ENABLING COLLABORATIVE E-HEALTH THROUGH TRIPLESPACE COMPUTING

Lyndon J.B. Nixon
Institute of Computer Science
Networked Information Systems
Free University of Berlin
Takustr. 9, 14195 Berlin, Germany
nixon@inf.fu-berlin.de

Dario Cerizza, Emanuele Della Valle
CEFRIEL - Politecnico di Milano, Italy
cerizza|dellavalle@cefriel.it

Elena Simperl, Reto Krummenacher
STI - Semantic Technology Institute
Universität Innsbruck, Austria
elena.simperl|reto.krummenacher@sti-innsbruck.at

Abstract

The design and promotion of electronic patient summaries as an instrument to facilitate the pervasive delivery of healthcare is emerging as a key technology in eHealth solutions. From the technical point of view this requires powerful middleware systems supporting interoperability, multi-lingualism, security and patient privacy. In this paper we present a semantic coordination model and describe how it can be used to support pervasive access to electronic patient summaries.[1]

Keywords: Triplespace, triplespace computing, e-health, RDF, semantic

1 INTRODUCTION

eHealth relates to a wide range of healthcare related activities being supported by computer systems and communication networks. Despite increasing uptake in major healthcare institutions, healthcare delivery remains highly fragmented and it is difficult to integrate the various types of information and IT platforms. The most representative example in this respect is probably the pervasive access to patient information. Data about an individual is created, processed and stored in different systems spread across several healthcare institutions, often without interrelation. Hence it cannot be accessed, integrated and used instantly by healthcare professionals or administrative personnel, independent of where they are and how they wish to access it, thus leading to additional costs for locating or obtaining information or replicating particular procedures and to a deterioration of the perceived quality of service.

In order to increase the efficiency of patient care delivery, healthcare parties must be able to access and exchange patient information independent of organizational and technological heterogeneities. The European Commission is performing a first step in this direction by defining guidelines for the *European Patient Summary* (EPS) [7]. We aim at providing this summary in the Semantic Web language RDF, which is based on a formal semantics [12], and hence supports data validation, integration and the inference of new knowledge through well established logical reasoning approaches. It also abstracts from presentation formats, allowing for heterogeneous client access, including in low bandwidth and restricted client resource situations.

The realization of the EPS demands a powerful coordinating middleware for exchanging primary clinical information across European healthcare networks that guarantees ubiquitous access to distributed and multi-faceted data objects with a focus on scalability, persistency and interoperability. *Triplespace computing* [8] is an emerging proposal for providing Web-scale data coordination for information formalized using Semantic Web representation languages such as the aforementioned RDF. To enable this, a new tuplespace model

[1]This work has been supported by the TripCom (IST-4-027324) Specific Targeted Research Project.

for handling interpreted information with assigned truth values and an extended Linda-based coordination language for knowledge coordination are specified.

In this paper we present our specification for triplespace computing and outline its use in an European Patient Summary scenario. The rest of this paper is organized as follows: Section 2 describes our proposal of a coordination model for triplespace computing. We illustrate its application in the electronic patient summary scenario in Section 3. Related and future work are considered in the Sections 4 and 5, respectively.

2 TRIPLESPACE COMPUTING

Most available IT applications and systems depend largely on synchronous communication links which tightly couple the communicating agents, requiring that agents know how to reach their communicating partner (direct addressing), that they are active at the same time (temporal dependency) and that if one communication partner loses their connection, the communication is broken and possibly lost (point-to-point). A further implication of this popular communication paradigm is in many cases the need for a priori knowledge of the required messages and protocols. This makes ad hoc coordination without specified contracts a task that is difficult to realize.

These obvious issues motivated the choice of a new communication paradigm at the intersection of tuplespace computing and the Semantic Web. Tuplespaces [11] are shared data stores which allow distributed processing of information - stored in ordered vectors known as ”tuples” - by various devices without requiring synchronous connections amongst the interacting applications. Tuplespace coordination is enabled by a simple yet powerful co-ordination language called Linda [5] that permits agents to emit a tuple into the space (operation *out*) or associatively retrieve tuples either removing them (*in*) or not (*rd*). Retrieval is governed by matching rules. Tuples are matched against a template, which is a tuple that contains both literals and typed variables. A match occurs if the template and the tuple have the same length and field types and if the value of literal fields are identical. The tuple (”N70241”,EUR,22.14) will match the template (”N70241”,?currency,?amount) and bind the variables ?currency and ?amount to the values EUR and 22.14 respectively.

The Semantic Web [2] extends the Web with machine processable semantics, allowing data exchange in heterogeneous application fields. The first layer in the stack of knowledge representation languages which is seen to build the Semantic Web is the Resource Description Framework (RDF) [12]. RDF is based on a directed graph data model where knowledge statements take the form of triples with the structure (subject, predicate, object). For example, to say that Joe Blogg is the author of a particular book, one could write:

```
db:Joe_Blogg dc:author isbn:085756353
```

Each value uniquely refers to a concept using the URI scheme made popular by the Web, while the URI given does not necessarily in that case have to refer to an actual Web resource. Rather, concepts can be named within particular namespaces (to avoid unambiguity), for example the predicate dc:author here is taken from the Dublin Core namespace and has the unambiguous meaning of an author of a resource. The resource in question is uniquely identified by its ISBN number and the subject, Joe Blogg, would have use an URI to identify himself. RDF Schema [3] can be used to add additional schematic facts about these concepts, e.g. that Joe Blogg is a person (’instance of’ the class Person) or that the Dublin Core author property must have a person as its subject. RDF thus combines simplicity with a formal semantic basis that enables logical inference over sets of statements.

Combining the two introduces a new communication platform that provides persistent and asynchronous dissemination of machine understandable information. We call this combined communication platform *Triple Space*: semantic information encoded in RDF triples provides a natural link from the Semantic Web and tuplespaces to triplespace computing.

To realize triplespace computing it is necessary to revisit the definition of tuples and tuplespaces, to adapt them for our purpose of the co-ordination of statements of knowledge. Here after we concentrate on the description of the required adaptations and extensions to the tuplespace coordination language, e.g. the novel coordination primitives that are necessary to make the Linda language compatible to the requirements of the Semantic Web.

2.1 Triples and triplespaces

Following the Linda paradigm a Triple Space 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. Sets of RDF statements are interpreted as directed graphs and are used as the main data structure for communication, which implies a more expressive data model than in classical Linda. In most cases the meaning of a single

triple is very limited, and any piece of RDF in the Semantic Web consists of a non-empty set of triples – an RDF graph.

A triplespace is defined as a container for triples which encapsulate the RDF statements - the whole of all triplespaces provides the Triple Space infrastructure. A triplespace can moreover be divided into virtual subspaces and physically partitioned across distributed kernels. Every space is addressed using a unique identifier, e.g a URI, and may contain multiple (sub-)spaces, while it can only be contained in at most one parent space. Consequently a tuple can be contained in a space - and implicitly in all the direct and indirect parent spaces of that space. Communication can be restricted to a boundable part of the Triple Space, i.e. a subspace, to allow for greater efficiency in interactions (local scalability and completeness). The distribution of data across triplespaces and the decentralization of the overall system (distribution of responsibilities) provides for a self-organising solution to growing knowledge in the Triple Space. Users having common interests shared their data in a particular space instead of using the global Triple Space system.

A triple's contribution to a triplespace is described in TripCom by use of a triplespace ontology. The ontology modules so far developed are concentrating on the description of triples 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 1.[2] An example annotation of a triple stored in the space is given in Table 2.

In Triple Space the metadata infrastructure is not only used to link triples to spaces, but also to annotate the data with access logs, links to the physical implementation and security and trust credentials. The use of an ontology to describe the space and the data allows using the available infrastructure to manage and reason about the space and hence provides the necessary tools for reflection and self-adaption of the space middleware. This in turn opens up new means for enhancing the retrieval procedures, distribution algorithms (e.g. through replication and caching), and the integration of dynamically coupled external components. Hence, ontology-driven space management is seen to be one of the major assets of semantic tuplespaces compared to traditional approaches.

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.

2.2 Coordination model

The original 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. The development of the Triple Space (TS) API was guided by a review of previous extensions of Linda combined with an understanding of the requirements for knowledge co-ordination. An outline of the TS API is given in Table 3.

The coordination API extends Linda to support the reading and writing of sets of RDF triples as well as individual tuples. Hence the core retrieval operations, *in* and *rd*, while maintaining a version supporting "traditional" Linda approach of returning the first match found, are also extended in *rdmultiple* and *inmultiple* for retrieving a RDF graph constructed from a set of all found matches to the request.

The matching procedure for the retrieval operations is - analogous to Linda - based on templates. The precise syntax and semantics of a template depends on the maturity and complexity of the space implementation and on the query languages and engines employed in this implementation. In the current approach we use

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

Table 1: Excerpt of the Triplespace Ontology

```
Triple                        AccessLogEntry
 formalism URI                 publisher Agent
 isContainedIn Space           date xsd:dateTime
 hasLogEntry AccessLogEntry    type AccessType
 isManagedAtKernel Kernel
                               Kernel
Space                          sharesSpace Space
 isSubspaceOf Space            hasQueryEngine QueryEngine
 isSharedAtKernel Kernel       seeAlsoKernel Kernel

                               QueryEngine
                                language QueryLanguage
                                usesRepository Repository
```

Table 2: A triple represented by the triplespace ontology

```
_:t a ts:Triple;
    :isManagedAtKernel <http://ts.example.de/>;
    :isContainedIn <http://ts.example.de/space>.
```

both simple triple patterns that are very close to traditional Linda templates (hence a triple pattern with one or more variables, e.g. *<x:TripleSpacePaper ?p ?o>*) as well as graph patterns (cf. Table 4) through support for the principle RDF query language SPARQL [19]. Eventually, we expect the templates to evolve to supporting more expressive or more efficient semantic query resolution algorithms as empowered by rule languages and engines. In the API definition we however generally refer to it as template in order to proceed with a stable interaction model.

A noteworthy extension to the original Linda model is the incorporation of a publish-subscribe mechanism. An agent is able to subscribe to a particular type of information by providing a template and will be informed whenever some other agent has introduced new data which leads to a match against that template in the chosen space. This extends the expressiveness of the coordination model by providing a new type of interaction pattern which cannot be supported by the original Linda primitives.

The two operations depicted in Table 5 are management methods used to create and delete spaces. A space is created by giving it a new unique identifier and 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.

2.3 Implementation and optimisation

A first prototype of Triple Space has been implemented. This is a proof of concept development from the first stage of the TripCom project, as not all specifications were finalised at that time. Following the stabilisation of the relevant specifications a first full implementation is now in progress, and the project has a public Web site at Sourceforge [5].

Figure 2.3 shows a logical view on a single Triple Space instance, termed a kernel, with both, kernel-internal components and kernel-external clients and services that may be connected to it. From a birds-eye view on the architecture one can see two different areas. The upper area shows kernel-external entities that may connect to the kernel. The area below shows the components of a kernel itself, i.e. APIs and components implementing these APIs. The components that form a Triple Space kernel communicate over a kernel-internal bus system which is implemented using a space-based middleware system. The integrating middleware allows all components that are drawn directly above or below to communicate with each other. Here JavaSpaces is used to enable space-based co-ordination of the components.

3 A EUROPEAN PATIENT SUMMARY SYSTEM

Several health applications have been widely deployed within major healthcare institutions all over Europe. Regardless, healthcare delivery remains highly fragmented and it is an open challenge to integrate the various types

[5] https://sourceforge.net/projects/tripcom

out(Graph g, URI space):void Inserts the triples included in the graph into the given space. A graph could consist of a single triple or a set of triples.
rd(Template t, URI space, int timeout):Graph Returns a single match to the template provided. A template is a query upon the triplespace following some supported syntax, e.g. a single triple pattern or a SPARQL (graph pattern) query. Depending on the query, a match may be a single triple, a set of bound triples (e.g. following the Concise Bounded Descriptions approach) or a set of individually matched triples (in a complex query). The scope of the query is restricted to the given space. The timeout is used as means to restrain the blocking characteristics of Linda rd to a finite time period.
rd(Template t, int timeout):Graph Operates as the above rd but not restricted to a named space. Rather, the scope of the query is the entire Triple Space and the answer may be taken from (a combination of) any triplespaces. The client has no influence over where knowledge is found to answer their query.
rdmultiple(Template t, URI space, int timeout):Graph Operates as the original rd but not restricted to returning a single match. Rather, attempts to return as many matches as can be found within the scope of the query and the given timeout.
in(Template t, URI space, int timeout):Graph This is the destructive version of rd, not only returning a graph which contains the query match but also deleting this graph from the triplespace. There is also a version of in without the space URI.
inmultiple(Template t, URI space, int timeout):Graph This is the destructive version of rdmultiple, not only returning a graph which contains the query matches but also deleting this graph from the triplespace.
subscribe(Template t, URI space, Callback c):URI Establishes a notification mechanism for the insertion of triples which result in the generation of a match to the given template within the named triplespace. The callback object enables the system to notify the client of the occurrence of a match, and the callback itself will contain a graph representing that match. The operation returns a handle to the successfully registered subscription in form of a URI.
unsubscribe(URI subscription):boolean This operation cancels a given subscription and returns true in case of successful unsubscription.

Table 3: The Triple Space API

TEMPLATE	DESCRIPTION
`?s a doap:Project; foaf:member ?o.`	Matches all triples where the subject is of type doap:Project [3] 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 [4].
`?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 4: Examples of Semantic Templates

create(URI path, String name):boolean
This primitive creates a new space as a subspace of the space found at the given path. The name of the newly created space provides an identification of the new space (as the concatenation of path and space name). It returns true after successful creation.
destroy(URI space):boolean
This operation destroys the space identified by the given URI. Its subspaces and all contained triples are also destroyed. Particular attention has therefore to be paid to rights management to avoid undesired deletion of knowledge.

Table 5: The TS Management API

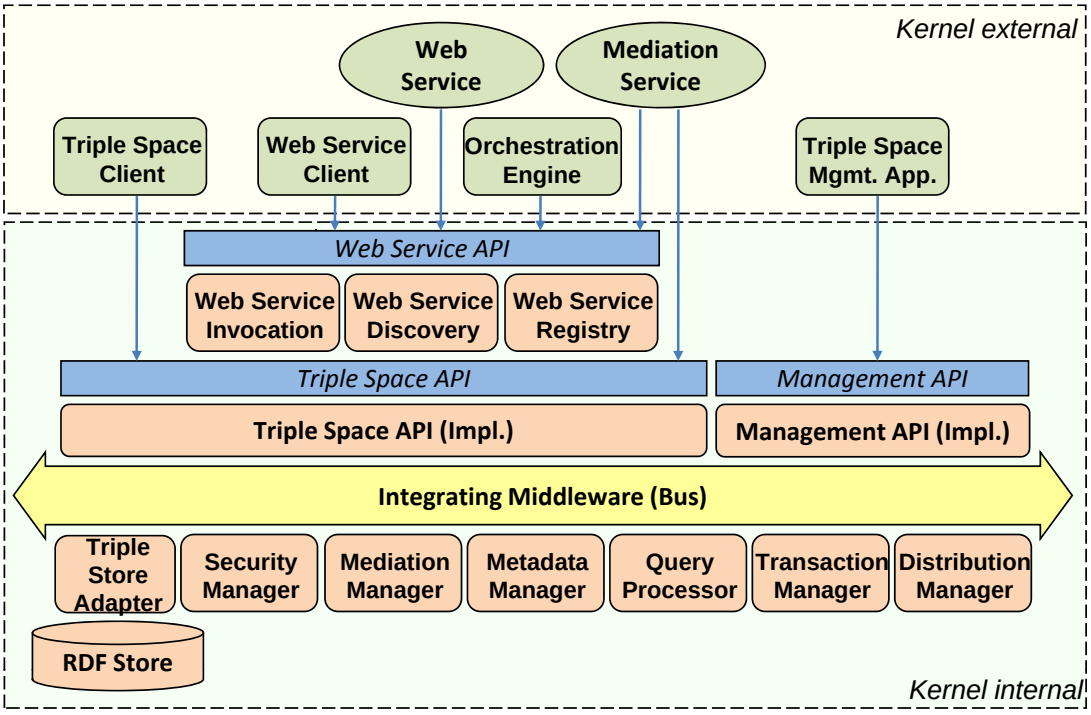

Figure 1: Triple Space Architecture – Logical View

of information and IT platforms. The clinical information about an individual is created, processed and stored in different systems spread across several healthcare institutions. In order to increase the efficiency of patient care delivery, healthcare parties must be able to access and exchange patient information independently of their organizational and technological particularities. But this is a very challenging objective since it needs to deal with not only technical requirements but also with political, organizational and social aspects.

Patient summaries represent concise clinical documents that manage the most crucial information related to the health status of citizens. For this reason they really represent a first step towards a network of complementary and heterogeneous healthcare systems.

The provision of a patient summary at European level is a strategic challenge of the European Union because they represent also an enabling factor for the pervasive delivery of high-quality health-care services across Europe. The European Patient Summary (EPS) is a strategic initiative by the European Commission towards the realization of a European eHealth infrastructure capable of integrating information and applications in order to ensure the pervasive delivery of high quality care services. The EPS initiative should provide the core building blocks required to access and process primary clinical data in arbitrary eHealth applications across the European healthcare delivery network: hospitals, general practitioners, specialists, laboratories, home care services, ambulance services, administration and researchers.

However such a goal demands very strong requirements for a technological platform that can support the scale of the scenario. In particular, there are about 500 million citizens within Europe and 1 million care-givers. Given that a summary will contains around 1 thousand triples, the EPS as a whole will need to be able to manage 500 billion of triples over a network of 10 thousands triplespace kernels. In addition, mobility of citizens must be guarantee and the access to their health information must be ensured anytime and everywhere.

The technical realization of the patient summary needs to build on a specialized middleware which is able to deal with the design principles specified in the European Interoperability Framework (EIF, [6]): multilaterality, subsidiarity, multi-lingualism, and privacy. We can name several requirements for an efficient EPS infrastructure:

- **decentralization/distribution** is a pre-requisite for the realization of **multi-laterality** and **subsidiarity**. A highly distributed EPS infrastructure allows arbitrary healthcare parties to publish and retrieve patient information efficiently and ensures a feasible level of fault-tolerance.
- support for **asynchronous** and **anonymous inter-**

action among institutions is equally important. The coordination of information should happen independently of communication partners.

- to cope with the inherent heterogeneity problems (e.g. different encoding schemes and languages) and to align different eHealth systems the middleware should provide means for flexible **data and application integration**.

- support for appropriate **security** mechanisms is important with respect to **privacy** and **multi-laterality**. The privacy of citizens must be respected, according to EU and specific country policies that guarantee that only authorized care givers will have access to sensitive data. Concerning multi-laterality, healthcare authorities that are responsible for managing the data of the citizen needs to maintain their control over the data published.

An infrastructure that can meet all of these requirements would enable the provision of added-value services beyond today's state of the art. Take for example critical emergency situations where several actors operate in the same moment on several victims and there is a critical need for rapid access to health data and coordination of emergency healthcare actions in order to provide the most effective (and potentially life saving) service.

Such very demanding requirements could be met by the scalable triplespace infrastructure we are developing. The following use case reports on an example of a critical emergency situation where all the capabilities provided by the triplespace are used for effectively coordinating actors that pervasively access the EPS space through mobile devices.

3.1 Use Case Scenario

Mr. Christian is an English citizen that is spending his holidays in Northern Italy and Austria. While traveling by bus along the highly frequented highway Modena-Brennero, he is suddenly involved in a major accident. The bus overturns near Bolzano, in South Tyrol, and many of the travelers are injured, some even severely. Mr. Christian suffers an open fracture of the leg and shows symptoms of shock.

Due to the seriousness of the accident and the high number of victims, the volunteer first aid corps and most ambulances of the region are sent for, which calls upon complex coordination work (also the nearby Italian-speaking region of Trentino is involved). All medical staff has access to the EPS through mobile devices. This allows them to instantly gain access to relevant information about their patients in order to provide the best possible treatment on the spot. Moreover, the EPS built over Triple Space permits the different units to collaboratively treat the victims and to synchronize their activities.

We use for the example snippets a shorthand for the tuples and templates, both of which are marked within angled brackets. RDF concepts are normally full URIs, but for readability here we use QNames, where before the colon a string represents the "namespace" of the concept [6] and after the colon a string represents the name of the concept in that namespace. Templates (queries) here are restricted to triple patterns, where variables are marked with a question mark as first character, and matches are also then single triples. Of course, in a real world scenario we can expect both the triple graphs and templates used to be more complex. The arrow -> separates the request from the client from the response received.

Rescuer Roman, the South-Tyrolese rescue worker that first finds Mr. Christian, searches for Mr. Christian's clinical data in the EPS system using his PDA device. This can be done after having identified Christian in the system through some unique ID, for example the passport number which Christian is thankfully carrying on him. We expect this aspect will be simplified in the future through the development of secure digital identification cards which can be read by enabled EPS mobile devices (through e.g. RFID tags). The system returns for Mr. Christian his Universal Healthcare Identifier (UHID) [7].

```
rd(
  <?foaf:Person, id:passNr, "1234...">, null
  )
  -> <p:js39, id:passNr, "1234...">

rd(
  <p:js39, dh:records, ?dh:record>, null
  )
  -> <p:js39, dh:records, uhid:026253645>
```

The PDA of the rescuer has been easily integrated with the EPS since the triplespace leverages the RDF language to provide semantic interoperability techniques that help in solving heterogeneity issues among existing eHealth applications and eHealth standards. Given the transmission of sensitive data in such a context, the system encrypts and decrypts data in its transmission between the user and the triplespace making use

[6] For the purpose of these examples we use some dummy namespaces which use the QName prefixes id, dh, inn and loc. Other namespaces are explained in the text.

[7] http://www.ncvhs.hhs.gov/app7-2.htm

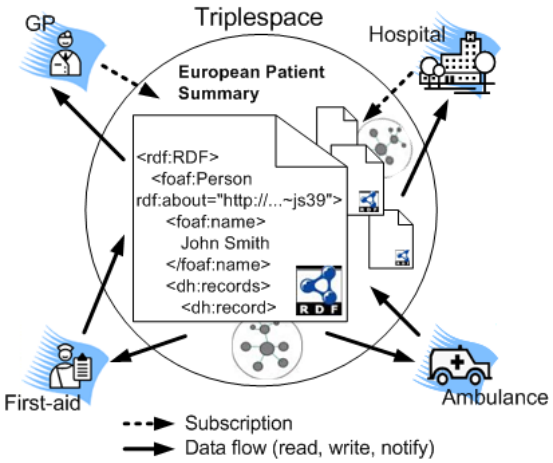

Figure 2: Realization of the EPS with Triplespace

of a key that is only allocated to Roman's current interactions [4].

Now Roman registers the injury to Mr. Christian, taking from the International Classification of Diseases (ICD) [8] the one that describes a "open wound in lower leg" (i.e., `icd:s81`), and placing this in a space belonging to the local healthcare authority (the space to which he has access).

```
out(
   <uhid:026253645, dh:hasInjury, icd:s81>,
   http://gesundheit.sudtirol.at/
   )
```

According to European, English and Italian policies, he is allowed to read all necessary information about Mr. Christian's allergies, immunizations, currently prescribed medication and contagious diseases. The data protection laws restrict first aid workers and ambulance doctors from consulting further details of the patient summary. The information requested by Roman is presented to him in German as the application running on his mobile device can query terms inside the EPS which follow standardized medical terminologies using the appropriate language setting, as they are defined and official translations for their terms in all European languages exist. To make queries on Mr. Christian's EPS more efficient, Roman can find out which triplespace contains the EPS by querying the triplespace metadata for which triplespace *contains* a tuple which identifies the concept uhid:026253645 as being of type eps:EPS (i.e. is an instance of an European patient summary). The subsequent rd operations are restricted to this space.

```
rdmultiple(
   <uhid:026253645, dh:diagnosedWith, ?dh:Contagion>,
   http://health.merseyside.co.uk/wirral, null
   )
   -> <<uhid:026253645, dh:diagnosedWith, icd:b24>>
rd(
   <icd:b24, rdfs:label@DE, ?string>, null
   )
   -> <icd:b24, rdfs:label@DE,
                "Humanes Immundefizienz-Virus"@DE>
rdmultiple(
  <uhid:026253645, dh:allergicTo, ?inn:Analgesic>,
  http://health.merseyside.co.uk/wirral, null
  )
  -> <<uhid:026253645, dh:allergicTo, atc:N02AA01>>
rd(
  <atc:N02AA01, rdfs:label@DE, ?string>, null
  )
  -> <atc:N02AA01, rdfs:label@DE, "Morphin"@DE>
```

Roman learns that Mr. Christian was diagnosed with HIV (which in German is called "Humanes Immundefizienz-Virus") and can take all the necessary precautions. Under normal conditions, Roman would provide Mr. Christian with a dose of the analgesic morphine (i.e., `atc:N02AA01`in the Anatomical Therapeutic Chemical Classification System [9]). According to Christian's summary, he repeatedly showed allergic reactions to morphine and the first aid assistant prefers to administer oxycodone (i.e., `atc:N02AA05`) that does not trigger the same consequences. Furthermore, Roman takes care of stopping the bleeding of Mr. Christian's wounded leg using a bandage (i.e., `dh:CottonGauze`). He logs all the treatments administered to Mr. Christian into the EPS and he moves to take care of other casualties.

```
out(
   <uhid:026253645, dh:rcvdTreatment, atc:N02AA05>,
   tstp://gesundheit.sudtirol.at/
```

[8] http://www.who.int/classifications/icd/en/index.html
[9] http://www.whocc.no/atcddd/

```
   )
out(
   <uhid:026253645, dh:rcvdTreatment, dh:CottonGauze>,
   tstp://gesundheit.sudtirol.at/
   )
```

Other rescuers and ambulance doctors can now read the information published by the rescuer, becoming aware that Mr. Christian requires further medical care. Only shortly thereafter Dr. Anna, an American ambulance doctor working in Trentino, and her team take over the care of Mr. Christian. From his latest EPS entry (now presented in Italian thanking to the terminology mediation), they notice the information published by Rescuer Roman: the description of the injury, the medication and the treatment.

```
rdmultiple(
  <uhid:026253645, dh:hasInjury, ?dh:Injury>,
  tstp://gesundheit.sudtirol.at/, null
  )
  -> <<uhid:026253645, dh:hasInjury, icd:s81>>
rd(
  <icd:s81, rdfs:label@IT, ?string>, null
  )
  -> <icd:s81, rdfs:label@IT,
                "Ferita aperta alla gamba"@IT>
rdmultiple(
  <uhid:026253645, dh:rcvdTreatment, ?dh:Medicine>,
  tstp://gesundheit.sudtirol.at/, null
  )
  -> <<uhid:026253645, dh:rcvdTreatment, atc:N02AA05>,
      <uhid:026253645, dh:rcvdTreatment, dh:CottonGauze>>
rd(
  <atc:N02AA05, rdfs:label@IT, ?string>, null
  )
  -> <atc:N02AA05, rdfs:label@IT, "Ossicodone"@IT>
rd(
  <dh:CottonGauze, rdfs:label@IT, ?string>, null
  )
  -> <dh:CottonGauze, rdfs:label@IT,
                       "Bendaggio di cotone"@IT>
```

Dr. Anna decides that Mr. Christian needs to be hospitalized and she publishes a new emergency case in the EPS with the location of the accident. Local hospitals monitor the space using the publish/subscribe mechanism and are alerted of an emergency case in their vicinity. The hospital consumes the emergency call of Dr. Anna when they have the necessary capacities and emits a tuple acknowledging they can handle the emergency case. Not only can no other hospital erroneously allocate resources to Mr. Christian, but also the ambulance crew can know which hospital they should bring their patient to. The hospital can additionally query on properties of the location strl:str539 to determine if it is nearby, or the ambulance can additionally query on properties of the hospital strl:hospital856 to know how to reach it best. This can be done by automated systems thanks to the use of RDF to unambiguously define properties and values. Furthermore, the given location is actually a member of a RDF class for streets, and yet is still found by a query on an object of type loc:Location given the semantic knowledge (expressible in RDF Schema, a language for defining the semantics of RDF vocabularies) that this class is a subClass of loc:Location. This illustrates that logical inference based on RDF can ensure that concepts can be described specifically without breaking queries, as queries can use more general classes which subsume all relevant concepts. For example, an emergency case should be notifiable to a hospital regardless of the type of location where it occurs yet access to more specific information based on the type of location may be vital knowledge for the healthcare professionals going to attend that emergency case.

```
hospital:
subscribe(
   <?dh:record, dh:emergencyCase, ?loc:Location>,
   tstp://gesundheit.sudtirol.at/, Callback.instance
   )

ambulance:
out(
   <uhid:026253645, dh:emergencyCase, strl:str539>,
   tstp://gesundheit.sudtirol.at/
   )

hospital:
   Callback.instance
   -> <uhid:026253645, dh:emergencyCase, strl:str539>

hospital:
in(
   <uhid:026253645, dh:emergencyCase, strl:str539>,
   tstp://gesundheit.sudtirol.at/, null
   )
out(
   <uhid:026253645, dh:emergencyTaken, strl:hospital856>,
   tstp://gesundheit.sudtirol.at
   )
```

While Mr. Christian is on the way to the emergency room in Trento, Dr. Erica, the emergency doctor, can already take action to initiate the treatment at the local hospital. Dr. Erica and her team access the EPS of Mr. Christian and study the clinical information about his current health status (added by Dr. Anna and Rescuer Roman) and his past medical history. In that way they are ready to receive Mr. Christian and can treat him faster and more accurately.

Back home in England, the general practitioner Dr. Gabriela, that is responsible for Mr. Christian, will be informed automatically by the EPS through her eHR about the changed medical status of the patient (this is possible through a subscription made by the GP on her cared citizens applicable to the entire Triple Space). This notification contains, in the English language of the general practitioner and in the correct format supported by her eHR, all the necessary information about the emergency recovery, the specific medical problems and the treatment received by Mr. Christian.

The entire scenario is illustrated by a UML sequence diagram.

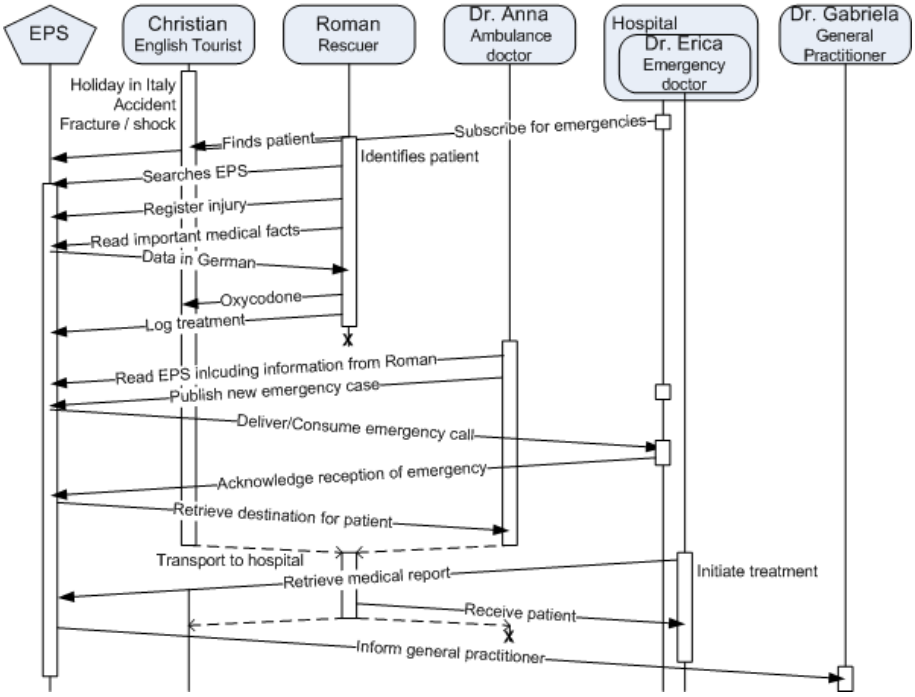

Figure 3: Sequence diagram of scenario

3.2 Scalability, Pervasive Deployment and Usability

This use case scenario is only one sample of the hundreds of possible concurrent requests to the EPS that the scalable and pervasive triplespace infrastructure is required to support.

Therefore the deployment of the triplespace infrastructure must be carefully planned. The underlying triplespace nodes must be pervasively deployed across Europe and an effort must is required to make the EPS accessible through all kinds of devices including hospitals', ambulances' and rescuers' ones.

Moreover, EPS imposes triplespace to assure usability of such an approach to retrieve data, especially in difficult, dangerous and emergency context where timely operations constitute a very strict constrain. To such purpose the following action could be undertaken:

- Usage of subspaces to minimize the numbers of kernels of the distributed infrastructure involved in each interaction and to isolate data which can be locked in order to guarantee completeness of query answers.

- Usage of triplespace support to self-organizing data in order to ensure proximity of knowledge to the client and hence reduce response times.

4 RELATED WORK

A number of approaches are developing in the field of triplespace computing. A review of current activities has been carried out in the interests of identifying commonalities and differences [18]. Besides the core commonality of coordinating the exchange of semantic data, generally RDF, in a tuplespace, these approaches differ in their aims and hence their conceptualisations and realizations.

The Triple Space Computing proposal [8] has spawned a number of initiatives. In the TSC project a first attempt is made to extend the initial proposal with a concrete proposed architecture in which the coordination model is enriched with publish-subscribe capabilities and transactions [9]. cSpaces was born as an independent initiative to extend Triple Space Computing with more sophisticated features and to study their applicability in different scenarios apart from Web Services [15]. The work in this paper is also inspired by this proposal, and differs from earlier work in that it concentrates on a new conceptualization of semantic triplespaces – while previous efforts extended existing systems – and is focused on supporting a number of vital communication scenarios such as the eHealth case given here.

Parallel to this, Semantic Web Spaces [17] have been presented which aims to act as a communication middle-

ware for a Semantic Web of heterogeneous, distributed intelligent agents. Its focus is on the communication of Semantic Web clients rather than Web Services. However both initiatives learn from one another in dealing with the challenges of implementing semantic data exchange on top of the Linda coordination model.

Other tuplespace implementations have focused on providing lightweight, ad-hoc collaboration support for autonomous agents in a pervasive environment. LIME [16] provides coordination over transient sharing of identically-named tuple spaces carried by individual mobile units. Limone [10] decouples the communication by limiting each agent access to its own and its acquaintances' tuplespaces and modelling the Linda coordination language to minimize dependencies on the network (e.g. no remote blocking, timeout on distributed operations). sTuples is the first instance of a semantic tuplespace with a focus on ubiquitous computing but it was limited to exchanging OWL (ontology) documents in tuple fields [13], rather than semantic information in a suitable tuple format.

The benefits of a distributed, peer-to-peer approach for ad-hoc collaboration in hospitals has been described in [1]. An application of tuplespace technology to the healthcare sector has been acknowledged in [14]. The authors describe a tuplespace realized with the use of RFID technology and elaborate on its usage in eHealth scenarios. However, the work cited here does not consider the use of semantic data nor an extension of the coordination model for the coordination of such data.

Hence our work presents a first development in the direction of the integration of semantic and tuplespace computing with application in pervasive scenarios such as the presented use case of ubiquitous access to patient information.

5 CONCLUSIONS AND FUTURE WORK

In this paper we presented the usage of the tuplespace paradigm as a pervasive semantic middleware for realizing the emerging European patient summary. Tuplespaces are a good alternative to common information management and interaction models on the Web, since they allow agents to publish and retrieve information in an uncoupled manner in terms of space and time.

Current middleware technologies do not cover some significant aspects related to heterogeneity, dynamics, openness and scalability. We propose an extension of the electronic patient summary scenario into triplespace computing. The Linda model for coordination is suited to this scenario, as it provides the basic requirements of the system: a common data store, support for multiple agents and their interaction, coordination of that interaction and decoupling from time and space.

The functionality of the system is also abstracted into external agents who interact with the data store. This not only is a basis for modularizing the EPS system and hence supporting reusability and updatability, but also makes system knowledge directly available to any interested (and access enabled) agents in a pervasive environment. Simple agent operations (reading some knowledge from the system) are then standardized (through Linda) and supported from the tuplespace platform without requiring any specific functionality to be executed from the EPS system.

Patient data, as well as associated coding systems and exchange message types, are represented using machine-understandable representation languages such as RDF which have formal semantics to allow for logical inference. This permits automatic mediation between heterogeneous data formats and reasoning over the knowledge of the system to deduce new information which is of use to the healthcare providers. The internal use of RDF also abstracts from actual delivery and presentation format, allowing for the support of also low bandwidth and restricted resource clients.

Our scenario will, in the next step, be integrated with the first Triple Space prototype in order to demonstrate and validate the coordination of European Patient Summary data through triplespace computing. Furthermore, we will continue to specify how patient summaries are modelled and manipulated in RDF and the prototype will be extended to support richer querying and scalable distribution.

References

[1] J. Bardram and M. Mogensen. DOLCLAN - Middleware Support for Peer-to-Peer Distributed Shared Objects. In *7th Int'l Conf. on Distributed Applications and Interoperable Systems*, pages 119–132, June 2007.

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

[3] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004.

[4] C. Bryce, M. Oriol, and J. Vitek. A Coordination Model for Agents Based on Secure Spaces. In P. Ciancarini and A. Wolf, editors, *Proc. 3rd Int. Conf. on Coordination Models and Languages*, volume 1594, pages 4–20, Amsterdam, Netherland, 1999. Springer-Verlag, Berlin.

[5] N. Carriero and D. Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.

[6] Commission of the European Communities. European interoperability framework for pan-european egovernment services, 2005.

[7] Commission of the European Communities. An action plan for a European e-Health area (COM (2004) 356 final), 2006.

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

[9] 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), Feb. 2007.

[10] C.-L. Fok, G.-C. Roman, and G. Hackmann. A lightweight coordination middleware for mobile computing. In R. De Nicola, G. Ferrari, and G. Meredith, editors, *Proceedings of the 6th International Conference on Coordination Models and Languages (Coordination 2004)*, volume 2949, pages 135–151, Pisa, Italy, 2004. Springer-Verlag.

[11] D. Gelernter. Generative Communication in Linda. *ACM Trans. Prog. Lang. Syst.*, 7(1):80–112, 1985.

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

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

[14] M. Mamei, R. Quaglieri, and F. Zambonelli. Making tuple spaces physical with rfid tags. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 434–439, New York, NY, USA, 2006. ACM Press.

[15] F. Martín-Recuerda. Application Integration Using Conceptual Spaces (CSpaces). In *1st Asian Semantic Web Conf.*, pages 300–306. Springer Verlag, Sept. 2006.

[16] A. Murphy, G. Picco, and G. Roman. Lime: A coordination middleware supporting mobility of hosts and agents. Technical report, 2003.

[17] L. Nixon, E. Simperl, O. Antonenko, and R. Tolksdorf. Towards Semantic Tuplespace Computing: The Semantic Web Spaces System. In *22nd ACM Symposium on Applied Computing*, 2007.

[18] L. Nixon, E. Simperl, R. Krummenacher, and F. Martin-Recuerda. Tuplespace-based computing for the Semantic Web: A survey of the state of the art. *Knowledge Engineering Review*, 23(1), March 2008.

[19] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Working Draft, October 2006.