Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres

Elena Nardini Andrea Omicini Mirko Viroli
Alma Mater Studiorum – Università di Bologna, Cesena, Italy
{elena.nardini, andrea.omicini, mirko.viroli}@unibo.it

Michael I. Schumacher
University of Applied Sciences Western Switzerland (HES-SO), Sierre & Fribourg, Switzerland
michael.schumacher@hevs.ch

ABSTRACT

Open and distributed application scenarios like e-Health systems mandate for new coordination models and technologies. In particular, they require middleware providing coordination and security services modelled with abstractions promoting run-time observability and adaptation.

Along this line, in this paper we describe the architecture of the TuCSoN coordination infrastructure, and show its application to an e-Health application scenario.

1. INTRODUCTION

Healthcare supported by software systems – in short, e-Health – is evolving quickly [7, 24]. Among the several e-Health research activities, research on Electronic Health Record (EHR) is particularly intensive [24, 43]. An EHR represents a set of medical information called fragments, which are stored in a digital format over different healthcare institutions. The introduction of EHR offers several benefits [26]: better patient safety, lower cost of health services, and improvements in healthcare audit and research. The main challenge in the EHR domain is to ensure interoperability among EHR fragments belonging to an environment that is *distributed* and *open*, and where the *security support* represents a fundamental requirement to protect the patient privacy [23]. Several efforts have been made in the EHR domain in order to cope with such requirements, in particular:

- Definition of standards for EHR-fragment format and communication such as Health Level Seven (HL7) [11] and Digital Imaging and Communications in Medicine (DICOM) [21].
- Definition of standards like openEHR [13] and CEN EN 13606 [15] promoting semantic approaches to face heterogeneity and dynamism of fragment formats.
- Definition of specifications like the IHE (Integrating the Healthcare Enterprise) specifications [20], to build EHR coordination middleware able to coordinate EHR-fragment providers and requesters.
- Definition of EHR coordination middleware based on the semantic tuple-space computing [5], such as the one promoted by Triple Space Computing (TSC) [1].

As shown by TSC, semantic tuple-space computing provides models for EHR coordination middleware which overcome the limits of the solutions proposed in the IHE specifications. In fact, unlike IHE, TSC provides a solution based on semantic techniques as suggested by openEHR and CEN EN 13606, and models for building applications that support more complex dynamics than the mere exchange of EHR fragments. On the other hand, like IHE, TSC provides coordination media that cannot be tailored to the specific application needs. Thus, any coordination law not directly supported by the model has typically to be charged upon coordinated entities, thus increasing their complexity, especially in open scenarios. Moreover, TSC cannot cope with the evolution of applications over time, since it makes it difficult to adapt the coordination middleware in case of changes to application requirements.

Along this line, in this paper we aim at providing models and approaches extending the solutions proposed in IHE and TSC, in order to augment their effectiveness in building EHR services, in particular as far as interoperability is concerned. To this end, we draw our inspiration from TuCSoN [33], an open-source coordination infrastructure based on *semantic tuple-centres* [27], and promoting *online engineering* [35] through a middleware adaptable at runtime, thus coping with the dynamism requirements.

In particular, this article is organised as follows. In Section 2 we briefly survey the existing approaches supporting interoperability among EHR fragments, and discuss their benefits and drawbacks. Then, in Section 3 we present an overview of the TuCSoN architecture. In particular, we describe the key-features of the architecture: *(i)* the *behaviour programmability* and the *semantic support* of *tuple centres*—the coordination abstraction exploited by TuCSoN; *(ii)* the *organisation* and *RBAC* models, used respectively to model the system structures and their relationships, and the security aspects; and *(iii)* the *online engineering* approach, used to support the corrective/adaptive/evolutive maintenance of software systems. In Section 4, we show how to exploit the key-features of the TuCSoN architecture in order to extend the solutions proposed in IHE and TSC. Finally, Section 5 concludes, providing final remarks.

2. EHR SYSTEMS INTEROPERABILITY

2.1 Towards electronic health records

The healthcare domain has been evolving quickly over the past decades. Nevertheless, advances are somewhat limited in several domain [3] and obstacles are still to be overcome [10, 36]. Most of the administrative processes have adopted an e-Health solution so as to become computerised. However, in some hospitals and for the large majority of *General Practitioners* (GPs), medical data is still acquired and exchanged on paper. The totally paperless hospital has yet a way to go [38]. Besides digital storage, the computerised acquisition of medical data also makes data accessible for computerised decision support [42] and has the potential to reduce the large number of adverse events, particularly in hospitals [7, 24]. Alongside local advantages, communication of health data is another important factor in computerised data acquisition to overcome limits of paper-based information exchange, which is often slow [41] and error prone [24]. e-Health information exchange strategies and solutions have been developed on a local regional or cross-institutional [25] and national [34, 14] level and some already have concrete implementation in research projects [41].

Among the several e-Health research activities concerning the health information exchange, research on *Electronic Health Record* (EHR) is particularly intensive [24, 43]. A Patient EHR-document refers to the medical record of a patient stored in a digital format. The information stored in an EHR might include patient information such as demographics, medical history, medication, allergy list, lab results or radiology. Medical data belonging to an EHR are called *fragments*, and can be distributed over different EHR systems. The introduction of EHR offers several benefits [26]:

Better patient safety — Storing and transferring patient information electronically allows reducing clinical errors caused for example by illegible handwriting, documents or images, as well as it allows clinicians to communicate more quickly and accurately and to identify relevant information more easily.

Lower cost of health services — EHR technology can reduce administrative work to manage medical data since it can increase medical-data search efficiency and reduce medical-data duplication and waste.

Better audit and research — Behind improving medical assistance of patients, EHR technologies are also useful for other purposes. In particular, electronic databases of health information can be exploited for healthcare audit and research.

In order to keep the EHR benefits, EHR systems should ensure interoperability among EHR fragments. *Interoperability* – that is, the ability for two or more heterogeneous systems to communicate together – is of paramount importance in health information communication [12]. In case of EHR systems, interoperability should satify the following conditions [23]:

Distribution — EHR fragments should be easy to share even if the information is widespread across multiple EHR systems.

Openness — EHR supporting servers at different caregivers could be heterogeneous and change dynamically.

Security — It is necessary to support security mechanisms in order to avoid failures that can cause injury to the patient and violations to privacy.

Accordingly, interoperability among EHR systems call for specialised middleware able to deal with distribution, openness and security requirements in a coherent and transparent way. In the next section, we discuss standards and solutions from the literature, which propose design principles for middleware of such a sort.

2.2 Existing approaches: a survey

In order to cope with distribution, openness and security, the first approach to the issue of interoperability is the definition of standards for EHR-fragment format and communication. The two most representative are [23]:

- *Health Level Seven* (HL7): a set of open standards for the exchange, management and integration of EHR fragments [11]. In particular, HL7 provides *Clinical Document Architecture* (CDA) – a standard for the representation and machine processing of clinical documents – and *Messaging* standard—a standard covering EHR-fragment messaging aspects.

- *Digital Imaging and Communications in Medicine* (DICOM): a standard for handling and transmitting information in medical imaging. It includes a file format definition and a network communication protocol [21].

Standards such as HL7 and DICOM are not enough to achieve interoperable health systems. In fact, the result is that EHR systems use different set of format and communication standards, often incompatible, incomplete or involving overlapping scopes, thus breaking the interoperability requirement [18]. As a response to these problems – and as a complementary step towards the requirements of interoperability among EHR fragments – the following standards and initiatives were proposed:

- *openEHR* [13] and *CEN EN 13606* [15]: standards aiming at facing interoperability among EHR fragments. In particular they propose semantic approaches based on *Archetype Definition Language* (ADL) [4] – a formal language for expressing application-domain concepts – in order to describe semantically EHR fragments. By exploiting such kind of semantic techniques it may be possible to support interoperability among EHR fragments with different syntactic structures depending on the adopted standard.

- *Integrating the Healthcare Enterprise* (IHE) [20]: a non-profit initiative founded in 1998 led by professionals of the e-Health industry. The initiative goal is not to develop standards as such, but to select and recommend an appropriate usage of existing standards (e.g. HL7 and DICOM) in order to improve the sharing of information among EHR systems.

In this context, openEHR, CEN EN 13606, and IHE emphasise two important requirements. The first is to describe semantically EHR fragments in order to face heterogeneity and dynamism of fragment formats. The second is to provide a coordination middleware able to coordinate EHR systems and actors interacting with such systems, hiding distributed-fragment management and security issues from

entities to be coordinated. In particular, XDS, ATNA and XUA are central profiles for building a coordination middleware that connects EHR systems.

A further important contribute in this context is represented by *Triple Space Computing* (TSC) [5], which provides a different solution based on the *Linda tuple space* model [17]. Through the tuple space model, coordination among system entities occurs by exchanging information in form of tuples, in a common shared space called tuple space. System entities to be coordinated communicate with one another through *out*, *rd* and *in* primitives to respectively put, read and consume associatively tuples to/from the shared space. In particular, TSC shows the following interesting features:

- It provides a general coordination model to manage all kinds of interactions among system entities.
- Tuple space model is based on *generative communication* [17]: tuples generated by a tuple producer have an independent existence in the tuple space leading to *time*, *space* and *name uncoupling*. Uncoupling is a requirement to satisfy in order to cope with openness. Entities belonging to an open environment can be heterogeneous and can be added, removed or modified at runtime. For this reason, an entity cannot make a-priori assumptions about other system entities.
- It is based on *semantic tuple-space computing* [28]: it adopts tuple spaces enriched semantically thus allowing an exchange of data (tuples) semantically described by means of an ontology.
- Like IHE, it provides a Web-service interface to tuple spaces which promotes interoperability.

TSC puts together the advantages of using openEHR, CEN EN 13606 and IHE. Moreover, it improves the IHE approach by proposing a more general coordination model suitable for open scenarios, and not only specialised on the storage and retrieval of EHR fragments. In particular, through TSC it is possible to exploit a unique coordination model – the tuple space model – to manage all the system interactions. This is useful in case it is needed to extend e-Health systems with coordination functionalities concerning different kinds of interactions. For example, interactions with patients, with scientific research systems or with systems providing consumers with access to medical developments and research.

However, TSC exhibits some limits, too. The first one derives from the Linda tuple-space model: the tuple space behaviour is set once and for all by the model and cannot be tailored to the specific application needs [31]. Thus, any coordination law not directly supported by the model has typically to be charged upon coordinated components, thus obstructing the way to open systems. A further feature that should be supported by an EHR coordination-middleware – and that it is not covered by either IHE or TSC – is the ability to change its configuration at runtime in order to cope with application dynamism. In fact, during the lifetime of an application, requirements could be changed, added or removed. For example, new nodes could be added/removed to/from the network, or, coordination algorithms could be changed in order to improve the efficiency and effectiveness of the overall application. If the middleware does not allow for runtime changes, it might be necessary to shut it down in order to update its configuration—which is definitely undesirable, especially in application scenarios like e-Health that require continuous service availability.

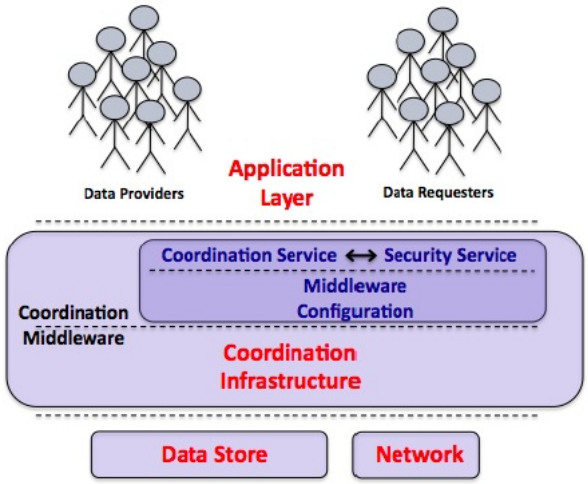

Figure 1: Logical levels for a coordination middleware

As far as middleware for the health domain is concerned, the current state of the art can be summarised as follows:

- IHE profiles, in particular XDS profile, do not provide a middleware model expressive enough to manage interactions among EHR actors. In particular, XDS provides a coordination middleware model not based on semantic techniques, and focused on coordinating meta-data in order to store and retrieve EHR fragments. As a consequence, it cannot be used for e-health applications going beyond the mere fragment coordination.
- TSC provides a solution that overcomes part of the XDS profile drawbacks. In particular, it exploits the Linda tuple-space model enriched with semantic techniques that exhibits features particularly useful for the realisation of an EHR middleware. On the other hand, TSC has some limits due to the fixed behaviour of its coordination abstractions, and to its inability to cope with middleware evolution over time.

Accordingly, an EHR middleware should be a coordination middleware supporting interactions among heterogeneous EHR-fragment providers and requesters—as in Figure 1. The middleware should be developed upon a coordination infrastructure giving the support for *distribution* of heterogeneous EHR-fragment-stores and providers/requesters of EHR fragments in a transparent way. Then, the infrastructure should provide the API required to build a *coordination* and a *security service*. The coordination service has the task to enable and rule interactions among system actors. It should exploit a general-purpose coordination model based on the tuple-space model and on semantic techniques, in order to cope with the openness requirement. In turn, the security service should be able to guarantee privacy of patient EHR-fragments, taking into account the federated nature of the healthcare system. Finally, the coordination infrastructure

should be based on *engineering approaches* making it possible to build a coordination middleware whose configuration is adaptable at runtime, in order to maintain a continuously-available EHR-service in front of dynamic changes of the application requirements.

3. THE TuCSoN ARCHITECTURE

TuCSoN (Tuple Centres Spread over the Network) [33] is a coordination model & infrastructure that manages the interaction space of a distributed software system by means of *tuple centres* [31]. Tuple centres are tuple spaces [16] enhanced with the ability to define their own behaviour in response to interaction events, according to specific application needs. In the same way as tuple spaces, tuple centres are information spaces structured as sets of tuples, i.e., structured and ordered chunks of information data; system components can interact with and through tuple centres by inserting, reading, and consuming tuples—so, the tuple centre model provides the same benefits of the generative communication as tuple spaces [16], which leads to system components uncoupling in space, time, and components' name. From the

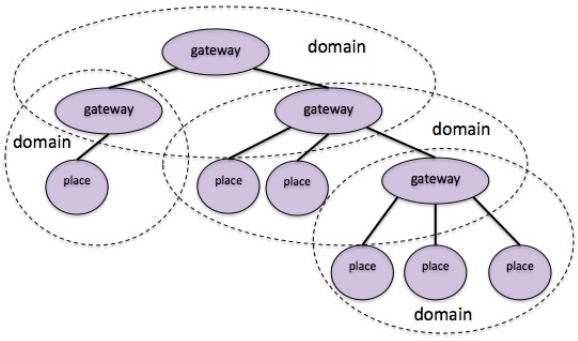

Figure 2: Topology of the TuCSoN coordination space

topology point of view, tuple centres are distributed and hosted in TuCSoN *nodes*, defining the TuCSoN coordination space [6]. In particular, the topological model of TuCSoN classifies nodes as *places* and *gateways*—as shown in Figure 2. The former represent the nodes hosting tuple centres used for specific applications/systems need, from supporting coordination activities to hosting information or simply enabling software components communication. The latter provide instead information about a the set of places belonging to a single domain—thus avoiding a single and centralised repository, which is unfeasible in complex and large environments. A *domain* is the set of nodes composed by the gateway and the places for which it provides information. A place can be part of different domains and a gateway can be a place in its turn. The overall picture of the TuCSoN topology is provided in Figure 2.

Besides topology, the key features of the TuCSoN architecture are: *(i)* the *behaviour programmability* and the *semantic support* in tuple centres, that represent the coordination model supported by TuCSoN—those aspects will be treated in Section 3.1; *(ii)* the use of the *organisation* and *RBAC* models: the former is exploited to describe the system structures and their relationships, the latter is exploited to model security aspects—both models will be treated in Section 3.2; and *(iii)* the *online engineering* approach used to support the corrective/adaptive/evolutive maintenance of software systems. Such an approach will be treated in Section 3.3.

3.1 Behaviour & semantics in tuple centres

The behaviour of the original tuple spaces – represented by their state transition in response to the invocation of the standard coordination primitives – is set once and for all by the model, and cannot be tailored to the specific application needs [16]. As a consequence, any coordination policy not directly supported by the standard behaviour of the coordination abstraction – the tuple space – has to be charged upon system coordinables, which hence grow in complexity—thus hampering the effectiveness of the coordination model especially in open scenarios [33]. Moreover, *associative access* [16] to tuples – tuples are retrieved by content and not by reference – in standard tuple spaces is based on a tuple matching mechanism which is purely syntactic. Although this might appear as a marginal aspect of tuple-based coordination models, it however imposes to coordinated components a design-time awareness of the structure and content of tuples: ultimately, components coordinated through a tuple space should be designed altogether—thus clearly working against the basic requirements for openness [40].

In order to overcome the above limits, the semantic tuple centre model adopts respectively *behaviour programmability* and *semantic support*. Through the *behaviour programmability* it is possible to program the tuple centre behaviour so as to embed coordination policies within the coordination media, without charging upon system coordinables. Whereas, through the *semantic support* it possible to exploit *ontology languages* to semantically describe information in tuple centres. Thus, the associative access provided by tuple centres could exploit a semantic content description without requiring a design-time awareness of the structure and content of tuples, but only of the ontology describing the application domain.

3.1.1 Behaviour programmability

The tuple centres behaviour can be determined through a *behaviour specification*, defining how a tuple centre should react to incoming/outgoing coordination events [31]. The behaviour specification can be expressed in terms of a *reaction specification language* that associates any communication event possibly occurring in the tuple centre, to a set of computational activities called *reactions*. Each reaction can access and modify the current tuple centre state by adding or removing tuples and access all the information related to the triggering communication event such as the performing software component, the operation required and the tuple involved. So, differently from tuple spaces, tuple centres represents *general-purpose* and *customisable* coordination media that can be programmed with reactions tailored to the application needs.

TuCSoN exploits ReSpecT tuple centres [31, 30]. ReSpecT adopts a tuple language based on first-order logic, where a tuple is a logic fact, any unitary clause is an admissible tuple template, and unification is the tuple matching mechanism. ReSpecT reactions are defined through logic tuples, too. A *specification tuple* is of kind `reaction(E,G,R)`. It associates a communication event described through `E`, to the reaction `R`. `G` represents a set of conditions that has to be satisfied in order to execute a reaction `R`, if the incoming event matches `E`. A reaction is defined as a transactional sequence of *re-*

action goals, which may access properties of the occurred communication event, perform simple term operations, and manipulate tuples in the tuple centre.

ReSpecT tuple centres provides two main advantages. They are *logic* tuple centres, thus making it possible to spread intelligence through the system where needed, for example by exploiting cognitive agents [45]. Also, ReSpecT is Turing-equivalent [8], so any computable coordination law can be encapsulated into the coordination medium. In [30], a complete example of use of reaction specification is discussed.

3.1.2 Semantic support

The tuple centre model was extended in [27] in order to provide coordination media with a semantic support, making it possible to perform semantic reasoning over tuples—namely, the ability of matching tuples with respect to a template not only syntactically as usual, but also semantically. From a semantic viewpoint, a tuple space has a simple and natural interpretation as a knowledge repository structured as a set of tuples. Tuples may be seen as representing objects – called *individuals* – of the application domain, whose meaning could be described by an *ontology*—that is, in terms of *concepts* and *relations* among them.

In particular, according to [27], the key features making a semantic tuple centre are: *domain ontology*, *semantic tuples*, *semantic tuple templates*, *semantic reactions*, *semantic matching mechanism* and *semantic primitives*. Those components were formally defined in [27] through *SHOIN(D)* [19]—a *Description Logic* (DL) language [2] representing the counterpart of *OWL DL*, that is one of the three species of the W3C standard *OWL*.

Domain ontology.

An ontology describing domain concepts and relations has to be attached to a tuple centre, so as to interpret the semantic associated to the knowledge (set of tuples) stored in a tuple centre. The ontology language exploited for ReSpecT tuple centres is OWL DL. The following OWL-DL code provides an example of description of the concept `Car` with the mandatory and functional relation `hasMaker` with the concept `Maker`—i.e., each car has precisely one maker:

```
<owl:Class rdf:ID="Car"/>
<owl:ObjectProperty rdf:ID="hasMaker">
    <rdf:type rdf:resource=
    "http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Car"/>
    <rdfs:range rdf:resource="#Maker"/>
</owl:ObjectProperty>
```

Semantic tuples.

A semantic tuple represents an individual described so that it can be semantically interpreted by means of the domain ontology associated to the tuple centre. In ReSpecT tuple centres semantic tuples are, from a syntactical viewpoint, logic tuples. In [27] a SHOIN(D)-like language was defined aimed at representing semantic logic tuples like

```
f550:'Car'(hasMaker:ferrari, hasMaxSpeed:285,
           hasColour in (red, black))
```

which defines an individual named `f550` that belongs to the concept `Car` and that has three relations: *(i)* `hasMaker` with the individual `ferrari`, *(ii)* `hasMaxSpeed` with the numeric datatype `165`, and *(iii)* `hasColour` with `red` and `black` that are datatypes of kind string.

Semantic tuple templates.

Semantic templates are to be used to flexibly retrieve semantic tuples, hence they consist in specifications of set of domain individuals described by the domain ontology. As for semantic tuple, in ReSpecT tuple centres semantic tuple templates are logic tuples. In [27] was also defined a SHOIN(D)-like language to represent semantic logic templates like

```
'Car'(exists hasMaker : ford)
```

which describes the set of individuals of kind `Car` in `hasMaker`-relation with the individual `ford`.

Semantic reactions.

As shown in Section 3.1.1, a ReSpecT *specification reaction* is a tuple of kind `reaction(E,G,R)`. In a *semantic specification reaction*, `E` describes the kind of primitive which it refers and a semantic tuple template. For example, `E` could be something like `out(Car)` describing every *out* of a semantic tuple representing an individual that has to belong to the concept `Car`. Otherwise, `E` could be something like `in(Car)` referring to every *in* with a semantic template describing a concepts that has to be sub-concept of `Car`.

Then, the guard `G` represents a set of conditions that has to be satisfied in order to execute a reaction `R`. Thus, in a semantic reaction specification it may contain conditions about the semantic tuples and templates described in `E`. In other words, `G` can contain concept descriptions that are to be respected by tuples or templates unifying in `E`.

Finally, since `R` can contain the use of the primitives *in*, *out* and *rd*, it can semantically interact with the tuple centre, as well as every system component.

Semantic matching mechanism.

In case of primitives, semantic matching simply amounts at checkking whether an individual, in form of a semantic tuple, is an instance of the concept described by a semantic template. Considering semantic reactions, in case of *out*, semantic matching amounts at checking whether the individual described through the primitive is an instance of the individual-set specification described `E` and `G`. Whereas, in case of *in* or *rd*, semantic matching means checking whether the concept described through the primitive is a sub-concept of the concept described `E` and `G`.

Like every DL-based system, SHOIN(D)-based systems provide a *reasoner* offering a set of reasoning services [2]. In order to implement the semantic matching mechanism for ReSpecT tuple centres the *subsumption checking* and *instance retrieval* services of the Pellet OWL-DL reasoner [39] were exploited. The former check if a concept `A` is a sub-concept of a concept `B`. Whereas, the latter service retrieve a set of individuals belonging to a given concept `A`.

Semantic primitives.

Semantic primitives (*in*, *out* and *rd*) represent the language whereby system components can read, consume and write knowledge described by means of a domain ontology. Since the knowledge stored in the semantic tuple centre must be always consistent with the domain ontology, in face of the primitive *out* it is required to check – by exploiting the DL reasoner – the consistency of the semantic tuple to be written in the tuple centre, with the domain ontology. This means that, differently from the original tuple centre semantic, the primitive *out* performed with a semantic tuple can

fail in case the related semantic tuple is not consistent with the domain ontology.

According to the above description, ReSpecT semantic tuple centres provide two main advantages. First, they exploit SHOIN(D)-like languages, in particular OWL-DL as their ontology language. OWL-DL represents a good compromise between expressiveness and complexity [19], and fits well the openness requirement, since it is a standard ontology language introduced for the Semantic Web, by the W3C Consortium. Moreover, while other approaches have explored how to use tuple-based models in semantic-aware contexts [40], the approach adopted in [27] for semantic tuple centres aims at smoothly extending the standard tuple centre setting so as to fully exploit the power of DL in coordination of system components, independently from the application context. In this way, all the benefits of using ReSpecT tuple centres (see Section 3.1.1) are maintained along with the semantic support.

3.2 Organisation & security

Jennings [22] refers to *organisation* as a tool helping software engineers managing complexity of software system development. Organisation allows the interrelationships between the various components of the system to be defined and managed, by specifying and enacting organisational relationships. In this way, engineers can group basic components into higher-level unit of analysis, and suitably describe the high-level relationship between the units themselves.

TuCSoN exploits an organisational model that is *role-based* [32]. Organisation and coordination are strictly related and interdependent issues. Organisation mainly concerns the structure and the structural relations of a system—i.e. the static issues of the agent interaction space. Coordination mainly concerns the processes inside a system – i.e. the dynamic issues of the agent interaction space –, often related to roles that usually frame agents position in the structure of the system organisation. Moreover, coordination is strictly related to *security*, being both focused on the government of interactions inside a system, however according to two different (dual) viewpoints: normative for security, constructive for coordination [6]. Whereas security focuses on preventing undesired/incorrect system behaviours – which may result in problems like denial of services or unauthorised access to resources – coordination is concerned with enabling desirable/correct system behaviours, typically the meaningful, goal-directed interaction between different system components. Due the relations among coordination, organisation and security, TuCSoN exploits an unique and coherent conceptual framework to manage the complexity of modelling such three dimensions [32].

The TuCSoN conceptual framework is represented by an extended version of the *Role-Based Access Control* (RBAC) model [37] – as shown in Figure 3 – called RBAC-MAS [44]. The model interprets an *organisation* as a set of *societies* composed by software components playing certain *roles* according to the *organisation rules*, where each role has an associated set of *policies*. Organisation rules define two types of relationships among roles: *(i) component-role relationship*, through which it is possible to specify whether a specific component is allowed (or forbidden) to assume and then activate a specific role inside the organisation; *(ii) role-role relationship*, through which it is possible to specify

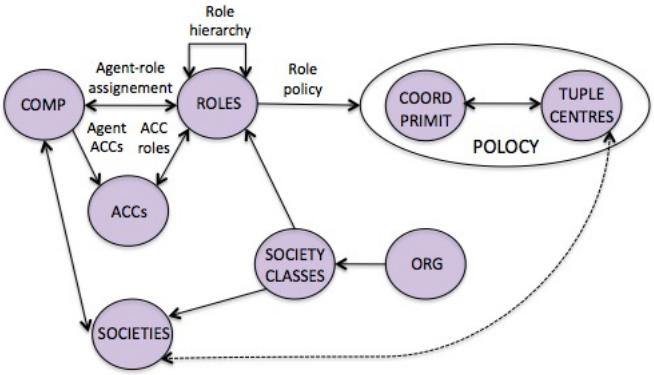

Figure 3: Logical levels in which the coordination middleware can be structured

structural dependencies among roles, so as to further define constraints on dynamic component role-activation.

A *policy* represents an admissible interaction protocol between the associated role and the rest of the organisation. An *ACC* (Agent Coordination Context [29]) represents an entity contracted by a component, on the basis of its identity, when it enters the organisation. The ACC is then released to components and used from components in order to interact with the resources (here, the tuple centres) belonging to a specific organisation. The interaction is enabled and ruled by the ACC in accordance with the rules and policies defined by the organisation.

From a topology point of view, an organisation is mapped onto a domain (including the linked domains or sub-domains). The description of the structures and rules characterising the organisation are stored and managed dynamically in a specific tuple centre, called `$ORG(OrgID)` – where `OrgID` is the organisation identifier –, hosted in a gateway node of the domain. The `$ORG` tuple centres host then information about societies, roles, components and the related relationships defined for the domain, represented by the gateway and its places.

3.3 Online engineering

The openness of software systems calls for keeping the abstractions *alive* [35]. Alive abstractions are defined in an explicit way in the meta-model of the system-engineering paradigm. Moreover, they are "kept alive" through the whole engineering process of a software system—from the analysis to the corrective/adaptive/evolutive maintenance phase. Such abstractions enable the inspection of their current state at runtime, so as to allow dynamic monitoring of system components that they model, and their creation and modification, so as to allow a dynamic evolution of system components. By exploiting such kind of abstractions, software engineers are enabled to perform *online engineering* [35], that is, the capability of supporting system design, development and test, debugging and evolution while the system is running.

Tuple centres are modelled and built as alive abstractions. Accordingly, TuCSoN allows the runtime maintenance of both coordination laws and organisation structure and rules. In particular, it is possible to maintain and evolve the coordination laws at runtime by inspecting and creating tuple centres, and by modifying their state or behaviour.

Then, it is possible to maintain and evolve the organisation model since the organisation structure and rules are reified as knowledge encapsulated in the tuple centre `$ORG`.

By means of its "alive abstractions", TuCSoN allows in principle both humans and (intelligent) software components to maintain and develop a software system. In order to support humans, TuCSoN provides the *Inspector* tool [9] enabling software engineers to first design and then observe and act on system structures and processes at runtime, working upon abstractions adopted and exploited for the design of a system. Besides, TuCSoN also provides intelligent agents with the API needed to create, inspect and modify tuple centres. In particular, since TuCSoN exploits ReSpecT tuple centres – which are *logic* tuple centres – it is possible to exploit agents capable of symbolic reasoning in order to autonomously maintain the structures.

4. EXPLOITING TuCSoN IN E-HEALTH

In the following, we show how the TuCSoN approach can be adopted in order to extend the solutions proposed by TSC and IHE (see Section 2): the overall goal is to increase the effectiveness of TSC and IHE approaches in coordinating EHR fragments. In particular, we discuss how such approaches can be integrated and extended with the key features of TuCSoN architecture, presented in Section 3. When dealing with IHE, we refer in particular to the following recommendations [20]: *Cross-Enterprise Document Sharing* (XDS) – i.e. profile describing an infrastructure for storing and registering medical documents –, *Audit Trial and Node Authentication* (ATNA) – i.e. profile describing security procedures – and *Cross-Enterprise User Assertion Profile* (XUA)—i.e. profile describing means to communicate claims about the identity of an authenticated principal (user, application, system,...) in operations that cross healthcare-enterprise boundaries.

TuCSoN *topology and XDS Affinity Domains.*

The e-Health environment is federated, that is, each healthcare enterprise belongs to a domain with other healthcare enterprises, using a common set of policies ruling interactions with and within a domain, and sharing common clinical documents. XDS calls each domain *Affinity Domain*. According to Section 3, the hierarchical topology of TuCSoN fits well with the sort of topology required by the EHR scenario. In particular, an Affinity Domain could be mapped in a TuCSoN domain whose gateway maintains the information about the policies and the structures associated to the domain itself. Then, each healthcare enterprise belongs to an Affinity Domain can be mapped in a TuCSoN place.

TuCSoN *semantic tuple centres as fragment coordination media.*

XDS provides a model to store and retrieve EHR fragments. Figure 4 shows the actor model defined by XDS. In particular the model is composed by:

Document Source | A healthcare point of service where clinical data is collected.

Document Consumer | A service application where care is given and information is requested.

Document Registry | A system storing *ebXML descriptions* of the clinical fragments to rapidly find them back.

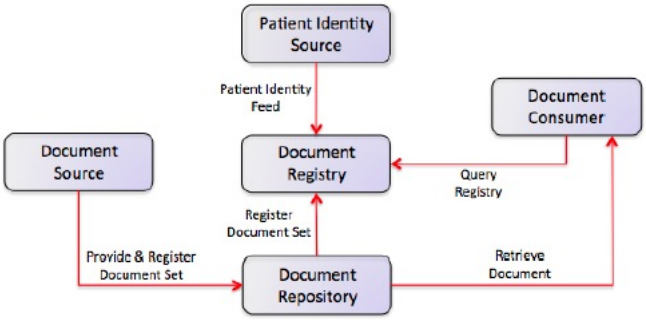

Figure 4: Actors for the IHE XDS profile

Document Repository | A system that stores documents and forwards the metadata to the document registry.

Patient Identity Source | A system that manage patients and identifiers for an Affinity Domain.

The XDS actor model has two main drawbacks. Document Registry is exploited to store and search metadata describing EHR fragments whereby it is possible to retrieve the related document from the Document Repository. In particular, XDS suggests to realise the registry through the *ebXML Registry* standards.

However, the main limit of an ebXML Registry is that it describes metadata in XML, and retrieves metadata in face of a query written in XML and SQL format. This kind of knowledge representation and retrieval lacks the expressive power provided by semantic approaches exploiting ontologies. In fact, unlike an ontology, an XML schema does not allow the description of complex taxonomies among concepts like those exploiting subsumption relationships. Also, XML tools does not perform powerful reasoning over metadata like semantic reasoning, which is able instead to infer new knowledge that is not declared in an explicit way. Thus, ontology-based approaches are more suitable for engineering knowledge in open context where the knowledge structure can evolve and where software components only have a partial awareness about the overall knowledge.

Another limit of the ebXML Registry is that it promotes a pre-defined behaviour only able to store and retrieve metadata. As a consequence, in order to extend the behaviour of the registry, a layer should be upon it that would enrich the operational semantic behind its interface in order to implement the new desired behaviour. This, of course, would definitely augment the complexity of the system. In order to cope with complexity, instead, it would be desirable to be able to define new behaviours directly in the registry, customising the registry with the policies associated to a particular healthcare domain. For example, a policy would allow the registry to be distributed over different nodes belonging to the domain, instead of having an unique registry per domain, as suggested by XDS. By exploiting behaviour programmability of the coordination media, it would be possible to coordinate a set of domain registries collaborating with one another in order to search and distribute metadata within the domain, in a smart way.

This is why the tuple centre model looks like a good candidate to build a Document Registry. On one hand, a semantic tuple centre supports the semantic representation of

the stored knowledge – like TSC –, but – unlike TSC – it also provides a tuple/template language that is independent from the technology exploited to implement the semantic support. Thus, each domain can choose to exploit a particular semantic technology guaranteeing interoperability with other domains. On the other hand, since the behaviour of a tuple centre is programmable, it is possible to tailor the registry to specific application needs. Moreover, by exploiting logic tuple centres like ReSpecT tuple centres, it is possible to promote cognitive processes by exploiting rational agents.

Exploiting TuCSoN *RBAC model.*

As shown in Section 3, TuCSoN provides the organisation abstraction to describe the structures and rules composing a system. In particular, an organisation in an Affinity Domain could be mapped in the TuCSoN `$ORG` tuple centre managing the domain structures, like Document Registries and the domain places where e-Health enterprises are hosted, and defining the set of roles that can interact with the organisation along with a set of related policies to rule such interactions.

In the context of Affinity Domains, a role represents a class of identities that can interact with EHR fragments, whereas policies represent the admissible interactions for a specific role. Accordingly, the RBAC-MAS model [44] can be suitable integrated with security recommendations defined in ATNA and XUA. Such recommendations in particular require: *(i)* an authentication service able to authenticate users, *(ii)* access control policies, *(iii)* a secure communication between system actors, and *(iv)* a security service supporting cross-authentication among EHR domains. In order to satisfy such requirements, TuCSoN can be integrated with two technologies suggested by IHE: *Kerberos* authentication service, and Web Services as interface to access to TuCSoN organisations, so as to promote the interoperability requirement.

Thus, Web Services can be used to access to TuCSoN organisation in secure way by exploiting *WS-Security*, that is, a secure communication protocol developed by the *OASIS-Open group*. In particular, *WS-Security* includes both *WS-SecureConversation* – which can be exploited to ensure secure conversations among system actors –, and *WS-Trust*— which can be exploited to support cross-authentication among EHR domains. Through *WS-Trust* it is possible to establish trust relations among domains that are exploitable to accept requests coming from different domains without having to authenticate users again. Finally, by integrating the authentication service Kerberos with TuCSoN, user identities can be associated to roles and policies in the `$ORG` tuple centre, and be authenticated.

Online engineering for continuos e-Health system interoperability.

As discussed in Section 3, TuCSoN exploits *alive* abstractions to model coordination, organisation and security, thus promoting their online engineering. By exploiting semantic tuple centres to model Document Registries and organisation to model the structures composing an Affinity Domain, TuCSoN makes it possible to support the runtime corrective/adaptive/evolutive maintenance of an e-Health fragment system—that is, with no need to stop the system. This is particularly useful whenever application requirements are expected to change substantially over time. For instance, it may happen that new places hosting e-Health enterprises need to be added by reconfiguring Affinity Domains dynamically, or that roles and policies have to be added/removed/modified to cope with dynamic organisation changes. This would require to change the behaviour of a Document Registry to face the new application requirements. Through online engineering as supported by the TuCSoN architecture, the system could be evolved in a consistent way at runtime, maintaining a continuos interoperability among EHR systems. We think this is a crucial aspect to be considered in the engineering of e-Health applications, where a continuos service availability is indeed fundamental—and this is why we promote the integration of IHE recommendations within the TuCSoN architecture.

5. CONCLUSIONS

The most important requirement to be satisfied in EHR fragment coordination is the interoperability among healthcare systems in a scenario in which: *(i)* EHR fragments could be distributed among different healthcare systems, *(ii)* healthcare systems can be heterogeneous and change dynamically, and *(iii)* security mechanisms play a fundamental role to ensure patient privacy and safety. Several efforts can be found in the literature trying to cope with such requirements—like HL7, DICOM, CEN EN 13606, openEHR, IHE and TSC.

However, such approaches exhibit two main shortcomings. First of all, they provide special-purpose models of coordination, which increase the complexity of building an EHR coordination middleware. This limits system interoperability by making it difficult to integrate independent e-Health systems. Moreover, they do not support any form of online engineering. As a consequence, the coordination middleware cannot be updated at runtime in order to cope with new application requirements without stopping the system.

Along this line, in this paper we proposed a coordination model and technology that could integrate the solutions and standards proposed in literature while addressing the aforementioned issues. In particular, we proposed TuCSoN as a reference architecture for coordination in the e-Health scenario since it provides a general-purpose model of coordination accounting for distribution and security issues in the engineering of EHR systems, and promotes online engineering for continuos service availability of e-Health applications.

6. REFERENCES

[1] Tripcom - triple space communication. Technical report.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] G. Barnett and H. Sukenik. Hospital Information Systems. In J. Dickson and J. Brown, editors, *Future Goals of Engineering in Biology and Medicine*. Academic Press, 1969.

[4] T. Beale. 2001.

[5] D. Cerizza, E. Della Valle, D. Foxvog, R. Krummenacher, and M. Murth. Towards European Patient Summaries based on Triple Space Computing. In *ECEH 2006*, 2006.

[6] M. Cremonini, A. Omicini, and F. Zambonelli. Multi-agent systems on the Internet: Extending the scope of coordination towards security and topology. In F. J. Garijo and M. Boman, editors, *Multi-Agent Systems Engineering*, volume 1647 of *LNAI*, pages 77–88. Springer, 1999.

[7] J. Denny, D. Giuse, and J. Jirjis. The Vanderbilt Experience with Electronic Health Records. *Seminars in Colon and Rectal Surgery*, 12:59–68, 2005.

[8] E. Denti, A. Natali, and A. Omicini. On the expressive power of a language for programming coordination media. In *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177, Atlanta, GA, USA, 27 Feb. – 1 Mar. 1998. ACM. Special Track on Coordination Models, Languages and Applications.

[9] E. Denti, A. Omicini, and A. Ricci. Coordination tools for MAS development and deployment. *Applied Artificial Intelligence*, 16(9/10):721–752, Oct./Dec. 2002.

[10] R. Dick and E. Steens. *The Computer-Based Patient Record: An Essential Technology for Health Care*. Institute of Medicine, National Academic Press, 1991.

[11] R. Dolin, L. Alschuler, S. Boyer, and C. Beebe. HL7 Clinical Document Architecture, Release 2.0. 2004.

[12] Y. Ducq, D. Chen, and B. Vallespir. Interoperability in enterprise modelling: Requirements and roadmap. *Advanced Engineering Informatics*, 18:193–203, 2004.

[13] L. eHealth Media. OpenEHR Foundation launches international standard.

[14] B. S. Elgera, J. Iavindrasana, L. Lo Iacono, H. Müller, N. Roduit, P. Summers, and J. Wright. Strategies for health data exchange for secondary, cross-institutional clinical research. *Computer Methods and Programs in Biomedicine*, 99:230–251, 2010.

[15] E. C. for Standardization. Health informatics Electronic health record communication Part 1: Reference model Draft European Standard for CEN Enquiry prEN 13606-1. 2004.

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

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

[18] W. Hasselbring and S. Pedersen. Metamodelling of Domain-Specific Standards for Semantic Interoperability. In J. Dickson and J. Brown, editors, *WM 2005*. Academic Press, 2005.

[19] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. 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, Dec. 2003.

[20] I. International. Integration Profiles: IHE IT Infrastructure Technical Framework. 2009.

[21] M. James and D. Thomas. Thomas The DICOM image formatting standard: What it means for echocardiographers. *Journal of the American Society of Echocardiography*, 8:319–327, 1995.

[22] N. R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001.

[23] D. Kalra. Electronic health record standards. *IMIA Yearbook of Medical Informatics*, pages 150–161, 2006.

[24] L. Khon, J. Corrigan, and M. Donaldson. *To Err is Human: building a safer health system*. National Academy Press, 2000.

[25] C. Lovis, S. Spahni, C. Cassoni, and A. Geissbuhler. Comprehensive management of the access to the electronic patient record: towards trans-institutional network. *International Journal of Medical Informatics*, 76:466–470, 2006.

[26] K. Malloch. The electronic health record: An essential tool for advancing patient safety. *Nursing Outlook*, 55:150–161, 2007.

[27] E. Nardini, M. Viroli, and E. Panzavolta. Coordination in open and dynamic environments with tucson semantic tuple centres. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, volume III, pages 2037–2044, Sierre, Switzerland, 22–26 Mar. 2010. ACM.

[28] L. j. b. Nixon, E. Simperl, R. Krummenacher, and F. Martin-recuerda. Tuplespace-based computing for the semantic web: A survey of the state-of-the-art. *Knowl. Eng. Rev.*, 23(2):181–212, 2008.

[29] A. Omicini. Towards a notion of agent coordination context. In D. C. Marinescu and C. Lee, editors, *Process Coordination and Ubiquitous Computing*, chapter 12, pages 187–200. CRC Press, Boca Raton, FL, USA, Oct. 2002.

[30] A. Omicini. Formal ReSpecT in the A&A perspective. *Electronic Notes in Theoretical Computer Sciences*, 175(2):97–117, June 2007.

[31] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[32] A. Omicini and A. Ricci. MAS organisation within a coordination infrastructure: Experiments in TuCSoN. In A. Omicini, P. Petta, and J. Pitt, editors, *Engineering Societies in the Agents World IV*, volume 3071 of *LNAI*, pages 200–217. Springer-Verlag, June 2004. 4th International Workshop (ESAW 2003), London, UK, 29–31 Oct. 2003. Revised Selected and Invited Papers.

[33] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999.

[34] J. M. Overhage, L. Evans, and J. Marchibroda. *Journal of the American Medical Informatics Association*, 12:107–112, 2005.

[35] A. Ricci and A. Omicini. Supporting coordination in open computational systems with TuCSoN. In *IEEE 12th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*, pages 365–370, 1st International Workshop "Theory and Practice of Open Computational Systems" (TAPOCS 2003), Linz, Austria, 9–11 June 2003. IEEE CS. Proceedings.

[36] C. Safran, D. Sands, and D. Rind. Online medical records: a decade of experience. *Methods Inf Med*, 38:308–312, 2000.

[37] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*,

29(2):38–47, Feb 1996.

[38] D. Sands, D. Rind, C. Vieira, and C. Safran. Can a large institution go paperless? In *MEDINFO*, 1998.

[39] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, 2007.

[40] R. Tolksdorf, L. J. B. Nixon, and E. P. B. Simperl. Towards a tuplespace-based middleware for the Semantic Web. *Web Intelligence and Agent Systems*, 6(3):235–251, 2008.

[41] W. van der Kam, P. W. Moormanb, and M. Koppejan-Mulder. Effects of electronic communication in general practice. *International Journal of Medical Informatics*, 60:59–70, 2000.

[42] J. Van der Lei, M. Musen, E. van der Does, A. Main in't Veld, and J. van Bemmel. Review of physician decision making using data from computer-stored medical records. *The Lancet*, 338:1504–1508, 1991.

[43] U. Varshney. *Pervasive Healthcare Computing*. Springer, 2009.

[44] M. Viroli, A. Omicini, and A. Ricci. Infrastructure for RBAC-MAS: An approach based on Agent Coordination Contexts. *Applied Artificial Intelligence*, 21(4–5):443–467, Apr. 2007. Special Issue: State of Applications in AI Research from AI*IA 2005.

[45] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, June 1995.