

Towards a Coordination Infrastructure for Pervasive Computing Environment Based on Tuplespace and Semantic Web

Liang Li¹, Bin Li^{1,2}, Junwu Zhu^{3,1}

¹ Corresponding author Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009, China

² State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210000, China

³ School of Information Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China
m05730@gmail.com, lb@yzu.edu.cn, jdkr@163.com

Abstract

Pervasive computing is an emerging computing paradigm, which is expected to be part of our everyday life in the foreseeable future. The coordination among heterogeneous devices, services, and software agents will be vital for ensuring the success of their interactions. Tuplespace (TS) offers a coordination infrastructure for communication. With pervasive computing environment which is characterized as an open and dynamic distributed system become more ubiquitous, autonomous, and complex, this can pose some limitations to TS. In this paper we introduce Semantic Tuplespaces (STS) based on a combination of Semantic Web technologies and tuplespace computing. It acts as a middleware to enable communication and coordination based on the principle of publish and retrieve of semantic information. We describe overview of the conceptual model and discuss the necessary extensions to the original tuplespace. An initial version of the STS prototype system has been implemented based on Jena and Pellet, a DL reasoner. Finally, through an elaborate use case, we show the usability and benefits of our approach.

Keyword

Coordination, Semantic Web, Tuplespace, middleware

1. Introduction

In the pervasive computing vision, devices, services, and software agents will seamlessly integrated into

human being's daily life and provide services and information to users in an anywhere-and-any-time fashion. As coordination is essential in pervasive computing environment that are open in that they don't preidentify a set of known participants, and dynamic in that the participants change regularly, not just due to occasional failures, it needs a computing infrastructure to solve the problems of coordination.

Linda coordination model [1] consists of a small set of coordination primitives which are orthogonal to any particular programming language and a shared dataspace (i.e., TS). Through abstract computing environment TS, publish and retrieve information in the form of tuple (an ordered list of typed field) can achieve communication/synchronization purposes. Linda uncouples the coordinating components both in space and time, supports synchronous and asynchronous operation, has coordination languages which are focus on the issue of coordination only and adopts retrieve mechanism based on template, as these features make it very useful in the open and dynamic distributed system [3].

As pervasive computing environments become more ubiquitous, autonomous and complex, the need to ground them on common data models grows stronger. Entities in such systems must be able to exchange information, queries, and requests with some assurance that they have a common meaning. To facilitate cooperation and prevent misunderstandings, better languages are needed for sharing knowledge about individuals, events, and situations. One possible approach is to employ Semantic Web [2] technologies for modeling and reasoning about information.

To enable semantic interoperability, we need coordination infrastructure that would allow heterogeneous entities to work together in pervasive

computing environments. It is important that coordination infrastructure have built-in capabilities to manage and process information formalized using Semantic Web (SW) representation languages, to infer and mediate meaningful information from semantic metadata associated with data, and to coordinate information exchange among entities processing this information. Semantic TupleSpaces (STS) work aims at providing one such coordination infrastructure by extending the Linda coordination model.

The rest of the paper is organized as follows. Section 2 describes the STS framework, discusses the extensions to the original tupleSpace when combining the Semantic Web technologies with tupleSpace paradigm. Section 3 presents the case study. Some related works are analyzed in Section 4. Finally, Section 5 discusses future work on the topic of this paper, together with some brief concluding remarks.

2. The STS coordination model

STS is mainly concerned with enhancing tupleSpace with Semantic Web technologies like URI, namespace, RDF(S) [21, 22] and OWL DL [24]. URI provides a reference mechanism that allows information to be distinguished on a world-wide scale. Namespaces provide a separation mechanism that allows different applications to use the same vocabulary without blurring their communication. SW ontology languages RDF(S) and OWL DL provide data modeling specification to represent machine-processable semantics of information. The Semantic Web technologies can be a solution to data and information heterogeneity just as the use of tupleSpace is a solution to protocol and process heterogeneity. This combination of Semantic Web technologies and tupleSpace computing can be a solution to allow heterogeneous entities to work together in the pervasive computing environments.

Then we present an overview of the STS framework to introduce the different entities involved.

2.1 The STS framework

The core of STS consists of the following six classes (see Figure 1):

STSOntology. The STS ontology contains information about all the tuples in data views, semantic views, inference views, existing subspaces, etc.

SemanticSpace. A semantic space is defined as a container for semantic tuples which encapsulate the statements. Class `SemanticSpace` contains all the

coordination operations an entity can perform on the semantic space.

DataView. In the data view all statements, despite their ontology-based typing, are seen as plain data, without semantics. To reflect their conceptual

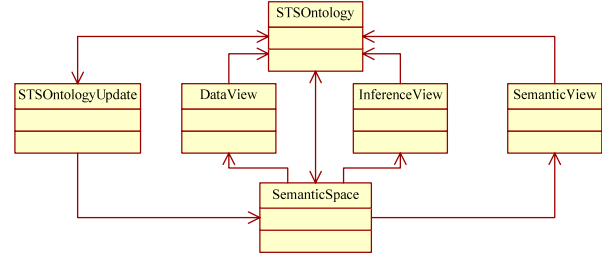


Figure 1. Overview on the core classes of STS

differences, all statements are reified in data view and there is no inference. In other words, a reification of a statement does not entail the statement, nor is entailed by it.

SemanticView. In the semantic view, a statement must be consistent with the specific ontology. It reflects the data view extended by statements inferable from its content according to the available ontologies.

InferenceView. In inference view, all statements are generated by inference when matching based on ontology. As far as the destructive read operation is concerned, the matching statement is removed from the tupleSpace. As a result, the new tuple containing the statement will not be updated into the STS ontology. Hence, the inferred statement is added to the inference view.

STSOntologyUpdate. Class `STSOntologyUpdate` performs update of the STS ontology. The STS ontology is updated not after each coordination operation in the tupleSpace, but after a certain amount of operations is performed. We consider two reasons as follows. On the one hand, frequent updating of the STS ontology generates too much overhead. On the other hand, as far as blocking versions of the primitives are concerned, if no matching tuple is available in model then the process blocks until a matching tuple is available.

In order to enhance tupleSpace as a semantic-aware coordination infrastructure, the traditional Linda model is extended from the following four aspects: STS ontology, new data model, new coordination primitives, and new matching mechanisms.

2.2 STS ontology

The structure of STS is explicitly represented in terms of the ontology (see Figure 2). The ontology

ontology language OWL DL [25] are not compatible with each other.

Therefore, there are however two special cases which we must consider in terms of deciding upon a suitable tuple-based representation in RDF(S) statements and OWL DL statements: blank nodes and collections.

2.3.1 Blank nodes

The concept of blank node is not explicitly used in OWL DL. However, when translating the OWL DL document which has concept of anonymous ontology, anonymous class, anonymous individual or enumerated datatype into the form of triple (or statement) by transformation rules in [25], it should have the aid of the concept of blank node in RDF.

As far as anonymous class is concerned, it can be the complex class defined by set-operator of *owl:unionOf*, *owl:intersectionOf*, *owl:complementOf* and the restriction class defined through value constraints of *owl:allValuesFrom*, *owl:someValuesFrom*, *owl:hasValue* and cardinality constraints of *owl:maxCardinality*, *owl:minCardinality*, *owl:cardinality* in property restrictions. Also, it can be the enumerated class that is defined through *owl:oneOf*. Moreover, a special anonymous datatype – enumerated datatype is also defined by *owl:oneOf* in OWL DL.

Blank node can represent any anonymous resources which can't be globally referenced in RDF(S). However, as aforementioned, blank node in OWL DL has its own specific usage and range. Therefore, we define *sts:RDFsBlankNode* and *sts:OWLBlankNode* which are subclasses of *sts:RDFsType* and *sts:OWLType* respectively in STS ontology to represent special field type – blank node in a tuple.

2.3.2 Collections

The collections can be identified by the global URI references, in which it is used as an explicit type in RDF(S). An RDF collection is represented as a list structure in the RDF graph which is constructed using a predefined collection vocabulary correspondingly.

The collection is usually used as a part of other definitions which can implicitly appear in OWL DL only. Although, it can't be identified by the universal URI references, it uses blank node identifier as its local reference. Moreover, it can only be used in the binding definitions, i.e., when using the properties of *owl:distinctMembers*, *owl:intersectionOf*, *owl:unionOf* and *owl:oneOf*, it should employ the collection as its range values. If we use the properties of *owl:intersectionOf* and *owl:unionOf* to define the

complex classes, the collection is a closed set of classes. If the enumerated class is defined by *owl:oneOf*, its range values is a closed set of individuals. Similarly, special data type – enumerated datatype in OWL DL is defined through *owl:oneOf*, it requires the range values of the property to be a closed collection of literals.

Consequently, distinguishing from *rdf:List*, OWL lists are represented by *sts:OWList* which is a subclass of *sts:OWLType* in STS ontology. However, the OWL lists in the *SemanticView* should be operated through a group of OWL lists property constructs defined in the STS ontology, rather than the RDF collection vocabulary (i.e., *rdf:first* and *rdf:rest*). Support for OWL lists is subject of ongoing work also at the level of the *SemanticSpace* API, which is introduced in the next section.

2.4 Coordination primitives for semantic tuples

The operations an entity can perform on *SemanticSpace* are shown in Figure 3. The API provides operations for writing, reading, and removing SW knowledge. Any entity accessing the *SemanticSpace* should work within a given context that defines a partial view upon the interpreted information. According to this, we consider that there are three views on the *SemanticSpace* in STS. As aforementioned, we make a fundamental distinction between *DataView* and *SemanticView* upon stored tuples in *SemanticSpace*. In the *DataView* all tuples, even though they are Ontology-based tuples, are seen as plain data. No ontological information is considered. The *SemanticView* interprets the tuples according to the semantics of the information they encode. In other words, ontologies underlie the *SemanticViews*' interpretation of the *SemanticSpace* content.

In STS, each tuple is placed to a particular subspace. Multiple subspaces strictly partition the global tuplespace and shield the view on the space from each other. The coordination primitives can specify explicitly which subspace they wish to operate upon. Otherwise, the primitives will operate upon the view of the global tuplespace. As a result, it offers a better resolution of performance and scalability issues through tuplespace partitioning and distribution.

Coordination primitives for *SemanticTuple* are backwards compatible with the primitives out, rd and in of the traditional Linda. Furthermore, we tailor the primitives outr, rdr and inr (originally introduced in the Semantic Web Spaces) for *SemanticTuple*. Bulk

```

public class SemanticSpace {
    public synchronized boolean out(Statement stmt, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized boolean out(Statement[] stmts, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized Statement rd(Triple semanticTripleTemplate);
    public synchronized Statement rd(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement[] copycollect(Triple semanticTripleTemplate);
    public synchronized Statement[] copycollect(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement in(Triple semanticTripleTemplate);
    public synchronized Statement in(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement[] collect(Triple semanticTripleTemplate);
    public synchronized Statement[] collect(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized boolean outr(Statement stmt, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized boolean outr(Statement[] stmts, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized Statement rdr(Triple semanticTripleTemplate);
    public synchronized Statement rdr(Node semanticTupleID);
    public synchronized Statement rdr(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement rdr(Node semanticTupleID, URI semanticSpaceURI);
    public synchronized Statement[] rdrcopycollect(Triple semanticTripleTemplate);
    public synchronized Statement[] rdrcopycollect(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement inr(Triple semanticTripleTemplate);
    public synchronized Statement inr(Node semanticTupleID);
    public synchronized Statement inr(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement inr(Node semanticTupleID, URI semanticSpaceURI);
    public synchronized Statement[] inrcollect(Triple semanticTripleTemplate);
    public synchronized Statement[] inrcollect(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized boolean ia(Statement stmt, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized boolean ia(Statement[] stmts, URI semanticSpaceURI, String semanticTupleSource);
    public synchronized Statement sir(Triple semanticTripleTemplate);
    public synchronized Statement sir(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized Statement dir(Triple semanticTripleTemplate);
    public synchronized Statement dir(Triple semanticTripleTemplate, URI semanticSpaceURI);
    public synchronized URI mir(Triple semanticTripleTemplate);
}
    
```

Figure 3. The class SemanticSpace. Exception declarations and non-blocking methods are omitted for the sake of clarity.

primitives are also considered. Finally, all primitives have a corresponding version of non-blocking manner. Due to the *SemanticTuple* has a *SemanticView*, small yet elegant set of primitives *ia* (*inferable addition*), *sir* (*inferable single read*), *mir* (*inferable multiple read*) and *dir* (*inferable destructive read*) are defined to operate upon the data according to the semantics of the information that it contains. Other primitives operate upon the *DataView*.

Detailed descriptions of the coordination primitives operating on the *SemanticView* of the *SemanticSpace* in a blocking version are given as follows.

ia(Statement stmt, URI semanticSpaceURI, String semanticTupleSource): boolean
ia(Statement[] stmts, URI semanticSpaceURI, String semanticTupleSource): boolean

A *SemanticTuple* existing in a *DataView* denotes its presence there. However, a *SemanticTuple* existing in a *SemanticView* represents the assertion of its truth within that view beyond its presence. Therefore, the *ia* operation is used to assert the tuple containing statement *stmt* within the *SemanticView* beyond its presence in the counterpart *DataView* of the *SemanticSubSpace* identified by the URI *semanticSpaceURI*.

sir(Triple semanticTripleTemplate): Statement

sir(Triple semanticTripleTemplate, URI semanticSpaceURI): Statement

These two template-based operations are applied to retrieve a single matching tuple in a non-destructive manner. The matching tuple must be satisfiable according to the current ontological information. In the case of first *sir* operation the matching tuple is retrieved from the *SemanticView* of the global *SemanticSpace*, whereas in the second one the matching tuple is retrieved from the *SemanticView* of the specified *SemanticSubSpace* identified by the URI *semanticSpaceURI*.

dir(Triple semanticTripleTemplate): Statement
dir(Triple semanticTripleTemplate, URI semanticSpaceURI): Statement

These two operations have similar semantics as the operations introduced above. However, they are operating in a destructive manner. In fact, the *dir* operation does not imply a deletion of the matching tuple from the space completely. Rather, it denies the truth value of the tuple in the *SemanticView*. Therefore, in both operations, the matching statement is removed from the *SemanticView* but it is retained in the counterpart *DataView*, and the tuple containing the statement is marked as “False” in STS ontology.

mir(Triple semanticTripleTemplate): URI

This is a multiple read operation that acts on the *SemanticView*. It returns a URI which is the identifier of a *SemanticLocalSpace*. A *SemanticLocalSpace* is a temporary tuple container created by the system into which all matching tuples are copied. When the above-mentioned operations are applied to a *SemanticLocalSpace*, the *SemanticSpaceURI* parameter specifies the identifier of the *SemanticLocalSpace*. Then insert, retrieve and deletion operations would be applied to the *SemanticLocalSpace* and not to the *SemanticSpace* as a whole. As a result, the changes are made in those temporary containers will not affect the rest of the persistent container such as *SemanticSubSpace*.

2.5 New matching mechanisms

STS supports three different levels of matching on *SemanticTuple*, which correspond to the three sets of coordination primitives. The traditional Linda primitives (rd, in, copycollect and collect) and the primitives (rdr, inr, rdrcopycollect and inrcollect) operate upon *DataView*. They match on *SemanticTuple* disregarding ontological information entirely. *SemanticTuple* is either read by its identifier (*TupleID*) or by a template. For the *SemanticTuple*, each field contains a URI (or, in the case of the *Object*, a literal). Each field is also typed accordingly, URIs by classes and literals by datatypes. However, the former consider the URIs as string. Therefore, the *SemanticTuple* matching is based on the equality of pure URI string. The latter regard the RDF/XML syntax form of an ontology. The special syntactic constructs such as blank nodes and collections are also specifically considered in this level matching. For example, the blank nodes and collections in OWL DL will match on variables of type *sts:OWLBlankNode* and *sts:OWList*, respectively or a *Node_ANY*. A *Node_ANY* is a meta-node that is used to stand for any other node in a query [18].

Alignment of Linda-like template matching with Semantic Web query provides a semantic matching. The semantic matching mechanism was motivated by recent achievement in RDF query language – SPARQL [20], which invokes reasoning engines and operates on original and inferred tuples. In that way the matching algorithms fully exploit the semantics of the information published.

The operations of STS will be illustrated through an example, in terms of *SemanticTuples*, how particular knowledge can be expressed through the STS ontology and domain ontologies and subsequently retrieved by a process using the STS.

3. The case study

An initial version of the STS prototype system has been implemented based on Jena [18] and Pellet [19]. To illustrate the use of STS, we define a domain ontology about U-University. The OWL representation of the U-University ontology is defined in a single ontology document (u-university.owl) under the XML namespace <http://www.u-university.edu/ontologies/STS/u-university#>. Figure 4 describes the part of the u-university ontology. This ontology imports the part of CoBrA ontologies (COBRA-ONT v0.4) tailored for the case. For sake of simplicity, we have kept at the minimum the information presented, thus the example is not completely specified, even though its generalization has no difficulties. Moreover, we shall assume that a single *SemanticSubSpace* (its URI is <http://www.u-university.edu/STS/SemanticSubSpace0> in the example) is used by all processes.

3.1 Scenario

Many people work and study at U-University which is an open environment. Each person carries a Bluetooth-enabled device that servers as a multipurpose mobile computing platform. These people are also highly dynamic. As a result, the different devices dynamically enter and leave the system and provide large amounts of varying information. All devices involved in the system hold a share of the space and the information is persistently stored in a shared tuplespace.

Bob is a teacher of department of computer science. He prepares a Q&A before examination (between 8:00AM and 10:00AM). As most of the classrooms are left for the examination, he can't decide which one can be used in advance. Then Bob publishes the classroom information to the *SemanticSpace* when he finds a vacant classroom at 7:45AM. A student named Chris is in the dormitory at 7:30AM and he wants to know the classroom information in order to be on time for the Q&A, so he queries to the *SemanticSpace* through a PDA. When the matching information is retrieved, an interaction decoupled in time, space and reference between Bob and Chris is carried out. After the Q&A, the information published by Bob makes no sense any more, so he deletes it from the *SemanticSpace*.

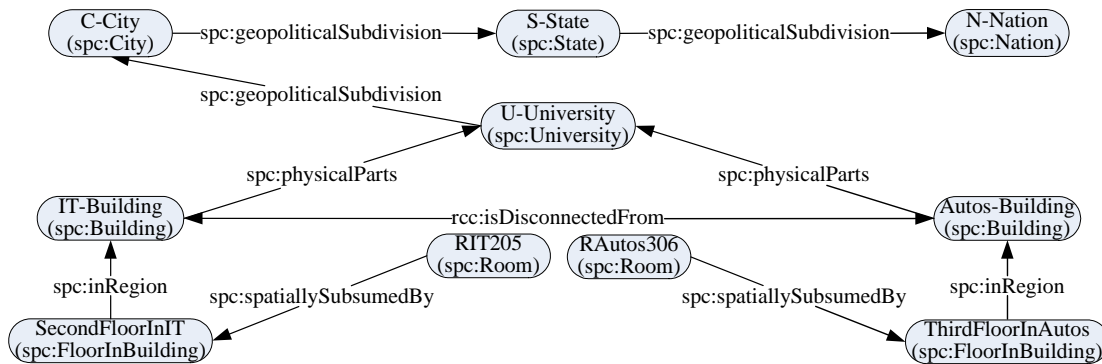


Figure 4. Part of the u-university ontology

3.2 Analysis

The interaction involves the execution of the following two concurrent steps:

Step 1 – ia @ Bob is in room RIT205 at 7:45AM.

In this step, Bob publishes a set of tuples containing location context description of himself to the *SemanticSubSpace* using the ia operation. This step consists of two ordinal sub-steps:

(1) Translating crude information into a set of statements through the dedicated API, the statements stmts are described as follows:

```
<univ:locationContextDescriptionOfBob,
loc:locationContextOf, univ:Bob>
<univ:locationContextDescriptionOfBob, loc:locatedIn,
univ:RIT205>
<univ:locationContextDescriptionOfBob,
tme:hasInstantDatatypeDescription, 2007-06-07T07:45:00^^xsd:dateTime>.
```

(2) Performing the ia(stmts, http://www.u-university.edu/STS/SemanticSubSpace0, "Bob") operation, Table 1 shows the result of the operation. DataView column lists the URIs of the reifiedstatements in the *DataView*, where the reifiedstatements reified the original statements stmts; SemanticView column lists the statements (i.e., the original statements stmts) in the *SemanticView*; and stmtsAddToSTSOntModel column lists all the statements added to the STS ontology, where the statements describe the properties of the new *SemanticTuples*.

Step 2 – mir @ where was Bob during 7:30AM-8AM. This step consists of five ordinal sub-steps:

(1) Performing the operation sir((?LocationContextDescription, loc:locationContextOf, univ:Bob), http://www.u-university.edu/STS/SemanticSubSpace0), as aforementioned, Bob published the information at 7:45AM, but Chris retrieved the information at 7:30AM. As a result, if a matching statement exists,

then the operation continues with (2), otherwise the operation blocks. However, the statement <univ:locationContextDescriptionOfBob, loc:locationContextOf, univ:Bob> is available at 7:45AM.

(2) Performing the operation sir((univ:locationContextDescriptionOfBob, tme:hasInstantDatatypeDescription, ?Instant), http://www.u-university.edu/STS/SemanticSubSpace0), the matching statement is <univ:locationContextDescriptionOfBob, tme:hasInstantDatatypeDescription, 2007-06-07T07:45:00^^xsd:dateTime>.

(3) Inferring whether the time instant '2007-06-07T07:45:00' is inside of the time interval ('2007-06-07T07:30:00', '2007-06-07T08:00:00') by the axiom associated with the property tme:inside. If the result is true, then the operation continues with (4), otherwise return null.

(4) Performing the operation sir((univ:locationContextDescriptionOfBob, loc:locatedIn, ?SpatialThing), http://www.u-university.edu/STS/SemanticSubSpace0), the matching statement is <univ:locationContextDescriptionOfBob, loc:locatedIn, univ:RIT205>. The object univ:RIT205 of the matching statement is an individual of the class spc:Room. However, Chris can't find the specific classroom based on this information. Obviously, Chris doesn't necessarily have the knowledge about the domain ontology, so he can't determine which building and which flat is the classroom in only through a specific identifier (univ:RIT205). To realize it, operation 5 needs to be executed.

(5) Performing the operation mir(univ:RIT205, spc:inRegion, ?SpatialThing), all matching statements are:

```
<univ:RIT205, spc:inRegion, univ:N-Nation>
<univ:RIT205, spc:inRegion, univ:S-State>
<univ:RIT205, spc:inRegion, univ:C-City>
<univ:RIT205, spc:inRegion, univ:U-University>
```

Table 1. Result of the ia operation in step 1

DataView	SemanticView
sts:OWLStatement0	univ:locationContextDescriptionOfBob loc:locationContextOf univ:Bob .
sts:OWLStatement1	univ:locationContextDescriptionOfBob loc:locatedIn univ:RIT205 .
sts:OWLStatement2	univ:locationContextDescriptionOfBob time:hasInstantDatatypeDescription 2007-06-07T07:45:00^^xsd:dateTime .
stmtsAddToSTSOntModel	
http://www.u-university.edu/STS/SemanticSubSpace0	rdftype sts:SemanticSubSpace .
http://www.u-university.edu/STS/SemanticSubSpace0	sts:hasSemanticTuple sts:SemanticTuple2 .
http://www.u-university.edu/STS/SemanticSubSpace0	sts:hasSemanticTuple sts:SemanticTuple1 .
http://www.u-university.edu/STS/SemanticSubSpace0	sts:hasSemanticTuple sts:SemanticTuple0 .
sts:SemanticTuple0	rdftype rdfs:Resource .
sts:SemanticTuple1	rdftype rdfs:Resource .
sts:SemanticTuple2	rdftype rdfs:Resource .
sts:SemanticTuple0	rdftype sts:SemanticTuple .
sts:SemanticTuple1	rdftype sts:SemanticTuple .
sts:SemanticTuple2	rdftype sts:SemanticTuple .
sts:SemanticTuple0	sts:isInferredSemanticTuple False^^xsd:string .
sts:SemanticTuple1	sts:isInferredSemanticTuple False^^xsd:string .
sts:SemanticTuple2	sts:isInferredSemanticTuple False^^xsd:string .
sts:SemanticTuple0	sts:hasStatus True^^xsd:string .
sts:SemanticTuple1	sts:hasStatus True^^xsd:string .
sts:SemanticTuple2	sts:hasStatus True^^xsd:string .
sts:SemanticTuple0	sts:containsStatement sts:OWLStatement0 .
sts:SemanticTuple1	sts:containsStatement sts:OWLStatement1 .
sts:SemanticTuple2	sts:containsStatement sts:OWLStatement2 .
sts:SemanticTuple0	sts:hasTupleSource univ:Bob .
sts:SemanticTuple1	sts:hasTupleSource univ:Bob .
sts:SemanticTuple2	sts:hasTupleSource univ:Bob .

<univ:RIT205, spc:inRegion, univ:IT-Building>

<univ:RIT205, spc:inRegion, univ:SecondFloorInIT>.

When the matching statements are available, student Chris realizes that the RIT205 is the location of Q&A, and then the RIT205 is on the second floor of the IT building in U-University. Therefore, he can easily find the classroom of Q&A.

4. Related works

Gelertner's Linda [1] project on generative communication discussed the greater flexibility available across both time and space by storing and manipulating anonymous *structured data* (generic tuples) via a tuplespace.

In commercial systems of the coordination concept, such as JavaSpaces [4] and TSpaces [5], have shown how the shared data space paradigm can be successfully used for building distributed applications. They allow *Java objects* to be contained in tuple fields. Furthermore, TSpaces provides some built in XML support. *XML documents* are stored as tuples in the tuplespace. Additionally, these documents are processed to produce DOM-objects which are stored as TSpaces tuples, in order to answer XQL queries with one or more matching nodes.

XMLSpaces [6] extends the Linda coordination language for Web-based applications. It supports *XML documents* as tuple fields and multiple matching routines implementing different relations amongst XML documents, including those given by XML query languages. XMLSpaces is distributed and supports

several distribution policies in an extensible manner. XMLSpaces.NET [7] implements the XMLSpaces concept as a middleware for XML documents on the .NET platform. It introduces an extended matching flexibility on nested tuples and richer data types for fields, including objects and XML documents. It is completely XML-based since data, tuples and tuplespaces are seen as trees represented as XML documents. XMLSpaces.NET is extensible in that it supports a hierarchy of matching relations on tuples and an open set of matching amongst data, documents and objects. In addition, the internal representation of the tuplespace is XML-based.

XMARS [8] provides agents a JavaSpace interface to access a set of *XML documents* in terms of Java object tuples. These XML documents are specified in terms of unstructured sets of tuples which can be accessed and modified by exploiting the associative operation which is typical of the Linda model.

None of these extensions have attempted to deal with semantics as in our approach. Little work currently exists in this field and most work is still ongoing. Semantic Tuple Spaces (sTuples) [9] supports semantic data that extends the JavaSpace implementation. A generic semantic tuple containing an object field of type *DAML+OIL* individual extends the JavaSpace entry interface. Fundamentally, a non-semantic implementation of tuplespaces was augmented with Semantic Web technology. The sTuples enables semantic matching on top of object based polymorphic matching. The underlying data model is a mix of non-semantic and semantic

technology (compared to clean semantic tuples in STS) thus inherits all the problems of tuple-based communication.

Triple Space Computing (TSC) [10, 11, 12] is an emerging technology as extended tuplespace computing to support Semantic Web technologies *RDF* and enables asynchronous communication among Semantic Web Services based on persistent publication and read of RDF triples. The approach is built upon an existing coordination system which led to many design decisions being simply carried over rather than reassessed, as we have done, in a Semantic Web technologies context. A minimal architecture for Triple Spaces is also proposed [13], in which however many of the powerful and beneficial aspects of Linda and the tuplespace model have been removed. It does not provide any details on the types of tuples and tuplespaces required in this context. However, the aforementioned approaches have not further considered the implications of coordinating Semantic Web technologies with Linda, as we have done. It is unclear what consideration has been made of how storing RDF/OWL into a tuplespace affects the fundamental issues of tuple and tuplespace representation, Linda coordination operations, different levels of matching and tuplespace partitioning. These issues are specifically handled in our STS. It is also unclear to what extent they continue to respect the basic principles of Linda, while our approach is clearly “backwards compatible”.

Semantic Web Spaces [14, 15] is an extension of XMLSpaces. It remodels the fundamental XMLSpaces ideas to provide a suitable middleware for the *Semantic Web* to support technologies like RDF(S) and OWL, so that it is possible to represent and process semantically-rich information such as RDF(S) and OWL ontologies. However it doesn't clearly explain how to combine the more expressive OWL language with tuplespace. Actually, it is only an RDFSpaces. Moreover, it does not take into account partial incompatibility between RDF(S) and OWL DL which has maximum expressiveness while retaining computational completeness and decidability also affects the issues of OWL tuple representation, matching, coordination operations and definition of tuplespace ontology. These issues are specifically considered in our STS.

5. Conclusions and future work

In this paper we presented the conceptual model and implementation of semantic tuplespaces as a coordination infrastructure for the pervasive computing environments, which combines the idea of

tuplespace computing with Semantic Web technologies. In this paper, we do not touch upon issues concerning distributed tuplespaces implementations.

On the one hand, tuplespace allow entities to publish and retrieve information in an uncoupled manner in terms of space and time. On the other hand, STS makes use of the well-defined URI reference mechanism using URIs and XML namespace separation mechanism to scale over the worldwide scope. It uses ontology language to represent resources and their semantical meanings, and enables information publication and retrieval in a machine-processable manner.

Therefore, the traditional Linda model is extended from the following aspects. Firstly, tuples are extended with ontologies to represent semantic information. Secondly, the tuple-template matching is improved with semantics. Thirdly, the coordination primitives on the spaces are associated with semantic effects. Finally, the STS architecture is represented through an ontology which is inherently extendible, so that support for the new requirements of the application environment. With these features, we believe that the STS can serve not only as a repository of semantic information, but also as a coordination infrastructure allowing heterogeneous devices, services, and software agents to communicate and cooperate over the semantic information.

Even though our initial results are promising, a great deal of research is required before making any conclusive remarks about STS. Presently, the data stored in the Jena model which is being stored in memory, future work include backend persistent storage of data. We also intend to produce a set of formal semantics for the coordination model, having deliberately kept the model very simple to render it tractable to formal analysis. Finally, a thorough and careful experimental evaluation of our prototype will be addressed in future work.

6. Acknowledgements

This paper is supported by the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312002, the National High-Tech Research and Development Plan of China under Grant No.2006AA12A106, the Natural Science Foundation of the Education Department of Jiangsu Province of China under Grant No. 06KJB520132 and the Natural Science Foundation of Jiangsu Province of China under Grant No. BK2007074.

7. References

- [1] D. Gelernter, "Generative Communication in Linda", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ACM Press, New York, vol. 7, no. 1, pp. 80-112, 1985.
- [2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, Holtzbrinck, New York, vol. 284, no. 5, pp. 34-43, May 2001.
- [3] D. Rossi, G. Cabri, and E. Denti, "Tuple-based technologies for coordination", In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, eds., *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 4, pp. 83-109, Springer-Verlag, Berlin Heidelberg, 2001.
- [4] E. Freeman, S. Hupfer, and K. Arnold, "JavaSpaces Principles, Patterns, and Practice", 1st edition, Addison-Wesley Longman Ltd, Essex, UK, 1999.
- [5] P. Wyckoff, S.W. McLaughry, T.J. Lehman, and D.A. Ford, "T Spaces", *IBM Systems Journal*, IBM, Danvers, MA, vol. 37, no. 3, pp. 454-474, 1998.
- [6] R. Tolksdorf and D. Glaubitz, "Coordinating Web-Based Systems with Documents in XMLSpaces", In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, eds., *Proc. of the 9th Int. Conf. on Cooperative Information Systems (CoopIS 2001)*, Springer-Verlag, Berlin Heidelberg, pp. 356-370, 2001.
- [7] R. Tolksdorf, F. Liebsch, and D.M. Nguyen, "Xmlspaces.net: An extensible tuplespace as xml middleware", In V. Skala and P. Nienaltowski, eds., *Proc. of the 2nd Int. Workshop on .NET Technologies (.NET Technologies'2004)*, Vaclav Skala UNION Agency - Science Press, University of West Bohemia, pp. 1-8, 2004.
- [8] G. Cabri, L. Leonardi, and F. Zambonelli, "XML Dataspaces for Mobile Agent Coordination", In J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, eds., *Proc. of the 2000 ACM symposium on Applied computing - Volume 1 (SAC '00)*, ACM Press, New York, pp. 181-188, 2000.
- [9] D. Khushraj, O. Lassila, and T. Finin, "sTuples: Semantic Tuple Spaces", In F.M. Titsworth, ed., *Proc. of the 1st Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004)*, IEEE Computer Society, Los Alamitos, CA, pp. 268-277, 2004.
- [10] D. Fensel, "Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information", In F.A. Aagesen, C. Anutariya, and V. Wuwongse, eds., *Proc. of IFIP Int. Conf. on Intelligence in Communication Systems (INTELLCOMM 2004)*, Springer-Verlag, New York, pp. 43-53, 2004.
- [11] D. Fensel, R. Krummenacher, O. Shafiq, E. Kuhn, J. Riemer, Y. Ding, and B. Draxler, "TSC - Triple Space Computing", *Elektrotechnik & Informationstechnik*, Springer Wien, Berlin Heidelberg, vol. 124, no. 1-2, pp. 31-38, 2007.
- [12] E. Simperl, R. Krummenacher, and L. Nixon, "A Coordination Model for Triplespace Computing", In A.L. Murphy and J. Vitek, eds., *Proc. of the 9th Int. Conf. on Coordination Models and Languages (COORDINATION 2007)*, Springer-Verlag, Berlin Heidelberg, pp. 1-18, 2007.
- [13] C. Bussler, "A minimal triple space computing architecture", In C. Bussler, D. Fensel, U. Keller, and B. Sapkota, eds., *Proc. of the WIW 2005 Workshop on WSMO Implementations*, volume 134. *CEUR Workshop Proceedings*, <http://CEUR-WS.org/Vol-134/>, 2005.
- [14] R. Tolksdorf, E.P. Bontas, and L.J.B. Nixon, "Towards a tuplespace-based middleware for the Semantic Web", In B. Werner, ed., *Proc. of the 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI '05)*, IEEE Computer Society, Washington, DC, pp 338-344, 2005.
- [15] L. Nixon, E.P.B. Simperl, O. Antonechko, and R. Tolksdorf, "Towards Semantic Triplespace Computing: The Semantic Web Spaces System", In A. Ricci, M. Schumacher, and B. Angerer, eds., *Proc. of the 2007 ACM symposium on Applied computing (SAC '07)*, ACM Press, New York, pp. 360-365, 2007.
- [16] P.F. Patel-Schneider and D. Fensel, "Layering the Semantic Web: Problems and Directions", In I. Horrocks and J. Hendler, eds., *Proc. of the 1st Int. Semantic Web Conf. on the Semantic Web (ISWC 2002)*, Springer-Verlag, Berlin Heidelberg, pp. 16-29, 2002.
- [17] J.Z. Pan and I. Horrocks, "RDFS(FA): Connecting RDF(S) and OWL DL", *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society, Los Alamitos, CA, vol. 19, no. 2, pp. 192-206, 2007.
- [18] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: Implementing the semantic web recommendations", In R. Stata, ed., *Proc. of the 13th international World Wide Web conference on Alternate track papers & posters (WWW Alt. '04)*, ACM Press, New York, pp. 74-83, May 2004.
- [19] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner", *Journal of Web Semantics*, Elsevier, Amsterdam, vol. 5, no. 2, pp. 51-53, June 2007.
- [20] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF", *W3C Candidate Recommendation*, <http://www.w3.org/TR/rdf-sparql-query/>, June 2007.
- [21] G. Klyne and J.J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax", *W3C*

Recommendation, <http://www.w3.org/TR/rdf-concepts/>, Feb 2004.

[22] D. Brickley, "RDF Vocabulary Description Language 1.0: RDF Schema", W3C Recommendation, <http://www.w3.org/TR/rdf-schema/>, Feb 2004.

[23] P. Hayes, "RDF Semantics", W3C Recommendation, <http://www.w3.org/TR/rdf-mt/>, Feb 2004.

[24] S. Bechhofer, F.V. Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, "OWL Web Ontology Language Reference", W3C Recommendation, <http://www.w3.org/TR/owl-ref/>, Feb 2004.

[25] P.F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax", W3C Recommendation, <http://www.w3.org/TR/owl-semantics/>, Feb 2004.