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation.

Kotis, K. & Vouros, G. A. 2003 Human centered ontology management with HCONE. In *Proceedings of the IJCAI'03, Ontologies and Distributed Systems Workshop*, Acapulco, Mexico. CEUR-WS.org/Vol. 71, ISSN 1613-0073. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-71/

Krummenacher, R., Ding, Y., Kilgarriff, E., Sapkota, B. & Shafiq, O. 2006 D1.3 Specification of Mediation, Discovery and Data Models for Triple Space Computing, TSC Project Deliverable. http://tsc.deri.at/deliverables/D13.html.

Krummenacher, R., Hepp, M., Polleres, A., Bussler, C. & Fensel, D. 2005 WWW or What is Wrong is with Web Services. In *Proceedings of the 3rd European Conference on Web Services*. Växjö, Sweden: IEEE Press, pp. 235–243.

Krummenacher, R., Simperl, E., Nixon, L., Cerizza, D. & Valle, E. D. 2007 Enabling the European Patient Summary Through Triplespaces. In *20th IEEE International Symposium on Computer-based Medical Systems*.

Kühn, E. 2001 *Virtual Shared Memory for Distributed Architectures*. Commack, NY, USA: Nova Science Publisher.

Lara, R., Lausen, H., Arroyo, S., de Bruijn, J. & Fensel, D. 2003 Semantic web services: description requirements and current technologies. In *Proceedings of the International Workshop on Electronic Commerce, Agents, and Semantic Web Services at ICEC 2003*. Pittsburgh, PA, USA.

Lehman, T., McLaughry, S. & Wyckoff, P. 1999 T spaces: the next wave. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-32)*. Maui, Hawaii: IEEE Press, p. 8037.

MacGregor, R. & Ko, I-Y. 2003 Representing contextualized data using semantic web tools. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems*, Sanibel Island, FL, USA: CEUR Workshop Proceedings.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T. et al. 2004 *OWL-S: Semantic Markup for Web Services*, W3C Member Submission. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

Martín-Recuerda, F. 2005 Towards CSpaces: A new perspective for the Semantic Web. In *Proceedings of the 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web*. University of Jyväskylä Finland. Berlin Heidelberg, Germany: Springer Verlag.

Martín-Recuerda, F. 2006 Application Integration Using Conceptual Spaces (CSpaces). In *Proceedings 1st Asian Semantic Web Conference*. Beijing, China. Berlin Heidelberg, Germany: Springer-Verlag, pp. 300–306.

Bonifacio, M., Bouquet, P., & Cuel, R. 2002 Knowledge nodes: the building blocks of a distributed approach to knowledge management. *Journal of Universal Computer Science* **8**(6), 652–661.

McGuinness, D. L., Fikes, R., Hendler, J. & Stein, L. A. 2002 DAML+OIL: An Ontology Language for the Semantic Web, *IEEE Intelligent Systems* **17**(5), 72–80.

McGuinness, D. & van Harmelen, F. 2004 *OWL Web Ontology Language Overview*, W3C Recommendation.

Merrick, I. & Wood, A. 2000 Coordination with scopes. In *Proceedings of the ACM Symposium on Applied Computing*, ACM Press, pp. 210–217.

Mocan, A., Zaremba, M., Moran, M. & Cimpian, E. 2006 Filling the Gap Extending Service Oriented Architectures with Semantics. In *Proceedings of the 2nd IEEE International Symposium on Service-Oriented Applications, Integration and Collaboration*. Shanghai, China: IEEE Press.

Nixon, L. J. B., Simperl, E. P. B., Antonenko, O. & Tolksdorf, R. 2007 Towards Semantic Tuplespace Computing: The SemanticWeb Spaces System. In *Proceedings of the 22nd ACM Symposium on Applied Computing, Track 'Coordination Models, Languages and Applications'*. Seoul, Korea: ACM Press.

Omicini, A., Zambonelli, F., Klusch, M. & Tolksdorf, R. (eds.) 2001 *Coordination of Internet Agents: Models, Technologies, and Applications*. Berlin, Germany: Springer Verlag.

Papadopoulos, G. & Arbab, F. 1998 Coordination models and languages. In *Advances in Computers, Vol. 46: The Engineering of Large Systems*. New York, USA: Academic Press.

Picco, G. P., Balzarotti, D. & Costa, P. 2005 LIGHTS: a lightweight, customizable tuple space supporting context-aware applications. In *Proceedings of the 20th ACM Symposium on Applied Computing*. Sante Fe, NM, USA: ACM Press, pp. 1134–1140.

Prud'hommeaux, E. & Seaborne, A. 2006 *SPARQL Query Language for RDF*. W3C Working Draft.

Rossi, D., Cabri, G. & Denti, E. 2001 Tuple-based technologies for coordination. In *Coordination of Internet Agents: Models, Technologies, and Applications*. Berlin, Germany: Springer-Verlag, pp. 83–109.

Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Lubell, J. & Lee, J. 1999 *The Process Specification Language (PSL): Overview and Version 1.0 Specification. NIST Internal Report (NISTIR) 6459*, National Institute of Standards and Technology, Gaithersburg, MD, USA.

Schlobach, S. & Cornet, R. 2003 Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico. San Francisco, CA, USA: Morgan Kaufmann, pp. 355–362.

Shafiq, O., Toma, I., Krummenacher, R., Strang, T. & Fensel, D. 2006 Using triple space computing for communication and coordination in semantic grid. In *Proceedings of the 3rd Semantic Grid Workshop in conjunction with the 16th Global Grid Forum*. `http://www.semanticgrid.org/GGF/ggf16/papers/TSC-semgrid_20060129.pdf`

Sivashanmugam, K., Verma, K., Sheth, A. & Miller, J. 2003 Adding semantics to web services standards. In *Proceedings of the 1st International Conference on Web Services*. Las Vegas, NV, USA: CSREA Press, pp. 395–401.

Heymans, S., Krummenacher, R., Martin-Recuerda, F., Nixon, L. J. B., Paslara Bontas Simperl, E. & Scicluna, J. 2007 D2.4.8 v2: Semantic TupleSpace Computing. Knowledge Web Project Deliverable. `http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.4.8.pdf`.

Suryanarayana, G. & Taylor, R. 2004 *A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications, Technical Report UCI-ISR-04-6, Institute for Software Research, University of California*, Irvine.

Tolksdorf, R. & Menezes, R. 2003 Using swarm intelligence in Linda systems. In *Proceedings of the 4th International Workshop Engineering Societies in the Agents World ESAW'03*. London, UK. Berlin Heidelberg, Germany: Springer Verlag, pp. 49–65.

Tolksdorf, R., Nixon, L., Paslaru Bontas, E., Nguyen, D. M. & Liebsch, F. 2005a Enabling real world SemanticWeb applications through a coordination middleware. In *Proceedings of the 2nd European Conf. on the Semantic Web*. Heraklion, Crete, Greece, pp. 679–693.

Tolksdorf, R., Paslaru Bontas, E. & Nixon, L. 2005b Towards a tuplespace-based middleware for the SemanticWeb. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence WI2005*. IEEE Computer Society, pp. 338–344.

Tolksdorf, R., Paslaru-Bontas, E. & Nixon, L. 2006 A co-ordination model for the Semantic Web. In *Proceedings of the 21st ACM Symposium on Applied Computing, Track 'Coordination Models, Languages and Applications'*. ACM Press, pp. 419–423.

Ullman, J. D. 1997 Information integration using logical views. *Theoretical Computer Science* **239**(2), 189–210.

Wells, G., Chalmers, A. & Clayton, P. 2004 Linda implementations in Java for concurrent systems. *Concurrency and Computation: Practice and Experience* **16**(10), 1005–1022.

Werthner, H., Hepp, M., Fensel, D. & Dorn, J. 2006 Semantically-enabled Service-oriented Architectures: a catalyst for smart business networks. In *Proceedings of the Smart Business Networks Initiative Discovery Session*. Rotterdam, The Netherlands.

Wyckoff, P., McLaughry, S., Lehman, T. & Ford, D. 1998 Tspaces. *IBM Systems Journal* **37**(3), 454–474.