A Coordination Model for Triplespace Computing

Elena Simperl[1,2], Reto Krummenacher[2], and Lyndon Nixon[1]

[1] Free University of Berlin, Germany
[2] DERI, University of Innsbruck, Austria

simperl@inf.fu-berlin.de, reto.krummenacher@deri.org, nixon@inf.fu-berlin.de

Abstract. Recent advances in middleware technologies propose semantics-aware tuplespaces as an instrument for coping with the requirements of *scalability*, *heterogeneity* and *dynamism* arising in highly distributed environments such as the Web or the emerging Semantic Web. In particular, Semantic Web services have inherited the Web service communication model, which is based on synchronous message exchange, thus being incompatible with the REST architectural model of the Web. Analogously to the conventional Web, truly Web-compliant service communication should, however, be based on persistent publication instead of message passing. This paper reconsiders *"triplespace computing"*, a coordination middleware for the Semantic Web. We look at how a coordination model for triplespace systems could look like - in order to manage formal knowledge representations in a space and to support the interaction patterns characteristic for the Semantic Web and Semantic Web services - as a precursor to the design and implementation of a triplespace platform in the context of the TripCom project. [1]

1 Introduction

The Semantic Web will have a significant impact on the next generation of worldwide network information systems. In order to build semantic applications, some middleware is necessary to offer uniform access to distributed information. This middleware differs from similar technologies for Web-based systems in that it incorporates means to cope with the machine-processable semantics of Web resources. Moreover, Semantic Web services - a core building block of the Semantic Web - have inherited the Web service communication model, which is based on synchronous message exchange, thus being incompatible with the Representational State Transfer (REST) architectural model of the Web [15]. In order to ensure truly Web-compliant service communication, there is need for a middleware solution that is based, analogously to the conventional Web, on persistent information publication instead of message passing, thus allowing services to exchange information in a time and reference decoupled manner.

In this paper we reconsider *"triplespace computing"*, a coordination middleware for the Semantic Web. Triplespace systems are able to manage information formalized using Semantic Web representation languages and to coordinate information exchange among agents processing this information. However, this new application scenario imposes several revisions of the Linda model. New coordination primitives as well as new

[1] TripCom (IST-4-027324-STP): http://www.tripcom.org.

types of tuples and spaces are needed in order to enable tuplespaces to deal not only with plain data, but also with interpretable information with assigned truth values. Further on, the new approach has to provide optimal support for the interaction patterns characteristic for the communication among Semantic Web services. In this paper we will use the terms *"triplespaces"* and *"triplespace system"* interchangeably to refer to an implementation of the triplespace computing paradigm.

The rest of this paper is organized as follows: Sections 2 and 3 present the basic principles of the Semantic Web and analyze the requirements that need to be satisfied by a (space-based) middleware in order to enable the realization of Semantic Web applications. We elaborate on triplespace computing and the underlying coordination model in Section 4. Section 5 gives an overview of similar initiatives, while Section 6 concludes with a discussion of open issues and future work.

2 An Introduction to the Semantic Web

The Semantic Web is an extension of the current Web in which information is given well-defined, machine-processable meaning, better enabling computers and people to co-operate [2]. It is intended to complement the current World Wide Web with a *network of URI-addressable information*, which is represented and linked in such a way that it can be easily processed by machines both at a *syntactical* and at a *semantical* level. For this purpose Web resources should be annotated with machine-understandable metadata that is formalized by use of common vocabularies with predefined semantics, known as *"ontologies"*. Further on, semantically enriched Web services should be able to process and exchange this information.

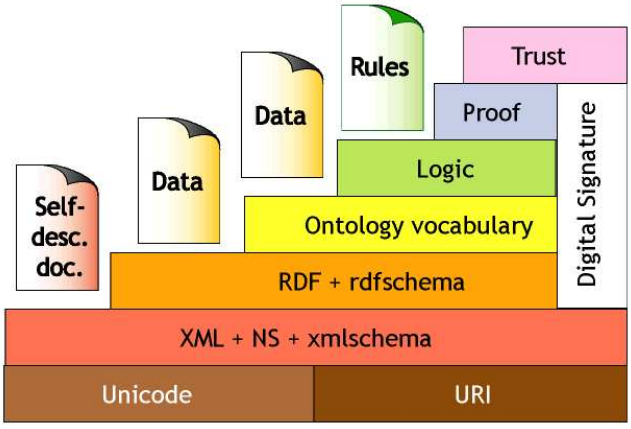

Fig. 1. The Semantic Web stack by Tim Berners-Lee [1]

The first step towards the realization of the Semantic Web have been made through the standardization of representation languages for Web knowledge like RDF [22], RDFS [3] and OWL [26] and the increasing dissemination of ontologies that provide a common basis for annotations. Additionally, recent efforts in the area of Web services propose methods and tools to represent Web services in a Semantic Web compatible manner in order to improve tasks like automatic service discovery or composition [5, 7, 14, 24].

The Semantic Web is built on XML-based syntaxes which use URIs to uniquely identify Web resources (cf. Figure 1). Resources can denote not only common Web documents, but any entity represented within a computer system (e.g. persons, physical objects, RDF statements) and are described by machine-processable metadata that provides data about their properties, capabilities, and requirements. Metadata is then a collection of RDF statements of the type ⟨*subject, predicate, object*⟩, where the three fields can be individually referenced by means of URIs. For exemplification purposes we annotate the present paper (if available as a Web document) with information concerning its author, content etc. Examples of such RDF statements could be:

The email address of Elena Simperl, one of the authors of this paper, is "simperl@inf.fu-berlin.de". This information can be represented in RDF in form of the following two triples:

⟨*x:Coordination07Paper, dc:contributor, http://userpage.fu-berlin.de/simperl*⟩
⟨*http://userpage.fu-berlin.de/simperl, adr:mail, "simperl@inf.fu-berlin.de"*⟩

The subject of the first statement is this paper, identified by an imaginary URI *x:Coordination07Paper*. The predicate *dc:contributor* is part of the Dublin Core (DC) metadata scheme and denotes a standardized property, while the object *http://userpage.fu-berlin.de/simperl*, the value of the property, represents the co-author Elena Simperl. The object in an RDF statement can be another RDF resource - represented by a URI, as in the first triple - a literal - represented through a simple XML datatype, as in the second triple - or a blank node. Further on, the example shows the way individual triples are naturally interlinked to RDF graphs: the object of the first statement can be used as subject in subsequent statements.

The contact address of the co-author is "Takustr. 9, 14195 Berlin, Germany". Structured information like addresses can be represented in RDF in aggregated form, by using so-called *"blank nodes"*, which denote anonymous resources, identified by application internal URIs:

⟨*http://userpage.fu-berlin.de/simperl, adr:contact, _:bn1*⟩
⟨*_:bn1, adr:street, "Takustr."*⟩
⟨*_:bn1, adr:number, "9"*⟩
⟨*_:bn1, adr:zip, "14195"*⟩
⟨*_:bn1, adr:city, "Berlin"*⟩

Here *_:bn1* identifies an anonymous resource which aggregates the address-related data, i.e. there is some entity that is characterized by an *adr:street*, an *adr:number*, etc.

RDF is considered to be the standard interchange format of the Semantic Web and is intended to be used as a simple yet powerful annotation language for Web resources. The next layers on Figure 1 add logically more expressive languages for ontologies (e.g. OWL) and support for rules (e.g. SWRL [19]). RDFS and OWL are used to define common vocabularies for metadata which enable interoperability among applications. Besides, they re-define Web resources in terms of classes and properties with a well-founded semantics which can be exploited by reasoners to validate the models and to automatically generate implicit knowledge. We demonstrate the usage of ontologies and ontology representation languages by extending the previous examples with additional information formalized in RDFS and OWL. In that way we align the local terms to external vocabularies. This provides a more precise specification of the underlying domain, thus forming the basis for more powerful information retrieval in a fictive publication repository.

This paper is a special kind of publication. Specialization and generalization hierarchies can be built in RDFS and OWL. A *Paper* is a special type of *Publication*:

⟨*x:Paper, rdfs:subClassOf, x:Publication*⟩

Further on, one can link ontological information to concrete instance data; the statement

⟨*x:Coordination07Paper, rdf:type, x:Paper*⟩

specifies that the current document is a particular instance of the class *Paper* in our imaginary publications ontology. In this way a query looking for publications of a given author would match not only to the documents which are explicitly defined (through annotations) as publications, but also to specific publication types such as conference papers or journal articles.

The concept* Publication *in the local ontology with the namespace 'x' is equivalent to the concept* Text *in the Dublin Core vocabulary. Equivalence relationships between entities can be expressed in OWL:

⟨*x:Publication, owl:sameClassAs, dc:Text*⟩

Aligning the personal ontology to a standard model such as Dublin Core is the first step towards increased syntactic and semantic interoperability. Ensuring syntactic compatibility enables more flexible retrieval services, which are then able to go beyond simple string matching. The semantic interoperability allows applications handling the local ontology to better understand the meaning of the corresponding concept, since the equivalence between concepts implies also an inheritance relationship by means of which the local concept is enriched with the properties externally defined for its equivalent.

A paper consists of several sections. The domain model is further refined in order to define typical (physical or conceptual) parts of a scientific paper. As in the previous example, extending the model supplies machines with deeper background knowledge on the application domain, thus enabling advanced, domain-tailored behavior. From a modeling point of view we introduce a new property *partOf* which relates entities to their parts:

⟨*x:Section x:partOf x:Paper*⟩

In addition to specifying their domain and range, one can refine the semantics of the domain properties by specifying features such as transitivity, symmetry or cardinality constraints.

The remaining layers of the Semantic Web are still at a much more immature stage. However, issues of proof and trust are vital to the success of the Semantic Web since one needs techniques to verify the quality of the created metadata. The proof layer (cf. Figure 1) is intended to provide languages and tools to prove whether statements created by arbitrary authors are true or not. The trust layer addresses the same issue from a different perspective. It should define mechanisms which, in correlation with digital signatures, enable the definition of provenance and reputation information for resource metadata.

3 Requirements for a Semantics-Aware Middleware

With the growing importance of open distributed systems, in particular the World Wide Web, new requirements arose from the way services communicate and coordinate their access to shared information. The coordination between clients in such open environment is more complex as a system cannot know in advance which clients will use it, nor which characteristics the active clients might have. For example, clients may not necessarily agree in advance on shared models, protocols and types for the data exchanged. It may become the task of the middleware to (partially) resolve such heterogeneities. For open distributed systems these twin issues of dynamism and heterogeneity need to be taken into account and are furthermore tightly related to the problem of scalability. One answer to these issues is the application of semantic technologies to enhance the descriptions of clients, services, protocols and data for example in form of the aforementioned *Semantic Web services*.

The advantage of adding semantics to service, protocol and data descriptions is that the interfaces to the different agents, services and information sources can be dynamically generated, while the semantics of the exchanged information is formally defined. Consequently there are new means for the automatization of service and data discovery, mediation and service composition. This suggests that (Semantic) Web services provide a good communication model for distributed computing scenarios, as envisioned by service-oriented architectures [10]. However, Semantic Web service infrastructures do not provide support for persistent publication of data, nor for time decoupling of messages so that data can outlive the services publishing or consuming it [11].

These features however are the characteristic of Linda-based systems [17]. Yet, applying tuplespaces to the open global environment of the (Semantic) Web raises new requirements [11, 20]:

– 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.
– a separation mechanism. Distributed applications which have independent naming schemes may use the same names for their resources. On the Web, vocabularies can be kept separate - even when using the same terms - by help of a namespaces mechanism. The concept of namespaces should be handed down to tuplespace systems as well in order to inherit the Web functionality.
– the nesting of tuples. Web data models such as XML and RDF permit the nesting of elements within a single document, i.e. RDF data adds to RDF Graphs. Likewise tuples should be able to explicitly show how information units are interlinked.

This gives evidence that the coordination model of Linda needs to be rethought and extended in order to meet the requirements of Web-scale systems. A coordination model for the Semantic Web and Semantic Web services must be able

– to support autonomous activity of participants by decoupling interactions in time and reference - two agents are neither required to be available concurrently, nor must they know each others location or address,
– to process semantic information as the data being coordinated, and
– to provide access to large amounts of heterogeneous data that is dispersed over broad and dynamic networks.

In spite of these complex requirements it is clear that combining Linda models and semantic technologies introduces a new and powerful communication paradigm that provides the desired grounds for persistent, asynchronous and anonymous dissemination of machine-understandable information. This communication paradigm has been referred to as *"triplespace computing"* [11]. The target application scenarios for this novel middleware approach range from sharing information on the Semantic Web, distributed knowledge management and pervasive computing to a fully fledged communication and coordination platform for Semantic Web services or the Semantic Grid [32]. There, the potential application areas are just as diverse: Enterprise Application Integration (EAI), eHealth, digital multimedia systems and recommender systems, to only mention a few.

3.1 Required Extensions to Linda or Why Linda Is Not Enough

In order to comply the requirements of large scale Semantic Web applications Linda must be extended in several directions. We distinguish two categories of extensions to the original approach: (1) new types of tuplespaces, and (2) new types of tuples.

The former category aims at overcoming the technical problems of large scale distributed systems (e.g. heterogeneity, scalability, fault-tolerance, multiuser access) by proposing distribution strategies for multiple spaces or hierarchic spaces and augmented naming approaches. Such approaches were already considered in various non-semantic tuplespace implementations [6, 9, 27, 30, 34], as it was recognized that the traditional Linda approach does not suffice in the large. The second research direction looks at the

necessary extension to tuples and tuple fields. The Semantic Web is mainly about the meaning of Web resources and their properties. The first problem with traditional Linda implementations is that tuples with the same number of field and the same field typing cannot be distinguished, which does not comply with the Semantic Web principle which foresees that all information is encoded in triples of URIs. Moreover, as we have seen in Section 2, the different semantic tuples in a space are not independent as in Linda, but highly correlated and depending on each other. Hence, the Semantic Web requires a reconsideration of the tuple model and the way tuples are matched in accordance to these concepts. The matching algorithms have to consider the meaning of the semantic tuples in order to provide the required degree of knowledge interpretation, and allow the retrieval of tuples taking into account the relationship to other tuples in the same space. This highly relates to the already discussed issues of resource identification – reference mechanism, namespace mechanism – which must be supported by the space in order to allow semantic data to be represented according to the specification of RDF.

Several research projects have already arisen that work on the specification and implementation of such extensions. We provide a short description of these projects in Section 5, but first we elaborate on the TripCom approach to semantics-aware tuplespaces.

4 Triplespace Computing in TripCom

The core vision of triplespace computing is to establish the Web paradigm of *"persistently publish and read"* for the Semantic Web and Semantic Web services. Currently these ideas are further conceptualized and implemented within the EU project TripCom. This section describes the basic concepts of this approach, as a precursor for the design and implementation of a TripCom triplespace system. First, it is necessary to revisit the definition of tuples and spaces, as it is indispensable to adapt these concepts to the norms of the Semantic Web. Thereafter we concentrate on the description of the required coordination primitives in order to make the Linda operations compatible to the requirements of the Semantic Web and Semantic Web services.

4.1 New Types of Tuples and Tuplespaces

Following the Linda paradigm a triplespace system should be able to represent *semantic information* through *tuples*. The expressivity of the information representation should be aligned to the expressivity of common Semantic Web languages, while respecting their semantics, so that tuples could be mapped to and from external Semantic Web resources. Regarding Semantic Web languages, we currently focus on RDF. RDF statements can be represented in a three fielded tuple (so-called *"triples"*) of the form $\langle$*subject, predicate, object*$\rangle$. Following the RDF abstract syntax each tuple field contains an URI (or, in the case of the object also a literal). Tuples can be addressed by means of URIs too, which are defined through the triplespace ontology (cf. Section 4.3). In this way tuples sharing the same subject, predicate and object can be addressed separately, which is consistent with the Linda model. If a tuple is removed from and reinserted into

the space, it is allocated with a new URI, as the reinserted tuple is regarded as a new tuple despite its relatedness with the one previously deleted. When performing reasoning however, tuples sharing the same content are interpreted as duplicates despite different identifiers, as foreseen by the RDF semantics [18].

A second issue to be considered in this context is the way sets of related statements (i.e. RDF graphs) are represented at the level of tuples. Since both RDF statements and graphs are associated to URIs in the triplespace ontology (cf. Section 4.3), we can specify the membership relation through an additional triple using a dedicated meta-property: ⟨tupleURI p:partOf graphURI⟩. Moreover, the Triplespace API foresees a series of operations allowing the manipulation of RDF graphs (cf. Section 4.2).

The model of a semantic tuple can be further refined in terms of the knowledge representation language whose semantics is relevant for processing the tuple content in relation with matchings. As the prospected system foresees a semantic matching behavior in addition to the classical Linda procedures, it might be useful to differentiate between tuples embedding RDF, RDFS, OWL or WSML [12] data. While this distinction does not affect the basic tuple model, which remains a set of three fields, it triggers the usage of particular matching procedures, and thus needs to be stored in the space. Again, the triplespace ontology can be an instrument to capture this type of metadata about the tuples and their content. The coordination operations take into account this metadata in order to enforce the execution of a particular matching algorithm.

A triplespace is defined as a container for triples which encapsulate the RDF statements. A triplespace can be divided into virtual subspaces and physically partitioned across distributed kernels. Every space is addressed using a URI, which is installed by the creating user and captured in the triplespace ontology. Just as in the case of individual tuples, this URI is useful in terms of the REST communication model. A space may contain multiple (sub-)spaces, while it can only be contained in at most one parent space. The latter holds also for tuples, which are associated to a single space. In order to allow for overlapping between space the triplespace model resorts to the notion of *"scopes"* [28]. Scopes are temporary tuple containers. Unlike subspaces, which form the virtual structure of the triplespaces, they can be created individually by clients based on arbitrary filters. Scopes can be given different semantics, e.g. they could be seen as an alternative view on the structure of the tuplespace, or as a temporary copy of some tuples for retrieval by a client - in the latter case insertion and deletion operations would apply to the scope and not to the tuplespace as a whole. We note two final differences between spaces and their temporary counterparts: while spaces are defined to be non-overlapping, the notion of scopes does not impose this restriction. Consequently a tuple can be contained in a space - and implicitly in all the direct and indirect parent spaces of the original space - and in a multitude of possibly independent scopes (cf. [28]). The scopes are furthermore not stored in the triplespace ontology, which captures meta-aspects on *persistent* data containers such as graphs or spaces (cf. Section 4.3). Support for scopes is subject of ongoing work also at the level of the Triplespace API, which is introduced in the next section.

4.2 New Coordination Primitives

The Linda operations *out*, *in* and *rd* form the basis for any tuplespace implementation. The basic tuplespace primitives have however soon proven to be insufficient in various application contexts, and implementations of tuplespace platforms based on Linda have liberally extended the coordination language for their needs. We note that extensions to the coordination language can take different forms:

- the extension of the coordination primitives, i.e. the specification of new operations
- the extension of the coordination semantics, i.e. the redefinition of the meaning of the existing operations
- the extension of the coordination model, i.e. providing other instruments for expressing coordination such as programmability.

In this section we introduce the operations defined for the triplespace coordination model and thus our extensions to the Linda coordination primitives and their meaning in this context. We take a look at the semantics and the signatures of the operations and discuss the rationales for taking certain design decisions. An outline of the Triplespace API is given in Table 1.

The core operations are defined around the traditional Linda primitives. However, due to the characteristics of RDF, where triples are not independent data containers, but rather sets of interlinked ⟨*subject, predicate, object*⟩-tuples, it was necessary to extend the semantics of the operations. The *out* operation received a multi-write characteristics in order to allow the publication of whole RDF graphs within one call to the space. A similar operation termed *multiWrite* was for instance already foreseen in TSpaces [23].

With respect to tuple retrieval we differentiate three cases depending on the scope of the returned data. A retrieval call to the space does not return a single tuple/triple, but rather a whole RDF construct. The way such RDF constructs are created is seen to be an implementation detail (e.g. based on the Concise Bounded Descriptions approach [33]). This decision is motivated by the fact that a single RDF triple hardly has any meaning by itself. Reconsidering the examples in Section 2 it can be seen that already very simple statements consist of several triples that isolated would not communicate any useful information.

The primitive *rda* retrieves a single triple (read a triple) and the triples directly bound to it. This is hence the Semantic Web version of the Linda *rd* operation. The more frequently used operation is expected to be the plain *rd*, therefore also the simpler name. *rd* returns an undetermined number of triples and its neighbors. In consequence *rd* is an information retrieval operation. The amount of information (the number of triples) returned is limited by the indeterminism of the space platform and the possible incomplete set of visible nodes, i.e. *rd* does not guarantee that all matching information in a space is returned, although it aims for it. This conforms well to the original Linda definition and is clearly manifested by the open nature of the (Semantic) Web and the open world assumption of Semantic Web languages such as OWL. The third retrieval primitive called *rdg* is used to search for whole graphs (read graph). As explained previously, the triplespace *out* operation accepts a triple set (i.e. RDF graphs), which can optionally be associated with an identifier (a graph URI) to construct a so-called named graph [8] as an input. The *rdg* operation allows the retrieval of a single graph that contains at least

out(Graph g, URI space, [URI graph, URI transaction]):void Inserts the triples included in the graph into the given space. By specifying a graph identifier a named graph is created. A transaction identifier can be provided to add the out to a given active transaction.
rda(Template t, [URI space, URI transaction, integer timeout]):Graph Returns *one* matching triple with any triples bound to it, e.g. following the Concise Bounded Descriptions approach. The request is executed against the given space, if provided, otherwise against the virtual global triplespace. The timeout is used as means to control the blocking characteristics of rda.
rd(Template t, [URI space, URI transaction, integer timeout]):Graph This operation generalizes rda: it returns an undetermined number of matching triples and their bound graphs; otherwise the functionality is the same.
rdg(Template t, [URI space, URI transaction, integer timeout]):Graph Returns the entire content of a named graph that contains a matching triple; used to coordinate whole objects.
ina(Template t, [URI space, URI transaction, integer timeout]):Graph This is the destructive version of rda.
in(Template t, [URI space, URI transaction, integer timeout]):Graph This is the destructive version of rd.
ing(Template t, [URI space, URI transaction, integer timeout]):Graph This is the destructive version of rdg.
subscribe(Template t, URI space, Listener l, [URI transaction]):URI Establishes a notification mechanism for triples matching the template. Subscriptions must be expressed against a given space. In case of a match the listener (e.g., in Java a class that is called in case of an event) is notified. The operation returns a handle to the successfully registered subscription in form of a URI.
unsubscribe(URI subscription, [URI transaction]):boolean This operation cancels a given subscription and returns true in case of successful execution.
create(URI space, [URI parent, URI transaction]):boolean This primitive creates a new space, as subspace of parent. In case no parent space is indicated the new space is installed as direct child of the virtual global space. It returns true after successful creation.
destroy(URI space, [URI transaction]):boolean This operation destroys the given space, its subspaces and all contained triples. Particular attention has therefore to be paid to rights management to avoid unauthorized removals.

Table 1. The Triplespace API

one triple matching the template. This is particularly useful when coordinating Web service descriptions or purchase orders, where it is indispensable that all knowledge about one construct is retrieved at all times. To ensure that all data of a graph can be read at once, the system will store RDF graphs at the same physical location and hence they are either seen as a whole or not visible to a particular user at all. The three operations are also available in destructive mode. In this case they are referred to as *ina*, *in* and *ing*, respectively.

The matching procedure for the six operations is based on templates. The precise syntax and semantics of a templates depends on the maturity of the space implementation and on the query languages and engines employed in the implementation; we generally refer to it as template in order to proceed with a stable interaction model. In the current release (March 2007) we use simple triple patterns that are very close to traditional Linda templates. A template is hence a tuple that contains both RDF resources and variables: ⟨*x:Coordination07Paper ?p ?o*⟩. In the future templates will consist of graph patterns (cf. Table 2) as common to most RDF query languages, e.g. SPARQL [31]. Eventually, we expect the templates to evolve to fully fledged semantic queries or rules, which invoke reasoning engines and operate on asserted and inferred triples. In that way we can guarantee that the matching algorithms fully exploit the semantics of the information published. Even though it is clear that this enhanced matching algorithms will be more complex than pure Linda matching, we argue that it provides a good compromise between simplicity and matching at the level of meaning.

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.

Table 2. Examples of Semantic Templates

Though retrieval by identifier does not correspond to the core Linda principle of associative addressing, it was previously considered in various tuplespace implementations such as for instance by the method *readTupleById* in TSpaces. As mentioned in Section 3 the Web uses unique identifiers (URI) to reference resources. The triplespaces are envisioned to provide a shared information space for the Semantic Web that inherits these basic notations regarding resource identification, and thus it was a natural choice to incorporate means to use URIs for retrieval. In TripCom this behavior is provided by use of metadata (cf. Section 4.3) that yields ontological descriptions of the triplespace contents. The name of graphs, the context of data and their interrelationships are modeled and stored in the triplespace ontology. The knowledge captured by this meta-model can be used to refine queries and thus also to query RDF graphs by their name if adequate: *match all triples belonging to the graph with identifier X*.

Furthermore, the retrieval operations will have a field of type integer for the specification of timeouts, and support for transactions. The timeout is used to control the blocking disposition of triplespace requests. A positive integer interrupts the search process after the provided number of seconds passed. If no timeout is specified the operation runs in the regular blocking mode. Note that this is particulary unadvised for *rd* calls due to its multi-search trait that does not come with a automatic retrieval interruption after a first discovery. Using *rd* without timeout results in data acquisition from the virtual global triplespace and this might eventually not terminate.

Transaction support was already added in tuplespace systems such as JavaSpaces [16] and TSpaces. Transactions play an important role in Web service communication and hence this extension was taken over to the coordination model of the triplespace. The first release does not yet integrate any support for transaction, while we plan to integrate transactions locally in a next step, i.e. a given transaction only wraps subsequent calls at a single access node. Eventually we want to install distributed transactions over multiple triplespace nodes too. This allows for transactional safe workflow executions involving multiple agents. Just as we do not guarantee that all data in a triplespace is visible at all times, due to physical or logical distribution, we do not guarantee that data encapsulated by transactions are visible to any other user.

The *subscribe* and *unsubscribe* methods are used to incorporate the notification service provided by triplespaces. Notification is an important tool for many interaction patterns and ensures flow-decoupling of concurrent processes. The use of notification is potentially a good compromise between easiness of implementation and expressive power in distributed systems, for which non-blocking operations do not provide appropriate solutions. A subscription is done by emitting a template to an interesting space. The subscription manager will then inform any subscriber about matching triples until a *unsubscribe* call is sent.

The last two operations depicted in Table 1 are management methods used to create and delete spaces. A space is created by giving it a new unique identifier and by possibly attaching it as a subspace to an already existing space. The semantics of *destroy* is more complex, as the removal of a space implies the deletion of all subspaces, and of the contained data. We therefore expect that removal is only allowed to the creator of the space, or at least that it depends on restrictive security measures. Security, privacy and trust measures are entirely neglected in this paper, as they are seen to be orthogonal to the presented concepts and are developed in parallel.

4.3 Triplespace Ontology

The coordination and data models for semantic tuplespaces provide the first building blocks towards the realization of the triplespace computing paradigm [11]. However, the crucial issues of distribution and scalability are only marginally addressed. The use of an ontology that describes the published data, the created spaces and their interrelationships and characteristics is expected to provide support in tackling the additional requirements of the open distributed scenarios targeted by TripCom. Ontology-driven middleware management is in fact seen to be one of the major assets of semantic tuplespaces compared to traditional space frameworks. The use of ontologies provides

sophisticated means for data and infrastructure handling which directly influences and likely improves the necessary distribution and scalability measures.

In this section we introduce first ideas for a triplespace ontology and the expected benefits resulting thereof.

Metadata is an important tool to optimize management tasks, access procedures and, in case of distributed data sources, also to improve the distribution algorithms such as for replication or caching. Knowledge about the relationship of data items and their context allows for more precise and effective handling and faster processing of requests. Relational databases for instance use meta-information about tables, column types and access privileges. Object-oriented databases rely on information about the data structures, while data warehouses and reference information systems provide meta-information about the provenance and quality of information and their sources. If such information is normalized and formalized, i.e. in terms of an ontology, it is possible to reason about it, to infer and combine new facts, validate them or counteract possible ambiguity or incompleteness of information. This becomes particularly evident and useful when dealing with distributed data sources. In other words understanding the semantics of the meta-information is expected to improve and facilitate the management tasks of the system.

In TripCom we define a triplespace ontology for three particular tasks:

- the optimization of access patterns as mentioned in Section 4.2,
- the improvement of performance and scalability of the triplespace middleware by help of enhanced distribution mechanisms, and
- the management of the security and trust framework. The last task addresses issues like the modeling of user roles and access permissions. Further details are out of the scope of this paper.

The ontology modules so far developed are concentrating on the description of triples, graphs and the spaces they are published in. Moreover the ontology addresses the functionalities of the triplespace kernels: language and reasoning support, storage infrastructure, installed query engines. A triplespace kernel (TS Kernel) is the implementation of a triplespace access node. An informal excerpt of the triplespace ontology (classes and properties) is given in Table 3.[2]

In accordance with the definitions in Sections 4.1 and 4.2 we can see that *Data* items are either RDF *Triples* or RDF *Graphs*. As already pointed out in Section 4.1 it is also possible to serialize *formalisms* more expressive than RDF to triples. In order to ensure an adequate interpretation during matching or reasoning we link *Data* items to an identifier (URI) for the respective underlying language. Whenever a piece of data is accessed, if published for the first time, altered or simply read, it is possible to register an *AccessLogEntry*. The log entry contains information about the *Agent* that addressed the data item, as well as the time of access and the type of access. At all times the data is contained in a *Space*. The space itself can be contained in another triplespace within a hierarchy of spaces (built using the *isSubspaceOf* transitive property) and is shared by all nodes running a *Kernel* that manages the access to and the data of the space. Inversely a *Kernel* shares a given set of spaces, relationship which is captured by the

[2] Further details on the triplespace ontology are available at http://www.tripcom.org/ontologies/.

```
Triple :: Data
  partOf Graph

Graph :: Data
  hasPart Triple

Data
  formalism URI
  isContainedIn Space
  hasLogEntry AccessLogEntry
  isManagedAtKernel Kernel

Space
  isSubspaceOf Space
  isSharedAtKernel Kernel

AccessLogEntry
  publisher Agent
  date xsd:dateTime
  type AccessType

Kernel
  sharesSpace Space
  hasQueryEngine QueryEngine
  seeAlsoKernel Kernel

QueryEngine
  language QueryLanguage
  usesRepository Repository
```

Table 3. Excerpt of the Triplespace Ontology

property *sharesSpace*. Moreover kernels are linked to repositories in order to guarantee data persistency. The *Repositories* are accessed via some *QueryEngine* that resolves queries expressed in a particular *QueryLanguage*.

The previous paragraph depicted a short example of what is envisioned with the triplespace ontology. The captured metadata could also be used to refine templates, i.e. one could ask for triples matching a given pattern, but only if they were published by a particular agent.

The more important objective of the ontology is however the scalability and performance of the space installation and the discovery process (latency and quality). A concretization of the algorithms and the required ontology support is however only in its infancy at this stage of the TripCom project. First attempts allow to associate data with a particular kernel by use of the property *isManagedAtKernel*, or to point from one kernel to another (*seeAlsoKernel*) if they share particular characteristics: same type of data, same users, to name only a few possibilities.

In summary, the use of an integrated ontology-based meta information infrastructure brings along two major advantages for triplespace computing:

- the inference framework of the space middleware allows reasoning not only about the application data that is published and consumed by space users, but also about the administrative data (metadata).
- the use of Semantic Web languages, in particular the application of RDF, allows for an integrated platform without additional requirements on the space infrastructure, i.e. the administrative data is processed and stored by the same tools as the application data.

5 Related Work

This paper has described the conceptual model behind triplespaces, a middleware for coordinating knowledge processes on the Semantic Web. We consider this work to be the first comprehensive and well-grounded specification of a model for semantic Linda.

The initial idea of combining tuplespaces and Semantic Web information has been previously proposed in [11]. Subsequent proposals for a semantics-enabled coordination model such as [4, 13, 25] have however failed to address issues covered in this paper such as the particular representation of RDF syntax, the necessary revisions of the coordination primitives and the means for tuplespace partitioning and kernel distribution required to cope with the scalability and dynamism of the Semantic Web. The approach in [25] is deliberately targeted at a much wider range of application areas, which induces a lack of focus on Semantic Web-specific issues in terms of tuple and tuplespace models and the associated coordination operations. [4] proposes a minimal architecture for triplespace computing, but does not provide any details on the types of tuples and tuplespaces required in this context. The TSC project applies coordination principles to realize a communication middleware for Semantic Web services [13]. The approach is built upon an existing co-ordination system which led to many design decisions being simply carried over rather than re-assessed, as we have done, in a Semantic Web context. For example, the access at API level is to Java objects in the space encapsulating RDF graphs, preventing any lower granularity access at the triple level such as here. Further on, the system is targeted at Semantic Web services, while our triplespace approach covers the whole range of application settings on the Semantic Web. It is also unclear to what extent the aforementioned proposals continue to respect the basic principles of Linda, while the triplespace approach is clearly “backwards compatible”.

Semantic Web Spaces [29] was conceived a generic lightweight coordination middleware for sharing and exchanging semantic data on the Web. While the types of tuples and tuplespaces foreseen in Semantic Web Spaces are similar to the approach introduced in this paper, their coordination model introduces a limited set of operations for handling RDF triples, while not considering extensions such as notifications or transactions, which are clearly required on the Semantic Web.

sTuples extended the JavaSpaces platform to support OWL data in tuple fields [21]. However, this approach has not further considered the implications of coordinating Semantic Web information, as we have done. Rather, OWL graphs are exchanged within tuples, and extracted and processed in other systems while in triplespaces we seek to integrate a Semantic Web framework within the system.

A further distinctive feature is the triplespace ontology. The usage of ontologies for middleware management purposes is acknowledged in [13, 25, 29]. However, only Semantic Web Spaces provides a brief outline how such a meta-model could look like. By contrast, our triplespace ontology is the result of a systematic ontology engineering process, carried on in collaboration with several potential users of such a triplespace platform in areas such as eHealth, Semantic Web services and Enterprize Application Integration.

6 Conclusion

In this paper we reconsidered triplespace computing and the envisioned added value of this novel paradigm for the Semantic Web and Semantic Web services with respect to heterogeneity, scalability and dynamism. First we discussed the requirements of a semantics-aware middleware and looked at the necessary extensions and adaptations of the original Linda coordination model with respect to the representation of formal knowledge and the interactions patterns on the Semantic Web. The paper presented moreover the concepts, models and interaction primitives for the triplespace platform of the TripCom project.

The triplespace model extends Linda mainly in what concerns the representation of semantic data in form of RDF triples and graphs. On the one hand it was necessary to revise the syntax and semantics of tuples and templates, on the other we adapted the interaction primitives, *out*, *in* and *rd* to reflect the fact that RDF triples are identifiable resources that presented nested and interlinked knowledge.

In order to make triplespaces scalable on Web-scale, in contrast to traditional approaches that rather focused on corporate and thus small-scale solutions, the tuplespace model had to be revised as well. Moreover, we expect additional support form ontology-driven space management. As matter of fact we expect the application of ontologies to be one of the major assets compared to conventional tuplespace installations.

While the heterogeneity problem is solved by the support for Semantic Web tools and dynamism is implicitly addressed by the inherited features of space-based computing, the scalability issue is still only marginally touched. We will therefore concentrate on mechanisms to tackle the significant challenges of distribution in large scale systems like the World Wide Web, Grid 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 future improvements to the existing semantic tuplespaces. Some of these ideas were already materialized by use of the triplespace ontology, but not yet implemented. We thus expect that upcoming work will take up these ideas, and that solutions for the distribution and scalability issues will be developed around them.

Moreover, the future of the Semantic Web is seen in the integration of rules 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 in triplespace computing.

The triplespace computing concepts and models presented in this paper are the first steps in substantiating the ideas of [11]. We expect that further development will push the integration process of triplespace computing with the Semantic Web architecture, particularly for machine-to-machine communication:

> *The triplespace may become the Web for machines as the Web, based on HTML, became the Web for humans.* [11]

Acknowledgment

This work is funded by the European Commission under the project TripCom (IST-4-027324-STP). The authors would like to thank all members of the consortium for their advice and input, in particular the colleagues of the work packages WP2 and WP3.

References

1. T. Berners-Lee. Semantic Web - XML2000. Talk: http://www.w3.org/2000/Talks/1206-xml2k-tbl/, December 2000.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
3. D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004.
4. Ch. Bussler. A Minimal Triple Space Computing Architecture. In *2nd WSMO Implementation Workshop*, June 2005.
5. L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, B. Norton, V. Tanasescu, and C. Pedrinaci. IRS-III: A Broker for Semantic Web Services based Applications. In *5th Int'l Semantic Web Conf.*, pages 201–214, November 2006.
6. G. Cabri, L. Leonardi, and F. Zambonelli. MARS: a Programmable Coordination Architecture for Mobile Agents. *IEEE Internet Computing*, 4(4):26–35, July 2000.
7. J. Cardoso and A. Sheth. *Semantic Web Services, Processes and Applications*. Springer Verlag, 2006.
8. J.J. Carroll, Ch. Bizer, P. Hayes, and P. Stickler. Named Graphs. *Journal of Web Semantics*, 3(4):247–267, October 2005.
9. P. Ciancarini, A. Knoche, R. Tolksdorf, and F. Vitali. PageSpace: An Architecture to Coordinate Distributed Applications on the Web. *Computer Networks and ISDN Systems*, 28(7):941–952, May 1996.
10. Th. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.
11. D. Fensel. Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In *IFIP Int'l Conf. on Intelligence in Communication Systems*, pages 43–53, November 2004.
12. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, Summer 2002.
13. D. Fensel, R. Krummenacher, O. Shafiq, E. Kuehn, J. Riemer, Y. Ding, and B. Draxler. TSC - Triple Space Computing. *e&i Elektrotechnik und Informationstechnik*, 124(1/2), February 2007.
14. D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer Verlag, November 2006.
15. R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California Irvine, 2000.
16. E. Freeman, K. Arnold, and S. Hupfer. *JavaSpaces Principles, Patterns, and Practice*. The Jini Technology Series. Addison-Wesley Longman Ltd., 1999.
17. D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.
18. P. Hayes and B. McBride. RDF Semantics. W3C Recommendation, February 2004.

19. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004.
20. B. Johanson and A. Fox. Extending Tuplespaces for Coordination in Interactive Workspaces. *Journal of Systems and Software*, 69(3):243–266, January 2004.
21. D. Khushraj, O. Lassila, and T.W. Finin. sTuples: Semantic Tuple Spaces. In *1st Ann. Int'l Conf. on Mobile and Ubiquitous Systems: Networking and Services*, pages 268–277, August 2004.
22. G. Klyne and J.J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, February 2004.
23. T.J. Lehman, St.W. McLaughry, and P. Wyckoff. T Spaces: The Next Wave. In *32nd Hawaii Int'l Conf. on System Sciences*, January 1999.
24. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services. W3C Member Submission, November 2004.
25. F. Martín-Recuerda. Towards CSpaces: A new perspective for the Semantic Web. . In *1st Int'l IFIP/WG12.5 Working Conf. on Industrial Applications of Semantic Web*, August 2005.
26. D.L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, February 2004.
27. R. Menezes and R. Tolksdorf. Adaptiveness in Linda-based Coordination Models. In *Workshop on Engineering Self-Organising Applications*, pages 212–232, July 2003.
28. I. Merrick and A. Wood. Coordination with scopes. In *ACM Symposium on Applied Computing*, pages 210–217, March 2000.
29. L.J.B. Nixon, E. Paslaru Bontas Simperl, O. Antonenko, and R. Tolksdorf. Towards Semantic Tuplespace Computing: The Semantic Web Spaces System. In *22nd Ann. ACM Symposium on Applied Computing*, March 2007.
30. A. Omicini and F. Zambonelli. Coordination for Internet Application Development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, September 1999.
31. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Working Draft, October 2006.
32. O. Shafiq, I. Toma, R. Krummenacher, Th. Strang, and D. Fensel. Using Triple Space Computing for communication and coordination in Semantic Grid. In *3rd Semantic Grid Workshop (16th Global Grid Forum)*, February 2006.
33. P. Stickler. CBD - Concise Bounded Description. W3C Member Submission, September 2004.
34. R. Tolksdorf. Laura — A service-based coordination language. *Science of Computer Programming*, 31(2/3):359–381, July 1998.