A Semantic Publish-Subscribe Coordination Framework for IHE based Cross-Community Health Record Exchange

Visara Urovi, Alex C. Olivieri, Stefano Bromuri,
University of Applied Sciences Western Switzerland, Sierre
{visara.urovi, alex.olivieri, stefano.bromuri} @hevs.ch

Nicoletta Fornara,
Università della Svizzera Italiana, Lugano, Switzerland
nicoletta.fornara@usi.ch

Michael I. Schumacher
University of Applied Sciences Western Switzerland, Sierre
michael.schumacher@hevs.ch

ABSTRACT
This paper presents a semantic agent coordination framework for the automatic exchange of Electronic Health Records (EHR). The framework enables health organizations that comply with the existing interoperability standards, as proposed by the Integrating Healthcare Enterprise (IHE), to dynamically connect in a P2P network. A set of autonomous agents and a set of distributed coordination rules are used to coordinate the agents in the search of specific health records. The framework models the interactions among the communities as a combination of the tuple centre agent communication model with the publish-subscribe paradigm and semantic web technologies. In order to illustrate the scalability of our approach, we evaluate the proposed solution in distributed settings. The contribution of this work is that it proposes the coordination mechanisms for IHE compliant communities to dynamically connect and share EHR of patients[1].

Categories and Subject Descriptors
H.4 **[Information Systems Applications]**: Miscellaneous

General Terms
Design, Experimentation

Keywords
Document exchange, EHR, IHE, Semantic Interoperability, Coordination, TuCSoN, OWL.

1. INTRODUCTION
The Electronic Health Records (EHRs) refer to the electronic collection of health information data about individual patients [15]. The advantage of EHRs is that information can be quickly transferred and linked to best-practice guidelines to provide decision support [14]. EHR based systems often operate in a closed environment where patient's EHRs can be exchanged only within one organization. As the focus of health care delivery shifts from specialist centers to community settings [7], new research approaches are focusing on the integration of such records across the institutional boundaries [33]. Integrating the Healthcare Enterprise (IHE) [20] is an initiative focusing on integration of healthcare information systems. IHE makes a major contribution to integration of these systems and enjoys high acceptance due to its practical complement to existing standards such as HL7 CDA, a standard supporting message-based information exchange of medical data. The significance of the IHE profiles stands on the fact that they are constantly checked against practical experiences and are continuously adapted [33]. Despite this, IHE lacks features to handle dynamic scenarios where caregivers can dynamically connect and exchange data [17], and mechanisms on how patient's data are found and exchanged are yet to be defined.

To address these problems, a system is needed where up-to-date patient's health records can be shared without prior knowledge of the health organizations that produced the data. In particular, semantic description of content has been recognized as a powerful tool for data sharing [16], that, combined with agent-based computing, can contribute to automate the collection and processing of patient's EHRs. Agent-based systems can perform distributed communication and reason with semantic knowledge thus enabling EHR sharing between such heterogeneous systems. Furthermore, agent coordination models, such as tuple centres [26], that focus on decoupling the interaction amongst the actors, can contribute on making the different health actors more interoperable. On top of coordination models, a Peer to Peer (P2P) [2] solution can link the heterogeneous health organization's systems as peers, allowing them to interact on top of existing network configurations and remove any central dependency for sharing patient EHR. P2P networks have the advantage that they scale up for a large number of peers and are more reliable that single server architectures. To further support a large-scale architecture, a publish-subscribe paradigm [11] can be used to model EHR update notifications with the advantage that it allows loose coupling between the different health organizations and it supports many-to-many communication between them.

In this paper we propose an orthogonal solution to the existing IHE profiles that deal with EHR exchange. We propose a semantic publish-subscribe coordination framework that enables various health organizations to dynamically discover and exchange EHRs of interest. The contribution of this work is that it provides a general coordination framework that extends the existing interoperability standards with the ability to dynamically exchange EHR between different health organizations. In particular, we extend the current IHE profiles with the ability to dynamically connect to other communities that comply with such profiles. Furthermore we use semantics to automatically interpret the shared knowledge between different healthcare environments and to define the agent-based coordination mechanisms that coordinate the exchange and interpretation of such medical knowledge.

The rest of this paper is organized as follows: Section 2 describes our motivating case study, Section 3 summarizes the background work for our coordination framework; Section 4 describes how we engineer the agent-based coordination framework to deal with the

dynamic exchange of EHRs; Section 5 describes the implementation of our proposed system architecture; Section 6 evaluates the performance of the system; Section 7 discusses relevant related work in the area of Semantic Interoperability and Multi-Agent Systems that provide solutions for the eHealth field; Finally, Section 8 summarizes the work and draws the lines for future works.

2. MOTIVATING CASE STUDY

Our scenario is based in Switzerland, a federal country divided into 26 counties called cantons. The health system of Switzerland is a combination of public (i.e. hospitals) and private systems (i.e. doctors in private clinics) and health conditions can be treated in any of the competent healthcare providers. The Swiss Government has recently recommended the adoption of IHE profiles to achieve interoperability. The first pilot deployments have just been released, such as the eToile project [12] in Geneva.

In this scenario, Mrs. Roux from Lausanne, canton Vaud, needs urgent hospital care due to a strong chest pain in Sierre, canton Valais, where she is on holiday. Previously she had a heart surgery in the Hospital of Lausanne, which is also her home community and keeps all the updates on Mrs. Roux health records. The home community does not necessarily have a copy of all the generated documents for Mrs. Roux, but it knows where every document is stored. Mrs. Roux has signed a privacy consent that allows the Hospital of Lausanne to share her health records with other communities. The identifier of the insurance card of Mrs Roux is used to search for her data in the Hospital of Lausanne. Based on the privacy consents given by Mrs. Roux, the query returns meta-data held on Mrs. Roux' records (attributes describing her health documents but not the documents).

The doctor who visits Mrs. Roux can view the discovered information and can consult the documents of interest by retrieving the content from the community where these documents are stored. This is possible because Mrs. Roux, through a web application, gave to medical doctors the right to access her medical data. The doctor asks for further tests to be carried out in the hospital of Sierre. After Mrs. Roux' agreement, the tests and the doctor's diagnosis are notified to the hospital of Lausanne.

The general practitioner (GP) and the cardiologist treating Mrs. Roux are subscribed with the hospital of Lausanne to receive notifications on the new generated data on Mrs. Roux. While the GP is interested to any new generated record, the cardiologists expresses her interest to be notified only if problems involving the heart are detected and recorded within the EHR of Mrs. Roux. Hospital of Lausanne automatically notifies the case to her two doctors. Next time, when Mrs. Roux visits such facilities, her doctors can view the relevant new information generated on Mrs. Roux.

3. BACKGROUND WORK

Motivated by our scenario, we define a semantic agent coordination framework that combines a P2P technique with semantic publish and subscribe methodologies to support health communities to dynamically connect, search, send and notify relevant updates on patient data. The choice of the P2P network is motivated by the need to have a scalable solution for a large number of health communities. This work extends upon the existing IHE profiles that propose a solution for EHR exchange. In this section we explain the background works related to our framework.

3.1 IHE Limitations in EHR Exchange

The IHE defines technical frameworks for different clinical and organizational domains. An important part of these frameworks are the integration profiles, which are defined in terms of actors and transactions. Actors are components that act on information associated with clinical and operational activities in the enterprise. Transactions are interactions between actors that communicate the required information through standards-based messages. There are many IHE profiles that address interoperability between health care systems. We focus on four profiles that propose solutions for the exchange and notification of EHRs, namely Cross-Enterprise Document Sharing (XDS), Cross-Community Access (XCA), Cross-Community Patient Discovery (XCPD) and the Document Metadata Subscription (DSUB) profiles.

The XDS [20] profile defines how health enterprises can inter-operate to share patient-relevant documents by working as one community with the same set of policies, patient identifications and security mechanisms. In XDS, the data produced on a patient can be stored in a distributed way. However a set of meta-data regarding the record must be stored in a central registry which is later used to find these documents.

Since XDS does not resolve document sharing among multiple communities, the XCA profile specifies how medical data held by other communities can be queried and retrieved. XCA assumes that communities have pre-established agreements and knowledge of one another. It also assumes that the community which initiates a query towards another community, can determine the correct patient identifier of the patient under the authority of the receiving community [19].

The XCPD profile can locates communities which hold patient's relevant health data and translate patient's identifiers across communities, however it does not automate the discovery of communities and it still requires communities to have pre-established agreements for exchanging the documents. In fact, the actor searching for documents in the cross-community must know beforehand which communities to contact.

Finally, the DSUB profile describes the use of subscription and notification mechanism within an XDS community and across communities [29]. The subscription matches the metadata of a new published document for a given patient, and results in the delivery of a notification. The limitation here is that the subscription is a restriction upon the metadata defined by the publisher. Currently, it is a common practice for different communities to use different domain-specific coding lists (these are the terms used to annotate the clinical documents). Thus, in practice, if these codes are not shared by the two, the publisher and the subscriber community have difficulties in locating the documents.

If we were to model our case study only with the current IHE profiles, we would encounter several limitations. We could use XCPD to locate Mrs. Roux identity in the hospital of Lausanne. However, in order to exchange the data, the two hospitals should have an agreement and the necessary integration in place to allow data exchange between the two. This is because IHE profiles define interactions as a simple message exchange and, in order for communities to interact, they need to know a priory which community to address. Nowadays this is not the case as patients move considerably and they may seek medical attention in different healthcare communities that may not know each other.

Even if we assume that every IHE community can achieve point to point integration with every other IHE community, we still have the problem of defining proactive propagation of health information in other communities. In fact, we can use the XCA profile to query the Hospital of Lausanne about Mrs. Roux' data and DSUB to subscribe to events of interest. However, in order to enable dynamic propagation of the updates, further integrations should take place so that the communities either share the same terminologies or map them in a way that the subscriber to some EHR uses the codes of the subscribing community. We can imagine that in the near future, patients will use many different health services that do not necessarily share this level of integration.

Motivated by the lack of support mechanisms in the above IHE specifications, we complement the IHE approach to enable communities to exchange data without prior knowledge of each other. We assume that a set of IHE compliant healthcare systems will be using our framework to discover and exchange information with other healthcare systems. Our proposed solution extends upon the existing XDS profile to allow any health community to dynamically exchange EHR without undergoing the point to point integration that would be needed with the current IHE models. The dynamic nature of the network we want to create requires a coordination model able to provide uncoupling among the communities. This is why we choose a coordination middleware, such as TuCSoN [22], which provides a general approach for combining semantics with coordination of messages for the purpose of EHR exchange. One limitation of TuCSoN is that it has an unstructured P2P model which considerably increases the time to answer semantic queries as reported in [30]. To overcome this limitation, we define a structured P2P model to reduce the number of propagated semantic search queries and to improve the performances.

3.2 Coordination in TuCSoN

TuCSoN [26] is an agent coordination infrastructure based on the concept of blackboard systems, like Linda [13]. In TuCSoN, interactions are mediated by shared tuple centres. The interacting entities of TuCSoN can use the tuple centres to write, consume and read tuples without necessarily having to synchronize (time decoupling), share the same space (space decoupling) or even without knowing each other (name decoupling) [13]. In addition to these advantages, the interacting entities communicate by writing and reading RDF triples, making them schema decoupled too [4]. With these advantages, the mediation mechanisms improve considerably the interoperability of EHR exchanging systems as opposed to the simple message exchange. Apart from reading, writing and consuming semantic tuples, TuCSoN allows to engineer additional primitives to coordinate the interacting entities.

Agents in TuCSoN interact through tuple centres by inserting (out operation), reading (rd operation) and consuming (in operation) tuples. Tuples are read and retrieved associatively. In order to read or retrieve a tuple, a tuple template has to be specified so that it can be used to find the requested tuple among all the existing tuples in the tuple centre [22]. The tuple centres can be syntactic, meaning that the structure of the tuple templates are known to the agents, or semantic, meaning that the information is produced and consumed following an ontology model. The behavior of the tuple centres is programmable with a set of coordination rules expressed in the ReSpecT language [25]. Using ReSpecT it is possible to define reactions that specify how a tuple centre reacts to incoming/outgoing communication events. The reaction rules syntax is defined as follows:

reaction(action, conditions, react)

where *action* is an operation made in a tuple centre (such as *out(tuple)*), *conditions* specify the conditions that should be verified in order to execute the *react* and *react* specifies a set of communication events that take place as a consequence of the performed action. In a ReSpecT *reaction* it is also possible to specify communication events *(out, in, rd)* towards other tuple centres. This is possible because tuple centres are hosted in nodes and distributed in a network [6]. Every node can host many tuple centres and there can be direct communications between distributed tuple centres by addressing the right tuple centre. In addition to point to point communications between tuple centres, using a structured P2P network enables us to search the location of the required information.

3.3 OWL DL and Query Language

TuCSoN uses the OWL Web Ontology Language [17] to model semantic tuple centres in terms of domain ontologies and objects [22]. OWL is a practical realization of a Description Logic known as *SHOIN(D)* [18]. Using OWL it is possible to define classes (also called concepts in the DL literature), properties, and individuals. An OWL ontology consists of a set of class axioms that specify logical relationships between classes, which constitutes a *TBox* (Terminological Box); a set of property axioms to specify logical relationships between properties, which constitutes a *RBox* (Role Box); and a collection of assertions that describe individuals, which constitutes an *ABox* (Assertional Box).

Classes are formal descriptions of sets of objects (taken from a non empty universe), and individuals are names of objects of the universe. Properties can be either object properties, which represent binary relations between objects of the universe, or data properties, which represent binary relationships between objects and data values (taken from XML Schema datatypes). Class axioms allow one to specify that subclass ($\sqsubseteq$) or equivalence ($\equiv$) relationships hold between certain classes and the domain and range of a property. Assertions allow one to specify that an individual belongs to a class: C(a) means that the object denoted by a belong to the class C; and that an individual is related to another individual through an object property: R(b,c) means the object denoted by b is related to the object denoted by c through the property R. Complex classes can be specified by using boolean operations on classes: $C \sqcup D$ is the union of classes, $C \sqcap D$ is the intersection of classes, and $\neg C$ is the complement of class C. Classes can be specified also through property restrictions: $\exists$ R.C denotes the set of all objects that are related through property R to some objects belonging to class C at least one; if we want to specify to how many objects an object is related we should write: $\leq$ nR, $\geq$ nR, = nR where n is any natural number.

To realize the framework presented in this paper, we need to express the conditions part of the reaction rules with instructions that reason on the semantic model. Thus, every condition is evaluated by querying the reasoning services of an OWL DL reasoner, sometimes Prolog predicates are used to construct some specific function. Given that there is not an oficial standard query formalism for OWL DL, we adopt the following formalism that is inspired from [5] and which allows to express the queries that are

available in the DL Query tab of Protege[2]. In our implementation those queries are executed using the JENA API:

? - C ⊑ D ⇒ true/false checks the subclass relationship;

? - C ≡ D ⇒ true/false checks class equivalence;

? - C ⇒ true/false checks if the class is satisfiable;

? - C(a) ⇒ true/false instance checking;

? - C(*)⇒{a1....an} retrieval, C can be a complex class.

3.4 Structured P2P

A P2P system consists of distributed and interconnected nodes able to self-organize into network topologies without requiring the support of a centralized authority [2]. P2P networks can vary from unstructured to structured topology. Unstructured P2P networks use flooding to search for peers with the disadvantage that the messages considerably increase with the number of peers. In large-scale networks, in order to reduce the number of exchanged messages, structures can be established within the P2P network. In a structured network, a query is not forwarded to all peers, as in unstructured P2P systems, but only to a selected set of peers. The selection is based on a distance metric that finds the peers that are close neighbors for a given key. The selected peers can be queried either to store or retrieve data. The structured P2P networks can be defined using distributed hash tables (DHT) which store and search data based on pairs (key, value). The realizations of DHTs, such as Kademlia [21] used here, have the advantage of scaling logarithmically with the number of peers in the system. In fact most of DHT based systems have equivalent search performance cost which is O(log N), where N is the size of network [2].

4. P2P AGENT COORDINATION FRAMEWORK FOR CROSS-COMMUNITY EHR EXCHANGE

Our framework describes semantically the knowledge bases of health communities and coordinates their interactions in cross-community EHR exchange. Given that different communities may have different ways to present their information, we specify an ontology that is used to define the knowledge base of every community. This enables communities to interpret and reason on the data that are generated from different healthcare providers. In case two communities adopt different ontologies, mechanisms for ontologies reconciliation may be adopted. We model the concept of community as an entity that exposes a set of services and its policies to enable interactions with other communities. An agent coordination architecture is used to coordinate the interactions across communities. Fig. 1(a) shows the architecture for one single community. The Policy Tuple Centre contains the coordination primitives in terms of action-reaction rules that are used to mediate interactions among the communities. The coordination primitives are coupled to a semantic tuple centre and are specified using the ReSpecT language [26].

Fig.1(a) shows how every community has its own Policy Tuple Centre containing a replica of the coordination primitives. Currently, we use a soft model of agency where agents simply react to specific messages exchanged in the tuple centers, as opposed to a hard model of agency where the agents have complex cognitive models to perform complex reasoning.

[2] http://protege.stanford.edu/

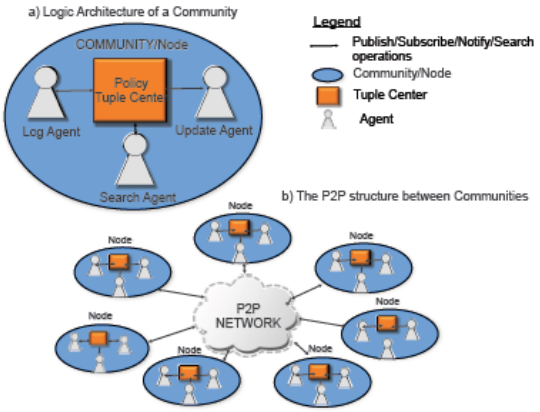

Figure 1. The logic architecture of the community nodes

Nevertheless they are essential to keep the distribution and the autonomy of all the communities of the system. We delegate them specific tasks that are performed when specific events happen in a tuple centre. Thus, in every community, we specify three agents that are responsible for performing different actions. Fig.1(a) shows the Log Agent which is responsible for logging the different queries performed by other communities, the Search Agent which is responsible for performing search queries in the P2P network and the Update Agent which is responsible for sending and receiving updates from/to other communities.

Fig. 1(b) shows the health communities connected in a P2P network which allows us to have a scalable mechanism for search queries and event notification. Each Node in the P2P network represents a community and the physical healthcare system behind it. The communities share in the P2P only meta-information about which community holds the data of a patient id. A community that is interested in finding data about a specific patient, queries the P2P to find which community holds these data performs a direct query to the community to receive the right information. There are other security and trust issues arising from the use of a P2P network [3] which we are currently investigating and are subject of future publications.

4.1 The Community Ontology

Every community has its own knowledge base. Other communities can query or subscribe to receive updates happening in more than one knowledge base. Fig.2 shows the classes and the data and object properties of the OWL Community Ontology[3] that is used to create those knowledge bases. The classes are all disjoint. The RBox of the Community Ontology contains the following object properties (where the name of a property is followed by its domain and its range). The TBox contains the subsequent axioms that define cardinality restrictions for the defined properties: The Community provides a set of Services, follows a set of Policies, cares about Patients and subscribes to other communities by specifying Filters. Every Patient in a Community has a set of Documents that describe its health records. The subscriptions to Patient's documents are done through Filters. The Policies of a Community relate to Patients or to the Role that an Actor assumes within that Community.

[3] The full ontology can be found in http://aislab.hevs.ch/assets/OntologyCommunity.xml

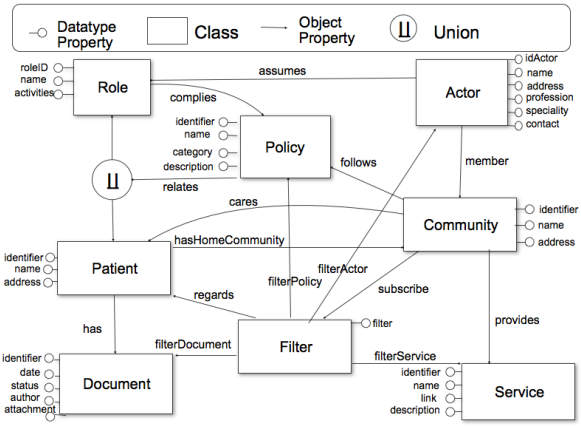

Figure 2. The OWL Community Ontology

has: Patient →Document; InvFun(has);

cares: Community → Patient; InvFun(cares);

subscribe: Community → Patient

member: Actor → Community

provides: Community → Service InvFun(provides);

follows: Community → Policy; InvFun(follow);

assumes; Actor → Role complies: Role → Policy;

relates: Policy → (Patient ⊔ Role); Fun(relates);

hasHomeCommunity: Patient → Community;

Fun(hasHomeCommunity);

subscribe: Community → Filter;

filterPatient: Filter → Patient; Fun(filterPatient);

filterDocument: Filter → Document;

filterService: Filter → Service;

filterPolicy: Filter → Policy;

regards: Filter → Actor;

Document ⊑= 1 has⁻; Service ⊑= 1 provides⁻;

Patient ⊑= 1 cares⁻; Actor ⊑= 1 member;

Policy ⊑= 1 follows; Policy ⊑= 1 complies;

Filter ⊑= 1 subscribe⁻ ⊓ (1 filterPatient ⊔ 1 filterDocument ⊔ 1 filterService ⊔ 1 filterPolicy ⊔ 1 regards);

Documents are generated and stored within a community and relate to a specific patient. Every document is described with a set of properties which indicate the author and the content of the document. The community that generates such documents can also update their status by making documents obsolete or deleting them. A Community has many Actors which can assume more than one Role. The actors are the users of the system, therefore they play roles such as a cardiologist, nurse, pharmacist, administration ect. The actors must act in the system by complying with the Policies of the Community. Such Policies define the actions that every role is allowed to perform.

When other communities are interested in updates regarding Patient's Documents, Policies or Actors of another Community they subscribe to it by defining a Filter. The Filter is stored in the knowledge base of the community that receives the subscription. In this way, when changes happen in the knowledge base of one community, they are notified to the other interested communities. The same filter can belong to more communities and a community can subscribe to different information using more filters.

4.2 The Subscribe Language Model

In our previous works [30, 31] communities could subscribe to any new EHR generated about a patient in another community. The implications of this were that every new EHR record, even those not significative for specialist medical decisions, were propagated. In order to allow the communities to be notified only about relevant EHRs of patients, we define a subscribe language model. The language can be used to subscribe with other communities. It enables actors (i.e. doctors) to specify what are the events of interest with respect to the multiple complications that may arise in a patient. For example, the cardiologist of Mrs. Roux is not interested in EHR regarding her dermatologic visit, however he is interested to see any document relating to heart complications.

Let C= $\{C_1.....C_n\}$ be the community set and P=$\{P_1,....Pk\}$ be the set of all patients. Every community in C will generate new documents in form of events E=$\{E_1.....E_m\}$ and it will follow a semantic knowledge base{K, R} where K are the concepts and R are the relationships between them. Every produced event Ei is defined as follows:

$$E_i = \{event(Id, C_j, Patient, Type, \{A_1.....A_n\}) \mid C_j \in C,$$

$$Patient \in P, Type \in K, \{A_1....A_n\} \in \{K,R\}\}$$

which specifies an event E_i in terms of an identifier *Id*, a community C_j that generates it, a reference to the Patient, a type Type that is an instance of the concepts K (i.e. document or policy) defined in the semantic knowledge base and a set of attributes that are instances of the semantic data. In order to allow other communities to express their interest on specific events, we define filters F. Filters are defined as part of the subscription and can be specified as follows:

F_i={filter(IdFilter, Type, Language) | Type K}

Language has the following production rules:

S → or(A) | and(A)

A → B | S

B → C | {C}

C → attribute(Attr, Op, V)

C →

Op → = | < | > | ∀

By specifying the filters at the moment of subscription, the caregivers within the communities will be able to restrict to a subset of events of interest regarding their patients. The above specified language uses triples *attribute*, *operation* and *value* that,

if combined with *and* and or operators, enables caregivers to restrict the notifications of new EHRs to only of those of interest. For example, following the Scenario presented in Section 2, the cardiologist of Mrs. Roux specifies the following filter:

F_1= filter (idf1, document,

and(attribute(bodypart, =, heart),

or(attribute(documentType, =, discharge),

attribute(documentType, =, laboratory),

attribute(documentType, =, emergency))))

The above filter specifies that any document that concerns the heart and it is either a discharge, laboratory or emergency report should be notified. Once the filters are specified, a subscription S is defined as :

S= {subscribe (Subscriber, Patient, Fi) | Subscriber ∈ C,

Patient ∈ P, F_i ∈ F}

For every new event, the subscriptions are checked to identify the communities that are interested to the notification of such events. Inspired by the work presented in [32], a set of declarative rules that parse the filter language and verify if the new event satisfies the subscription pattern that is specified in the filter:

Table 1. Matching Events with the Filters

Single attribute matching: attribute(Attr, Op, V)	**match(attribute(A, OP, Value), List, Result):- member(attribute(A, Value2), List), operation(OP, Value2, Value), append([attribute(A, Value2)],Result, R2), Result=R2.**
And matching: and(Filter)	**match(and([H|T]), List, Result):- match(H, List, R1), append(R1, Result, InterRes), match(and(T), List, R2), append(R2, InterRes, R3), Result = R3.**
Or matching: or(Filter)	**match(or([H|T]), List, Result):- (match(H, List, Result1), append(R1, Result, InterR), Result=InterRes; match(or(T), List, R2), append(R2, Result, R3), Result=R3).**
Possible matching operations	**operation("=",Value, Value).** **operation(">", Value1, Value2): Value1>Value2.** **operation("<", Value1, Value2): Value1<Value2.** **operation("∀", _ , _).**

Table 1 shows three *match/3* predicates:

1. **Single Attribute Matching** checks if an attribute *A* can be found in the list *List* that describes the properties of an event in terms of attributes-value pairs. Then, the operation/3 predicates are used to compare the value *Value2 that is* found in the list of attributes describing the event (in *List*) against the specified operation *OP* and value *Value* that are specified within a filter;

2. **And Matching** recursively checks that all of the triples a*ttribute(A, OP, Value)* that are specified within a filter satisfy the *check/3* predicate.

3. **Or Matching** recursively checks that at least one of the triples a*ttribute(A, OP, Value)* that are specified within a filter satisfy the *check/3* predicate.

In the following sections we show how the above specified language can be used to semantically match the events describing documents against the filters specified by different communities and whose dictionary upon medical terms might not be uniform.

4.3 Policy Tuple Centre

The Policy Tuple Centre (PTC) mediates the requests of agents to connect, subscribe to notification of events and/or to search data in the P2P network. There is one PTC in every community and all the communications towards a community are made in its PTC. The Log agent is used to log all the interactions within the PTC of a community. A protocol to extract the history of how documents are accessed and exchanged in the P2P network is subject to future work. The PTC specifies the coordination primitives for subscribing and unsubscribing to data generated in other communities and the primitives to search patient data within the P2P network.

4.3.1 Distributed Data Search in Other Communities

The A community can search other communities and patients by generating queries in the P2P network. The Search Agent answers to search queries by first querying the P2P network about the community that holds the data of a patient and later send a request query to the home community of the patient. The queries indicate the sender, the community that is requesting the data and a list of criteria to be used for the search. The behavior of the Search Agent can be summarized as follows:

1. The Search Agent listens to *search* and *reply* messages;

2. In case of a *search* message, it searches the information by performing a *rd* primitive in the PTC of its community. If the searched data are contained in the Knowledge Base of the Community, the retrieved information is given to the actor who performed the request. Otherwise, the research is forwarded automatically to the P2P network which provides the link to the community where the information is held. The Search Agent performs a request message in the Community holding the information to get the information required.

3. In case of a *reply* message, it means that another community replies with the searched results. The agent provides the results to to the requesting actor and returns to step 1.

In the P2P search some criteria may not be specified. For example, the patient may not be able to produce a home community therefore the homeCommunity of the patient may be unknown. The coordination primitive for requesting the health data of a patient in another community is defined as follows:

reaction(out(request(community, patient)), ⊒⊓⊔≡∃∀¬ ⊑→←

?- Patient(patient) ⇒ true,

?- Policy ⊓ (∃ relates.{patient}) ⊓

(∃ category.{"filesharing"}) ⊓

(∃ description.{"consent"}) ⇒ true,

?- Documents ⊓ (∃ has.{patient}}(*) {d_1....d_n},

out(reply(community, actor, {d_1.....d_n}))).

The above primitive is activated when the Search agent of a community requests patient data in another community. The PTC of the community holding health information about the patient checks that the identified patient belongs to its knowledge base, checks that exists a policy describing the patient's consent for file sharing, and finds the documents instances that relate to the patient and performs a reply message in the PTC specified by the Search agent.

4.3.2 Subscribing to Community Events

The Communities can subscribe to events generated by other communities by using the language specified in Section 4.2. A community may subscribe to: events regarding the documents of a patient, to changes on the services that a community offers, to the changes in the policies that a community follows and subscriptions to the changes regarding the actors of a community. In this paper we exemplify only the coordination mechanisms for receiving updates on patient's documents from other communities, however, the other subscriptions are specified in a similar way.

In order to receive updates from other communities, the Update Agent of one community interacts with the other communities to subscribe/unsubscribe to the interested event notifications. The requests to subscribe/unsubscribe with other communities are first generated by the Actors. They request to subscribe in the home community of the patient with a specific Filter. Upon such request, the Update Agent of a community executes the following steps:

1. Writes the *subscribe*/*unsubscribe* message in the PTC of the home community of the patient. If the patient's home community is not known, then it first searches the homeCommunity of the patient within the P2P. It also listens to *add*, *remove* messages in its own PTC in case other Communities request to *subscribe*/*unsubscribe* to patients in the community where the Update Agent operates;

2. In case it listens an *add* message, it means that a new community has subscribed to specific events generated in the community where the Update Agent operates. The agent adds the new community, the filter describing the subscription and the subscribe relationship to the knowledge base and returns to step 1;

3. In case it listens a *remove* message, it means that a community is unsubscribing to specific events generated in the community where the Update Agent operates. The agent removes the community, the filter and its subscribe relationship from the knowledge base and returns to step 1.

The coordination primitive for subscribing to patient updates is specified as follows:

reaction(out(subscribe(community, patient, filter)),

?- Patient(patient) ⇒ true,

?- Policy ⊓ (∃ relates.{patient}) ⊓

(∃ category.{"filesharing"}) ⊓

(∃ description.{"consent"}) ⇒ true,

out(add(subscribe(community, patient, filter)))).

The above primitive is activated in the PTC of a community when another community wants to subscribe to the data of a given patient with a specified filter. The PTC checks that the identified patient is already contained in the knowledge base and that it exists a policy describing the patient consent into sharing its own files (the complex DL class is satisfiable) and generates an add message for the Update Agent. When a new document regarding a patient is generated in the network, the home community of the patient is notified. If the document is generated in the home community or an update about a patient arrives in the home community, such update is propagated to all the interested subscribers. The following coordination primitive propagates the data to the subscribers of a patient:

reaction(out(event),

event=event(id, c, patient, document, {$attr_1$,....$attr_n$}),

?- Document(document) ⇒ true,

?- Patient(patient) ⇒ true,

semanticExpansion({$attr_1$...$attr_n$}, {$attr_1$,...$attr_n$,...$attr_{n+j}$})

?- {$attr_1$...attrn}⊑ {$attr_1$,...attrn,...attrn+j} ⇒ true

event=event(id, c, patient, document, {$attr_1$,..$attr_n$,..$attr_{n+j}$})

(∃ homeCommunity⁻ .{patient}(*)) ⇒ {c},

?- (Community ⊓ ∃ subscribes.{?filter})) ⊓

(Filter ⊓ ∃ filterPatient.{patient}))(*) ⇒ {s_1...sn}

findInterested(event, {s_1...s_n}, {c_1...c_n}),

out({c_1...c_n}, update(patient, document))).

The above coordination primitive specifies that if an event specifying a new document is generated in the home community, all the subscribers that are interested to this event should be notified. The *semanticExpansion/2* predicate takes the attributes of the *event* and uses the SNOMED OWL Ontology[4] to extend the list of event's attributes with other attributes. These new attributes are the super classes, the super properties or equivalences of the starting attributes of the original event. By expanding the event in this way we obtain that, even if the specified filter uses different synonymous terms or similar concepts, we are still able to notify the event to the interested community. In this way we can still detect more generic or equivalent terms that may have been used by different actors e.g. *Heart Attack*, *Infarction of Heart*. The *findInterested/3* predicate takes the expanded *event* and the list of *subscriptions* and uses the *match/3* predicates to identify which of the subscribers are interested to this particular *event*. No agents are used in this operation as the PTC can directly update other

[4] www.ihtsdo.org/snomed-ct/

PTCs by using TuCSoN coordination primitives. Similar coordination primitives are defined to propagate updates to the home community and to unsubscribe from other communities.

5. IMPLEMENTATION

The implementation of our framework is based on the TuCSoN semantic tuple centres as defined in [22]. In order to improve the search of semantic information in the distributed network, we create a P2P network over TuCSoN using ToM P2P JAVA library[5]. Additionally, we interface. with openXDS[6], an open source implementation of the XDS profile, in order to have documents stored and retrieved in an IHE compatible manner. All these infrastructures are JAVA based.

Every community is represented with a TuCSoN node and has its own semantic knowledge base where the semantic queries and reasoning are performed. When users add new information in the system, OWL assertions are generated and added to the knowledge base. For every assertion, a defined number of hash tags are created in the distributed hash table of the P2P network. In case of the addition or modifications of documents, IHE compatible meta-data are generated to be stored in the registry of the XDS profile and the same meta-data are stored as semantic data in the community knowledge base. Updates to the meta-data of the documents of a patient are propagated towards the home community of the patient and to the subscribed communities. We assume that the actual fetching of the documents is realized using one of the existing IHE profiles (XCA already addresses this issue).

Each Community operates with three Agents: a Log Agent, an Update Agent and a Search Agent. The agents react to tuples generated by external User Agents or to tuples generated by the PTC of their community. The agents use in and rd operations to search in the PTC the messages that trigger their tasks and, at their completion, they perform out operations to write the results in the PTC. The reaction primitives are specified by calling JAVA code from reactions specified in ReSpecT. This is possible because TuCSoN is based on tuProlog [8], a JAVA based implementation of Prolog. that allows a seamless integration between JAVA code and Prolog predicates.

For example, the coordination primitive to propagate new information about a patient to the subscribed communities is implemented as follows:

```
1. reaction(out(event(Id, Patient, Document, AttributeList)),
2.    in(event(Id, Patient, Document, AttributeList)),
3.    get_semanticKB(KB),
4.    KB ← getBase returns Base,
5.    KB ← getModel returns Model,
6.    java_object('ExpandList', [], MyExpandedList),
7.    MyExpandedList ← expandList(AttributeList) returns
                              ExpandedAttributeList,
8.    text_concat([
          SELECT ?community ?filterString
          WHERE{?community subscribe ?filterType,
                ?filterType filterPatient, Patient,
                ?filterType filter ?filterString.}],
      MyQueryString),
9.    java_object('QueryCreator',[Base,Model], MyQuery),
10. MyQuery ←createQuery(MyQueryString) returns FilterList,
11. NewEvent = event(Id, Patient, Document,
                              ExpandedAttributeList),
12. findInterested(NewEvent, FilterList, Subscribers),
13. out_tc(Subscribers, publish(Patient, Document)) ).
```

The above reaction rule specifies that when an *out* of an *event* is made into the tuple centre *(Line 1-2)*, then the reference to the JAVA object representing the semantic knowledge base *KB* is used (the ← notation represent a call to a JAVA module) to obtain the URI and the model *Model* of the ontology *(Line 3-5)*. We use an *ExpandList* JAVA module to expand the list of attributes of the event *(Line 6)*. In turn, *ExpandLis*t uses the following SPARQL queries and the SNOMED OWL Ontology to determine how to expand the attributes:

```
PREFIX snomedNS:<http://www.ihtsdo.org/snomedct.owl#>
SELECT DISTINCT ?argumentExtension
WHERE{
      { snomedNS:argument
            rdfs:subClassOf ?argumentExtension .}
  UNION{ snomedNS:argument
            rdfs:description ?argumentExtension .}
  UNION{ ?class rdfs:description snomedNS:argument .
        ?class rdfs:description ?argumentExtension .}
  UNION{ ?argumentExtension
        rdfs:description snomedNS:argument .} }
```

The above SPARQL query is executed when an attribute is describing a concept. In case the given attribute identifies a concept we find all the super concepts and the alternative descriptions of that concept and extend the list with these new attributes. In case the attribute identifies a description, we find the related class and all descriptions of this class and add these to the extended list.

Returning to the starting reaction, we save the new list of attributes in the variable *MyExpandedList (Line 7)*. Moreover we create a String representing a SPARQL query which is used to query all the subscriptions stored in the knowledge base that relate to the *Patient* specified with the *event (Line 8)*. This string is then passed as argument to the method *createQuery* of the JAVA object *QueryCreator (Line 9)*. This method returns the list of all filters that match with the query *(Line 10)*. Then the *event* is modified with the expanded attribute list *(Line 11)*, and it is used by *findInterested* to check which filters match the event in order to get the list of all communities subscribed to the information

[5] tomp2p.net/
[6] www.projects.openhealthtools.org/sf/projects/openxds/

contained in the event *(Line 12)*. Finally, the tuple *publish(Patient, Document)* is sent to all communities subscribed *(Line 13)*.

6. EVALUATION

We evaluated the proposed solution on the Amazon Cloud with thirteen micro version virtual machines that generated the peers and the data in the P2P network. One hundred peers at a time were added in the P2P. We measured the time to find information held by an unknown community. This time includes the time to search in the P2P network which community to contact and the time to receive the information from the contacted community.

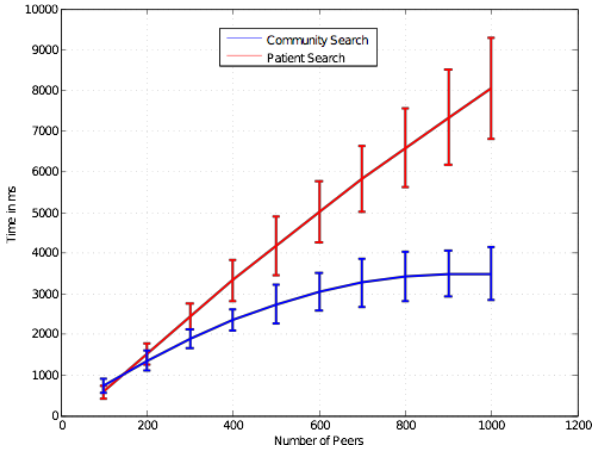

Figure 3. Cross-community and patient's identity time search

The Fig.3 shows the time to find information for different queries with respect to a growing number of peers in the network. Two different queries are performed. The first query is a search for a patient where we query twice the P2P network: first to find the home community for a patient and then to find the details on the home community. The second query is a search for a community. In this case the P2P network is accessed once.

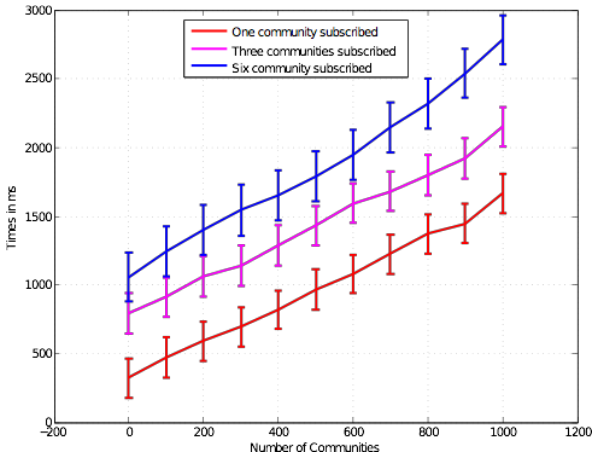

Figure 4. Cross-community EHR Update Time

Both results show that the time to find the information has logarithmic growth with the number of peers in the network, which also corresponds to the theoretical complexity of the P2P network (see Section 3.4). These results are encouraging as, in the current state of the art, data exchange may take days of human intervention. In the second test we varied the number of subscriptions for a patient from 1 to 6 and measured the average time for a community to send updates to the subscribed communities.

The Fig. 4 shows how the update time grows linearly with a growing number of subscriptions to a patient. This time depends on the number of instances held in the KB, which influences the time to search the subscriptions to a patient. Also, a growing number of subscriptions per patient introduces a latency as multiple update messages have to be sent.

7. RELATED WORK

The use of semantic representations for enabling interoperability between hospitals is not a new idea [1, 9], nor it is new the idea to use the publish and subscribe pattern to model the dissemination of events in healthcare [27], but, to the best of our knowledge, the use of an agent-based coordination infrastructure to govern the semantic interoperability between distributed nodes representing communities is new. In particular, the epSOS project[7] provides cross-border health-services to patients seeking healthcare in different countries, by defining an integration broker for cross border exchange of patient's health records. In epSOS there is no mechanism to handle the subscription of new communities, thereby the responsibility to connect different providers falls into the epSOS operator. On the contrary, we propose the use of coordination primitives and agent technology to dynamically connect communities and have a flexible approach towards subscription and notification of relevant events for a community. TheMediCoordination Healthcare Infrastructure (MHI) [1] aims at sharing medical data between medical actors. MHI's model consists of a registry/repository and two clients, one for submitting documents and one for receiving them. One XDS-based server is used for the repository and the registry. The MHI does not implement notifications [1]. General practitioners have to manually query the data in the registry. With respect to MHI, we propose a decentralized solution that can handle multiple communities, whereas MHI is limited to a centralized repository. Furthermore, the use of the Description Logic formalism, allows us a richer description of the events happening between different actors and across communities, and, to represent subscription and notification to complex events.

Triple space computing (TSC) applied to healthcare [24] is the approach that it is closer to ours. Also TSC uses tuple spaces to foster the exchange of information and proposes the use of semantic web technology to represent the data about the patients by associating RDF tuples to concepts defined in HL7 or SNOMED. In contrast with TSC, we are not concerned with translating HL7 concepts into a semantic web language but we deal only with the metadata associated to medical documents. From the perspective of the computation, also TSC considers the problem of publication and retrieval of health information, but it does not describe the notification and dispatching of the events happening in the distributed system, nor there is a clear representation of the concept of community. Finally, by using tuple centres and ReSpecT, we can modify the behavior of our communities, including new reactions at runtime, while this is not the case for TSC.

[7] http://www.epsos.eu

Works towards systems that actively notify changes of the patients' EHRs were presented in [23, 28, 10]. These solutions propose event notification systems that are based on the publish-subscribe paradigm [11]. The work presented in [23] addresses the exchange of patient records with a health care information system. Every patient can decide if to publish information to the subscribed medical doctors that are interested in his health records. The model does not provide the subscribers with a way to express which documents are of interest. Work towards notifying health records more selectively has been also presented in [28]. This work is focused in the home-care context, where new events are sent to the right caregivers. Three types of rules are used to control how data are released: *subscription authorization* defines which caregivers may subscribe to patient events; *event restrictions* define the events visibility to a subscribed caregiver; finally, *event transformation* alters an event instance to satisfy the information requirements of the caregiver. Despite the fine grained data control layer, the caregivers can only specify the type of events they are interested to receive, but there is no complex event processing involved. A more structured event notification mechanism is proposed in [10] where type based filtering is applied to medical data to create hierarchies of concepts for event notification purposes. Similarly to our work, the hierarchies of concepts are defined as a combination of *and* and *or* logic operators. However, this work does not directly address how the caregivers may join or leave the publish-subscribe network and the matching of the events is based on syntactic reasoning. Thus, the specified filters must reflect the exact used terminology used by the different communities.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented semantic agent coordination model that enables dynamic EHR exchange across communities. We have shown how the combination of a semantic publish and subscribe model with coordination languages such as ReSpecT can improve the current state of the art with respect to cross-community EHR exchange. The presented coordination model extends the IHE limitations by specifying a P2P network and a set of coordination primitives that enable communities to automatically update and search data in other communities without requiring a prior integration among them.

As part of our future work, we plan to further extend the proposed publish and subscribe model with a more expressive subscription language that also includes temporal expressions and reasoning. Additional work will focus on extending the set of the community policies in order to detect that the emerging behavior of the actors performing the queries is that expected within the community sub-system. We will further investigate how to log the access to the data in a distributed setting in such a way that it is possible to track back all the access to documents.

9. ACKNOWLEDGEMENTS

This work was partially funded by the Switzerland FNS grant nr. 200021 135386 / 1, the FP7 287841 COMMODITY12 project, the Hasler Foundation project nr. 11115-KG and by the SER project nr. C08.0114 within the COST Action IC0801 Agreement Technologies.

10. REFERENCES

[1] Alves, B., Muller, H., Schumacher, M., Godel,D. and Abu Khaled, O. Interoperability prototype between hospitals and General practitioners in Switzerland. *Stud Health Technol Inform*, 160:366{370, 2010.

[2] Androutsellis-Theotokis, S. and Spinellis, D. A survey of Peer-to-Peer content distribution technologies. *ACM Comput. Surv*., 36(4):335-371, December 2004.

[3] Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. Secure routing for structured peer-to-peer overlay networks. SIGOPS Oper. Syst. Rev., 36(SI):299-314, December 2002.

[4] Cerizza, D., Della Valle, E., Foxvog, D., Krummenacher, R., and Murth, M. Towards European Patient Summaries based on Triple Space Computing. In European Conference on eHealth, volume 91, pages 143-154, 2006.

[5] Colombetti, M.. Dispense corso di Ingegneria della conoscenza: modelli semantici, Politecnico di Milano. http://home.dei.polimi.it//colombet/IC/materiale/IC2011.

[6] Cremonini, M., Omicini,A., and Zambonelli, F.. Multi-agent Systems on the Internet: Extending the scope of coordination towards Security and Topology. In Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: MultiAgent System Engineering, pages 77-88, London, UK, 1999. Springer-Verlag.

[7] Kalra, D. Electronic health record standards. In Methods Inf Med, volume 45, pages 136{144, 2006.

[8] Denti, E., Omicini, A., and Ricci, A. Multi-paradigm Java-Prolog integration in tuProlog. Sci. Comput. Program., 57(2):217-250, 2005.

[9] Dogac, A., Laleci, G., Kirbas, S., Kabak, Y., Sinir, S., Yildiz, A., and Gurcan, Y. Artemis: deploying semantically enriched web services in the healthcare domain. Inf. Syst., 31:321{339, 2006.

[10] Esposito, C., Ciampi, M., De Pietro, G., and Donzelli, P. Notifying medical data in health information systems. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS '12, pages 373{374, New York, NY, USA, 2012. ACM.

[11] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. ACM Comput. Surv., 35(2):114{131, June 2003.

[12] Geissbuhler, A., Spahni, S., Assimacopoulos, A., Raetzo, M., and Gobet, G. Design of a patient-centered, multi-institutional healthcare information network using Peer-to-Peer communication in a highly distributed architecture. Medinfo, 11(Pt 2):1048{52, 2004.

[13] Gelernter. D. Generative communication in Linda. *ACM Trans. Program. Lang. Syst*., 7:80{112, 1985.

[14] Grimson, J., Grimson, W., and Hasselbring, W. The SI challenge in health care. *Commun. ACM*, 43:48-55, 2000.

[15] Gunter, T., and Terry, N. The emergence of national electronic health record architectures in the United States and Australia: Models, costs, and questions. *J Med Internet Res,* 7(1):e3, Mar 2005.

[16] Hendler, J. Agents and the Semantic Web. *IEEE Intelligent Systems,* 16:30-37, 2001.

[17] PHitzler, P., Krotzsch, M., and Rudolph, S. Foundations of Semantic Web Technologies. *Chapman & Hall/CRC*, 2009.

[18] Horrocks, I., Patel-Schneider, P., and van Harmelen, F. From SHIQ and RDF to OWL: the making of a Web Ontology Language. Web Semantics: *Science, Services and Agents on the World Wide Web*, 1(1):7 { 26, 2003.

[19] IHE. White paper: Cross community information exchange, 2008. www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_White_Paper_Cross_Community_2008-11-07.pdf.

[20] IHE. Technical framework integration profiles vol 1, 2011. www.ihe.net

[21] Maymounkov, P., and Mazieres, D. Kademlia: A Peer-to-Peer information System Based on the XOR Metric. *In Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53-65. Springer-Verlag, 2002.

[22] Nardini, E., Viroli, M., and Panzavolta E. Coordination in open and dynamic environments with TuCSoN semantic tuple centres. *In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC)*, pages 2037{2044. ACM, 2010.

[23] Neumann, C., Rampp, F., Lenz, R., and Daum M. A mediated publish-subscribe system for inter-institutional process support in healthcare. *In Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS '09*, pages 14:1-14:4, New York, NY, USA, 2009. ACM.

[24] Nixon, L., Cerizza, D., Della Valle, E., Paslaru Bontas Simperl, E., and Krummenacher, R. Enabling Collaborative eHealth through Triplespace Computing. *In WETICE*, pages 80-85. IEEE Computer Society, 2007.

[25] Omicini. A. Formal ReSpecT in the A&A Perspective. Electron. *Notes Theor. Comput. Sci.*, 175:97{117, 2007.

[26] Omicini, A., and Denti, E. From Tuple Spaces to Tuple Centres. *Science of Computer Programming*, 41(3):277-294, nov 2001.

[27] Singh, J., Bacon, J, Weerasinghe, D., Akan, O., Bellavista, P., Cao, J., Dressler, F., Ferrari, D., Gerla, M., Kobayashi, H., Palazzo, S., Sahni, S., Shen, X., Stan, M., Xiaohua, A. Zomaya, and Coulson, G. Event-Based Data Dissemination Control in Healthcare, volume 1, pages 167-174. Springer Berlin Heidelberg, 2009.

[28] Singh, J., and Bacon, J. Event-based data control in healthcare. *In Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion, Companion '08,* pages 84-86, New York, NY, USA, 2008. ACM.

[29] IHE Technical Framework. Document Metadata Subscription, 2013. www.ihe.net/Technical_Framework/upload/IHE_ITI_Suppl_DSUB_Rev2-0_PC_2013-06-03.pdf.

[30] Urovi, V., Olivieri, A., Bromuri, S., Fornara, N., and Schumacher, M. A Peer to Peer Agent Coordination Framework for IHE based Cross-Community Health Record Exchange. *InProceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1355-1362, 978-1-4503-1656-9, Coimbra, Portugal, 2013.

[31] Urovi, V., Olivieri, A., Bromuri, Brugués, A., S., Fornara, N., and Schumacher, M. Secure P2P cross-community health record exchange in IHE compatible systems. International J*ournal on Artificial Intelligence Tools (IJAIT)*, to apear, 2013.

[32] Urovi, V., and Stathis, K. Playing with agent coordination patterns in mage. *In Proceedings of the 5th international conference on Coordination, organizations, institutions, and norms in agent systems*, COIN'09, pages 86-101, Berlin, Heidelberg, 2010. Springer-Verlag.

[33] Wozak, F., Ammenwerth, E., Hoerbst, A., Soegner, P., Mair, and R., Schabetsberger, Th. IHE based Interoperability – Benefits and Challenges. volume 136 of *Studies in Health Technology and Informatics*, pages 771-776. IOS Press, 2008

ABOUT THE AUTHORS:

Dr. Urovi is a Post-Doc researcher at the University of Applied Sciences Western Switzerland. She obtained her PhD degree from the University of London and her MSc and BsC degree from the University of Bologna. Her research interests lie in the fields of intelligence and distributed systems. Most of her work is based on Mulit-Agent Systems (MAS) and Communication Protocols. Her work is being applied in social computing applications such as e-Health, Service Oriented Computing, Ambient Intelligence and Computer games. She is a team member of the Applied Intelligent Systems Laboratory (AIS Lab).

Mr. Alex C. Olivieri is a PhD student in the University of Applied Sciences Western Switzerland. He is involved in the European project “Universal Integration of the Internet of Things through an IPv6-based Service Oriented Architecture enabling heterogeneous components interoperability (IoT6). He has a master degree in Software Engineering from the University of Bologna.

Dr. Stefano Bromuri is a senior research scientist at HES-SO, and he obtained his PhD at Royal Holloway university of London in October 2009, with a thesis on distributed agent environments titled Generalised Ontological Environments for Multiagent Systems (GOLEM). Dr Bromuri was directly involved with the EU FP6 ARGUGRID project where he was in charge of the implementation and testing of the infrastructure. In particular, Dr Bromuri expertise lays in distributed multi-agent systems with particular focus on the concept of agent environment. At HES-SO, he is working as a working package leader in the EU FP7 COMMODITY12 project, in which he also leads the technical development of the project.

Dr. Nicoletta Fornara is Senior Researcher and Lecturer at the Faculty of Communication Sciences at Università della Svizzera italiana, Lugano, Switzerland. She has authored numerous journal and international conference papers and some chapter of books in the research area of multi-agent systems, in particular on agent communication languages, artificial institutions, normative systems and agent’s environment, and in the area of Semantic Web Technologies. For a complete list of publications visit http://www.people.usi.ch/fornaran/publications.html. She is principal investigator of a project connected to the COST Action "WEBDATANET" in which she represents Switzerland.

Prof. Michael Ignaz Schumacher is a full Professor in the Institute of Information Systems at the University of Applied Sciences and Arts Western Switzerland (HES-SO) since 2007. Previously, he was senior researcher at Ecole Polytechnique Fédérale de Lausanne (EPFL) and project manager in an international Swiss foundation committed in social investment. He was also visiting researcher at the Robotics Institute in Carnegie Mellon University in Pittsburgh (USA). He holds a PhD in computer science from the University of Fribourg in Switzerland. In 2009, he founded the Applied Intelligent Systems Lab (AISLab.hevs.ch), which focuses on distributed information systems and artificial intelligence applied to healthcare.