

# Agent-based Models, Technologies and Methodologies for the Engineering of Software Systems in the Context of Open Scenarios

Dottorato di Ricerca in Informatica, Elettronica e Telecomunicazioni  
XXIII Ciclo  
Relazione di Fine Secondo Anno

**Elena Nardini<sup>1</sup>**

<sup>1</sup>Alma Mater Studiorum – Università di Bologna  
[elena.nardini@unibo.it](mailto:elena.nardini@unibo.it)

Bologna - January 14, 2010



- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications



# Engineering Today's Software Systems I

- Today's software systems like pervasive systems, internet applications, and web services, are mainly characterised by two main features:
  - ▶ **Distribution** (of control, spatial, and temporal)
  - ▶ **Openness**
- **Agent-oriented** approach to software engineering provides the **Multi-agent System** (MAS) paradigm suitable for dealing with the engineering of such systems [Jennings, 2001].
- A MAS is a set of **autonomous**, **pro-active**, and **interacting** computational entities called **agents** situated in an **environment** where they interact typically producing a coherent global behaviour.



# Engineering Today's Software Systems II

- An open environment call for coordination models able to provide **uncoupling** among system components: agents and resources.
- Coordination in open environments leads current research trends in the area of coordination middleware to adopt **Linda-like tuple spaces** as basic coordination abstractions [Nixon et al., 2008].



# Linda Model

- Coordination medium: **Tuple Space**.
  - ▶ Multiset / bag of data object / structures called **tuples**.
- Communication Language:
  - ▶ **Tuple**: ordered collection of (possibly heterogeneous) information items (e.g. ("car", ka)).
  - ▶ **Templates**: specification of set / classes of tuples (e.g. ("car", ?x)).
  - ▶ **Tuple Matching Mechanism**: mechanism to match tuples and templates.
- Coordination Language: set of **operations out**, **in**, and **rd**, to respectively put, read, and consume associatively tuples to / from the space.



# Linda Benefits

- Linda it is based on **generative communication**: tuple generated by a tuple producer has an independent existence in the tuple centre [Rossi et al., 2001, Nixon et al., 2008].
  - Generative communication leads to **communication orthogonality**.
    - ⇒ **Space uncoupling** (distributed naming) among coordinables.
    - ⇒ **Time uncoupling** among coordinables.
  - In Linda the tuple space access is **associative**.
    - ⇒ **Data-reference uncoupling**.
- ⇒ Such features make the tuple space model interesting for the engineering of system coordination in distributed and open applications.



# Towards Tuple Centres

- The behaviour of Linda tuple space is set once and for all by the model, and cannot be tailored to the specific application needs [Omicini and Denti, 2001].
- ⇒ Any coordination law not directly supported by the model has typically to be charged upon system coordinables growing their complexity, especially in distributed, open and dynamic scenarios [Omicini and Zambonelli, 1999].
- Tuple centres extend the basic Linda tuple space model by making the behavior of the space programmable so as to encapsulate coordination laws directly in the coordination media.
- ⇒ Tuple centres can be conceived general-purpose customisable coordination media mediating interaction among all system components.



# Towards Semantic Tuple Centres I

- The main mechanism enabling coordination through a tuple centre, is the **tuple matching mechanism**:
  - ▶ coordination laws embedded in a tuple centre are enacted through the tuple matching mechanism, against the invocation of a communication primitive for inserting, reading, or consuming information items by means of tuples.
- Although the tuple centre model does not assume any matching mechanism [Omicini, 1999, Omicini and Denti, 2001], like the Linda implementation case, tuple centre implementations adopt a purely syntactic matching mechanism [Omicini and Denti, 2001].



# Towards Semantic Tuple Centres II

- As already observed for **service-oriented applications** in the context of Pervasive Computing [Bandara et al., 2008] and Web Services [Paolucci et al., 2002], syntactic matching mechanisms pose several limitations in dealing with open and dynamic scenarios where the information exchanged could have different syntactic structures.
  - In our context it may mean that a system component (agent or also a system resource) refers to the concept **Car** and in the shared spaces there is information about **SportCar** or **CityCar** concepts – that are both types of **Car** –, with the syntactic matching mechanism it is not possible to match those concepts.
- ⇒ It could become difficult or not possible to coordinate heterogeneous and dynamic system coordinables.



# Towards Semantic Tuple Centres III

- The advent of the Semantic Web has increased the interest in the use of semantic description of information through an **ontology language** and the use of **logical reasoning** over such descriptions to support information matching [Bandara et al., 2008, Paolucci et al., 2002].
  - An ontology language coupled with a logical reasoning offers two main advantages:
    - ▶ the ability to identify a match between logically equivalent information with syntactically different structures;
    - ▶ the ability to dynamically acquire, extend, and adapt the vocabulary used to describe the knowledge.
- ⇒ Such benefits are necessary within open scenarios where no assumptions can be made about the available knowledge.



# Towards Semantic Tuple Centres IV

- The introduction of a semantic matching mechanism in tuple centres becomes desirable.
- Through sharing domain ontologies it is possible to face openness problems concerning:
  - ▶ heterogeneous coordinables that refer to information that is semantically equivalent but has a different syntactic representation;
  - ▶ the inability of coordinables to have a prior knowledge about all possible information stored in a tuple centre;
  - ▶ the dynamic evolution of the application domain, by dynamically adapting the domain ontology.



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres**
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications

# Existing Approaches

- **Semantic tuple space computing** is aimed at augmenting tuple spaces with semantic techniques [Nixon et al., 2008].
- Introduced for coping dynamism in the context of Semantic Web and Semantic Web Services by means of the use of the tuple space coordination model, since it promotes decoupling among system coordinables.
- Proposes extensions of classical Linda-based systems with semantically enriched tuple spaces, designed specifically for the Semantic Web and Semantic Web Services.
- The proposed solutions are not well suited to deal with scenarios different from Semantic Web and Semantic Web Services:
  - ▶ provide abstractions depending from the chosen semantic technology;
  - ▶ suffer of the tuple space model limits.



# Our Aim

- We assume tuple centre as coordination model to manage system interactions,
- with the aim to semantically enrich such model in order to provide a general-purpose and customisable coordination media also able to cope requirements like dynamism,
- maintaining the model identity and without any assumption about the application context.



# Ontologies and Individuals in Tuple Centres I

- From an abstract viewpoint, a tuple centre can be seen as a knowledge repository structured in a set of tuples that encapsulates a specific portion of the system coordination laws.
- In a semantic view, such knowledge represents a set of objects occurring in the application domain, whose meaning is described by the domain concepts and the description of relations among such concepts, within an ontology.

⇒ The two first ingredients to design the semantic tuple centre model are:

- ▶ **domain ontology** allowing to interpreter the semantic associated to the knowledge (set of tuples) stored into a tuple centre;
- ▶ **domain objects** – represented by tuples – described so that they can be interpreted in a semantic way, by means of the domain ontology.



# Ontologies and Individuals in Tuple Centres II

- In order to formally define the domain ontology and objects in the context of a tuple centre, the literature makes available a family of knowledge representation formalisms called **Description Logic (DL)** [Baader et al., 2003].
- By adopting such formalism, then we can exploit a DL-based language, that is a logic – i.e., formal language with well defined semantics – allowing [Baader et al., 2003]:
  - ▶ ontologies, and information using vocabulary defined by ontologies, to be shared and exchanged without disputes as to precise meaning;
  - ▶ automated reasoning techniques over ontologies that can be exploited – for example by intelligent agents – to face the dynamic evolution of the vocabulary used to describe the knowledge.



# Ontologies and Individuals in Tuple Centres III

- In particular, we exploit the SHOIN(D) Description Logic formalism in order to model a semantic tuple centre in terms of domain ontologies and objects, since it represents:
  - ▶ a good compromise between expressiveness and complexity [Horrocks et al., 2003];
  - ▶ the theoretical counterpart of **OWL DL**, that is one of the three species of OWL [Horrocks et al., 2003]—the W3C standard ontology description language for Semantic Web. As standard OWL well fits the openness requirement.



# Domain Ontology Description I

- DL allows to describe the tuple centre ontology in the form of a **terminology** through a so called **TBox**.
- TBox consists of **concept descriptions**, which denote sets of objects called **individuals**, and **role** descriptions, which denote binary relationships between individuals.
- In order to express such descriptions, SHOIN(D) provides a set of constructors described in [Horrocks et al., 2003], where the main of them are:

| SHOIN(D)      | Meaning                    |
|---------------|----------------------------|
| $C \sqcap D$  | Intersection               |
| $C \sqcup D$  | Union                      |
| $\neg C$      | Negation                   |
| $\forall R.C$ | Value Restriction          |
| $\exists R.C$ | Existential Quantification |
| $\leq N$      | Number Restriction         |
| $\geq N$      | Number Restriction         |



# Domain Ontology Description II

- Moreover, TBox formalism provides the axioms  $\sqsubseteq$  (Inclusion) and  $\equiv$  (Equality), in order to define a taxonomy among concepts or roles.
- To give an explicative example of the TBox formalism use, we can think to describe the knowledge stored in a tuple centre as a set of individuals belonging to the *car* domain.
- We can define the following assertions:

$$Maker \sqsubseteq \top$$

$$Car \sqsubseteq (= 1 \text{ hasMaker})$$

$$(= 1 \text{ hasMaker}) \sqsubseteq Car$$

$$\top \sqsubseteq \forall \text{hasMaker}. Maker$$

$$CityCar \sqsubseteq Car$$



# Domain Individual Description I

- Having understood how to describe the domain ontology of a tuple centre, the next step is to understand how to describe the domain objects of a tuple centre in form of tuples, so to obtain **semantic tuples**.
- For this purpose, DL defines through a so called **ABox** the **assertions** about the individuals and roles, in terms of the terminology defined through the TBox.



# Domain Individual Description II

- ABox defines two kinds of assertions:

$$C(a) \quad R(b,c)$$

- Taking the example about the **car** domain, the ABox formalism [Baader et al., 2003] allows to assert individuals like:

$$\begin{aligned} &CityCar(ka) \\ &Maker(ford) \\ &hasMaker(ka, ford) \end{aligned}$$



# Domain Individual Description III

- In order to represent an individual in form of tuple through the ABox formalism, a language to express the following information is needed:
  - ▶ the name of the individual;
  - ▶ the concept to which the individual belongs;
  - ▶ the set of relations in which the individual is involved.
- Such information has to be described in accordance with the specific domain ontology.



# Interaction with a Semantic Tuple Centre

- Now that we have characterised the tuple centre knowledge in terms of domain ontology and individuals, it is important to understand how this influences:
  - ▶ **tuple templates**, that are specifications of sets of tuples;
  - ▶ **tuple matching mechanism** to match templates and tuples;
  - ▶ **tuple centre primitives** to read, consume, and write tuples in the tuple centre.



# Semantic Tuple Templates

- Tuple templates become specifications of set of domain individuals described by the domain ontology.
  - ⇒ A tuple template becomes a description in TBox formalism, of the set of individuals one is interested in.
  - ⇒ For example, if we are interested in the set of cars that have ford as a maker, the tuple template has to provide a concept description like  $Car \sqcap (\exists \text{ hasMaker.ford})$ .
- ⇒ As well as the tuple case, a language is required in order to express **semantic tuple templates** in accordance with such requirements.



# Semantic Tuple Matching Mechanism

- Tuple matching mechanism represents the mean whereby it is possible to identify tuples that match the specifications described by tuple templates.
- In a semantic tuple centre this means that the matching mechanism has to identify and retrieve the set of individuals specified by a TBox-based concept description, by means of the domain ontology.
- To this end, it is possible to exploit the reasoner service provided by each Description Logic-based system that allows to query the ABox through a TBox-based description and obtain the set of individuals that satisfy such query.
- In particular, it is possible to exploit the **instance checking** mechanisms provided by Description Logic reasoners whereby it is possible to verify if an individual is an instance of a concept [Baader et al., 2003].



# Semantic Tuple Centre Primitives

- In a semantic view, tuple centre primitives (in, rd, and out) represent the language whereby system components can read, consume, and write knowledge described by means of a domain ontology.
  - Each primitive can fail in case of both a syntactic error and not consistency with the TBox.
- ⇒ Differently from the original tuple centre semantic, the **out** can fail in case the related tuple is not consistent with the domain ontology.



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN**
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications

# TuCSoN Infrastructure

- Tuple Centre Spread over the Network [Omicini and Zambonelli, 1999].
- TuCSoN provides tuple centres distributed over the network, collected in **nodes**—internet nodes identified by the IP (logic) addresses.
- The topological model of TuCSoN distinguishes the nodes in **gateways** and **places**.
  - ▶ **Gateways** provide information for a limited set of places (a single and centralised repository is unfeasible in complex and large environments).
  - ▶ **Places** represent the nodes hosting tuple centres used for specific applications need.
  - ▶ A **domain** represents the set of nodes composed by the gateway and the places for which it provides information.

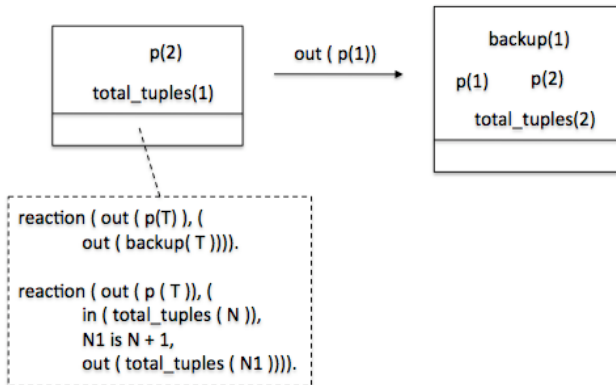


# ReSpecT Tuple Centres

- TuCSoN provides ReSpecT tuple centres:
  - ▶ tuple centres are identified by means of a first order logic term.
  - ▶ tuples and tuples templates are first order logic terms.
  - ▶ tuple matching mechanism is based on **logic unification**.
  - ▶ tuple centres are programmed through as set of **logic tuples** called **reactions** specifying medium behaviour reacting to interaction events.



# Simple Example



# Semantic Tuple Centres in TuCSoN

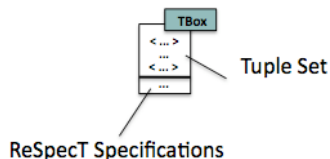
- To give a semantic to the information stored in ReSpecT tuple centres we can describe it through an ontology in terms of the concepts and relationships among them, belonging to the particular application domain.
- By adopting the [Description Logic](#) formalism we can define an ontology by means of a [TBox](#) and an [ABox](#).



# TBox for Semantic Tuple Centres

- Each ReSpecT tuple centre is associated to a specific TBox describing the stored information.
  - For the TBox definition in ReSpecT tuple centres a SHOIN(D)-like description language it is exploited: **OWL-DL** ontology description language [Horrocks et al., 2003].
    - ▶ OWL is the ontology standard description language for the Semantic Web.
- ⇒ OWL well fits the TuCSoN openness requirement.

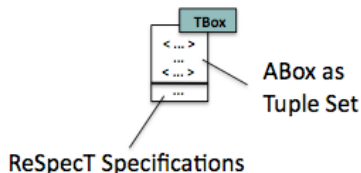
## ReSpecT Tuple Centre



# ABox for Semantic Tuple Centres I

- Each **tuple** stored in a tuple centre is described as an ABox individual specifying the following information:
  - ▶ **name of the individual** we want to describe;
  - ▶ **concept** to which the individual belongs;
  - ▶ **set of roles** in which the individual is involved.

## ReSpecT Tuple Centre



# ABox for Semantic Tuple Centres II

- A possible SHOIN(D)-like description language for semantic tuples is showed:

Individual ::= Iname : Descr

Descr ::= Cname | Cname (F)

F ::= Pname : V | Pname in ( Vset ) | F , F

Vset ::= V | V , Vset

V ::= Iname | N | S

Cname ::= <atom , in Prolog style >

Iname ::= <atom , in Prolog style >

Pname ::= <atom , in Prolog style >

N ::= <number >

S ::= <atom , in Prolog style >

- We can obtain the following semantic tuple:

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



# Semantic Tuple Templates I

- In order to describe a tuple template in a semantic way, we need a SHOIN(D)-like description language to express a tuple template as a description in the TBox formalism.
- A possible SHOIN(D)-like description language for semantic tuple templates is showed:

$$C ::= \text{all} \mid \text{none} \mid \text{Cname} \mid C, C \mid C ; C \mid \\ \text{not } C \mid R \mid \{ \text{Iset} \} \mid \text{Cname} ( R ) \mid \\ ( C )$$
$$R ::= F \mid \text{exists } F \mid \text{only } F \mid M$$
$$F ::= P \text{ in } C \mid P : \text{Iname} \mid P : D$$
$$M ::= \# \text{ Msymb PosInt} : P$$
$$D ::= \text{Msymb } N \mid = S$$
$$P ::= \text{Pname} \mid \text{Pname} / \text{Binding}$$
$$\text{Msymb} ::= > \mid < \mid >= \mid <= \mid =$$
$$\text{Iset} ::= \text{Iname} \mid \text{Iname}, \text{Iset}$$


# Semantic Tuple Templates II

Cname ::= <atom , in Prolog style >

Iname ::= <atom , in Prolog style >

Pname ::= <atom , in Prolog style >

Binding ::= <variable , in Prolog style >

N ::= <number >

PosInt ::= <positive integer number >

- We can define a semantic tuple template like:

```
'Car' ; 'Vehicle'(hasMaker in (hasCountry : italy),  
                  hasPrice < 15000)
```



# Semantic Tuple Centre Primitives

`Op ::= SyntacticOp | SemanticOp`

`SyntacticOp ::= out( SyntacticTemplate ) | in( SyntacticTemplate ) |  
rd( SyntacticTemplate ) | inp ( SyntacticTemplate ) |  
rdp ( SyntacticTemplate )`

`SemanticOp ::= out( Assertion ) | in( Retrieval ) | rd( Retrieval ) |  
inp( Retrieval ) | rdp ( Retrieval )`

`SyntacticTemplate ::= <first order tuple ; tuple name != 'semantic'>`

`Assertion ::= semantic Individual`

`Retrieval ::= semantic Result matching (C)`

`Result ::= Individual | Var`

`Var ::= <variable , in Prolog style >`

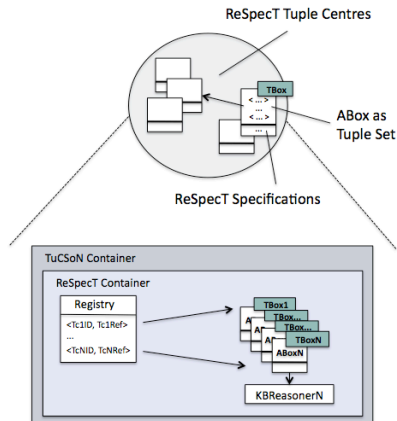
- We can define operation like:

```
out(semantic fiesta:  'Car'(hasMaker :  ford))  
in(semantic X matching ('Car' , exists hasMaker / Y in {ferrari,  
maserati}))
```



# An Extension of ReSpecT Framework I

- Tuple centres are provided in each TuCSoN node by the ReSpecT **container**.
  - The container represents the manager of the tuple centre life-cycle and provides the API to create, access, and use them.
- ⇒ The container has to provide the API to:
- ▶ create a tuple centre associated with a specified TBox and a reasoner;
  - ▶ set and obtain the TBox related to a particular tuple centre.



# An Extension of ReSpecT Framework II

- We adopted the OWL reasoner [Pellet](#) [Sirin et al., 2007] because:
  - ▶ is easy to integrate it with ReSpecT since it is open-source and it is written in java;
  - ▶ is a complete and capable OWL-DL reasoner with good performance.
  - ▶ is able to answer to [SPARQL](#) queries.
- When the container create a new tuple centre:
  - ▶ by exploiting the Pellet API, a new instance of ontology is created from a specified OWL file and provided to the tuple centre;
  - ▶ a new instance of the Pellet reasoner is created and provided to the tuple centre.



# An Extension of ReSpecT Framework III

- In face of an **out** primitive a tuple centre:
  - ▶ through **semantic** operator the individual expressed by the received semantic tuple is interpreted;
  - ▶ by exploiting the Pellet reasoner, the individual consistency with the ontology is checked;
  - ▶ the individual is inserted in the ABox.
- In face of a **in** or **rd** primitive a tuple centre:
  - ▶ through **semantic** operator the concept specification expressed by the received semantic tuple template is interpreted;
  - ▶ by exploiting the Pellet reasoner the concept specification is checked with the ontology;
  - ▶ through **matching** operator the concept specification is interpreted and converted in **SPARQL** query;
  - ▶ the first individual obtained by the reasoner is converted in a logic term;
  - ▶ the obtained individual is unified first operand **matching** operator.



# An Extension of ReSpecT Framework IV

- Besides semantic tuples, tuple templates, and tuple matching mechanism, in a semantic tuple centre it is useful to preserve the possibility of using no semantic ones.
  - ⇒ No semantic tuples, tuple templates, and tuple matching mechanism are useful to realise coordination mechanisms in tuple centres.



# Simple Example

user\_preferences\_tc

```
user(user1, 'Car'(hasMaxspeed>160,  
hasMaker in {renault, ford}))
```

car\_tc

```
ka:'CityCar'(hasMaker=ford, hasMaxSpeed=165,  
hasPrice=8000)  
mondeo:'FamilyCar'(hasMaker=ford, hasMaxSpeed=195,  
hasPrice=11000)  
twingo:'CompactCar'(hasMaker=renault, hasMaxSpeed=150,  
hasPrice=6000)
```

```
reaction(out(entered_user(UId)),  
  (operation, invocation)  
  (rd(user(UId, Preferences)),  
   rd(semantic Car matching Preferences),  
   write display(Car))).
```

- When an operation like `out(entered_user(user1))` is executed on the `user_preferences_tc` tuple centre, for example the command `write_display(ka:'Car'(hasMaker=ford, hasMaxSpeed=165, hasPrice=8000))` is executed.



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points**
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications

# Main Open Points

- A SHOIN(D)-like definition language for TBox.
- Extension to **SROIQ(D)** (OWL 2)
- Distributed ABox / TBox.



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres**
- 6 Bibliography
- 7 Publications

# Medical-Data Sharing I

- In collaboration with the Prof. Dr. Michael Ignaz Schumacher of Institute of Business Information Systems at the University of Applied Sciences Western Switzerland in Sierre.
- Scenario:
  - ▶ Patient electronic health records are distributed in fragments over the network.
  - ▶ A doctor, a medical assistant, or more in general people with a specific role should be able to store and / or access patient data at any time and from any location.



# Medical-Data Sharing II

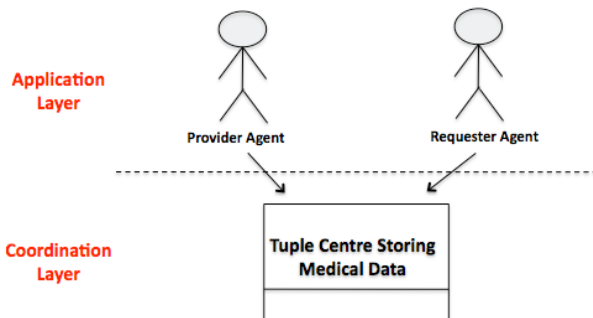
- Requirements:

- ▶ **Time independence**: patient data should be published and accessible at any point in time even if a prior data provider is currently not available.
- ▶ **Location independence**: patient data should be accessed without need to know where data providers are placed.
- ▶ **Reference independence**: patient data should be accessed without need to know data providers.
- ▶ **Schema independence**: the internal representation of patient data should be independent of each provider schema, thus achieving interoperability.
- ▶ Medical data are private and distributed in a federated network. As a consequence, **security** at the application vs. node, node, domain, and inter-domain level, represents a further requirement.



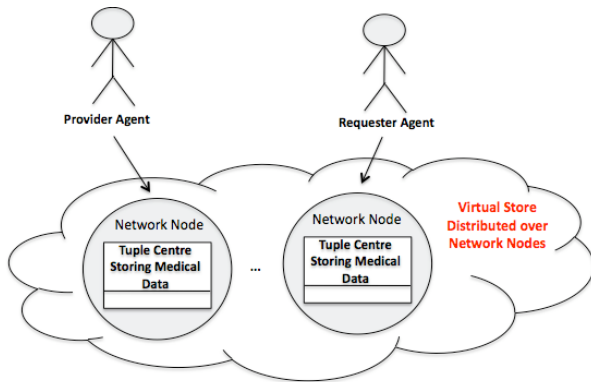
# Exploiting Semantic Tuple Centres I

- A coordination model is needed for managing interaction between medical-data providers and requesters in a mediated (no-direct) way.
- The tuple centre coordination model shows features suitable for such scenario:
  - ⇒ Generative communication lead to time and reference independence.



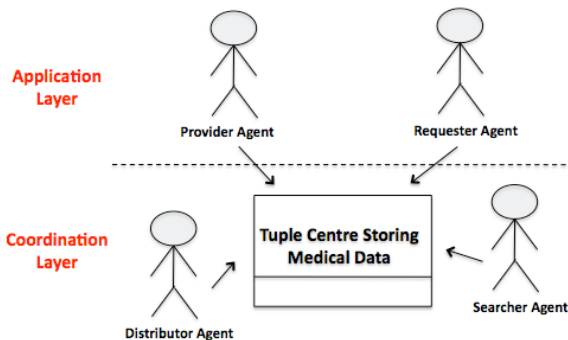
# Exploiting Semantic Tuple Centres II

- **Location independence:** providers and requesters will not need to know where medical data is placed.
  - ⇒ Data should be stored in some kind of virtual storage spread over several nodes.
  - ⇒ Medical data is both written and read locally by providers even though it is globally distributed among nodes.



# Exploiting Semantic Tuple Centres III

- It is possible to exploit a set of agents and tuple centres at the coordination level, in order to both distribute and search medical data in the virtual store.

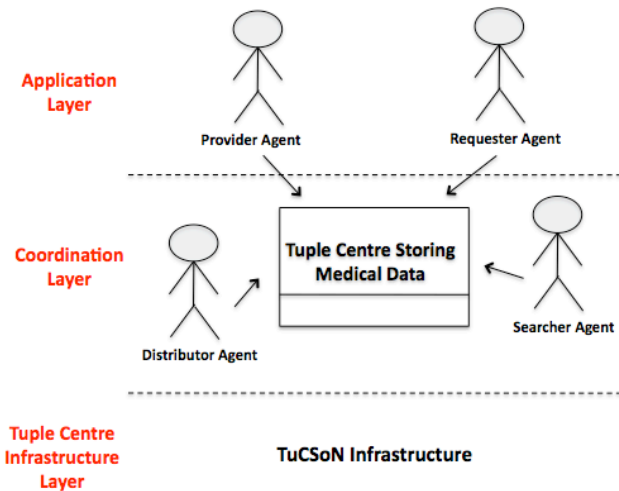


# Exploiting Semantic Tuple Centres IV

- Through **semantic tuple centres** it is possible to face **schema independence** requirement by:
  - ▶ adopting a common ontology and
  - ▶ describing each medical data fragment as an individual in terms of a semantic tuple.



# Exploiting TuCSoN Infrastructure



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography**
- 7 Publications

# Bibliography I



Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003).

*The Description Logic Handbook: Theory, Implementation, and Applications.*

Cambridge University Press.



Bandara, A., Payne, T., Roure, D. D., Gibbins, N., and Lewis, T. (2008).

Semantic resource matching for pervasive environments: The approach and its evaluation.

Technical Report ECSTR-IAM08-001, Southampton, UK.



Ciancarini, P. (1996).

Coordination models and languages as software integrators.

*ACM Comput. Surv.*, 28(2):300–302.



# Bibliography II



Horrocks, I., Patel-Schneider, P. F., and Harmelen, F. V. (2003).  
From shiq and rdf to owl: The making of a web ontology language.  
*Journal of Web Semantics*, 1:2003.



Jennings, N. R. (2001).  
An agent-based approach for building complex software systems.  
*Communications of the ACM*, 44(4):35–41.



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



# Bibliography III



Omicini, A. (1999).

On the semantics of tuple-based coordination models.

In *1999 ACM Symposium on Applied Computing (SAC'99)*, pages 175–182, San Antonio, TX, USA. ACM.



Omicini, A. and Denti, E. (2001).

From tuple spaces to tuple centres.

*Science of Computer Programming*, 41(3):277–294.



Omicini, A., Ricci, A., and Viroli, M. (2005).

RBAC for organisation and security in an agent coordination infrastructure.

*Electronic Notes in Theoretical Computer Science*, 128(5):65–85.

2nd International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'04), 30 August 2004. Proceedings.



# Bibliography IV



Omicini, A. and Zambonelli, F. (1999).

Coordination for Internet application development.

*Autonomous Agents and Multi-Agent Systems*, 2(3):251–269.

Special Issue: Coordination Mechanisms for Web Agents.



Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P. (2002).

Semantic matching of web services capabilities.

In *ISWC '02: Proc. of the First Int'l Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK. Springer.



Rossi, D., Cabri, G., and Denti, E. (2001).

Tuple-based technologies for coordination.

pages 83–109.



Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996).

Role-based access control models.

*Computer*, 29(2):38–47.



# Bibliography V



Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007).  
Pellet: A practical owl-dl reasoner.  
*J. Web Sem.*, 5(2):51–53.



# Outline

- 1 Introduction
- 2 Semantic Tuple Centres
- 3 Semantic Tuple Centre in TuCSoN
- 4 Open points
- 5 Medical-Data Sharing via Semantic Tuple Centres
- 6 Bibliography
- 7 Publications**



# Publications I

- Elena Nardini, Ambra Molesini, Andrea Omicini, Enrico Denti. SPEM on Test: the SODA Case Study. *23th ACM Symposium on Applied Computing (SAC 2008)*, 6-20 March 2008.
- Elena Nardini, Andrea Omicini, Maria Cristina Matteucci. Toward a Framework for Collaborative Learning based on Agent-based Technologies. *The 2008 International Education, Technology and Development Conference (INTED 2008)*, -5 March 2008.
- Elena Nardini, Matteo Casadei, Andrea Omicini, Pietro Gaffuri. A Conceptual Framework for Collaborative Learning Systems Based on Agent Technologies. *The 2008 International Conference on the Interactive Computer Aided Learning (ICL 2008)*, 24-26 September 2008.



## Publications II

- Elena Nardini, Andrea Omicini. Agent-Based Collaboration Systems: A Case Study. *Knowledge Construction in E-learning Context: CSCL, ODL, ICT and SNA in Education, CEUR Workshop Proceedings 398*, October 2008.
- Ambra Molesini, Elena Nardini, Enrico Denti, Andrea Omicini. Advancing Object-Oriented Standards Toward Agent-Oriented Methodologies: SPEM 2.0 on SODA. *9th Workshop "From Objects to Agents" (WOA 2008) volution of Agent Development: Methodologies, Tools, Platforms and Languages*, November 2008.
- Maria Cristina Matteucci, Andrea Omicini, Elena Nardini, Pietro Gaffuri. Knowledge Construction in E-learning Context: CSCL, ODL, ICT and SNA in Education. *CEUR Workshop Proceedings 398*, 1-2 September 2008.



## Publications III

- Ambra Molesini, Elena Nardini, Enrico Denti, Andrea Omicini. Situated Process Engineering for Integrating Processes from Methodologies to Infrastructures. *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, 8-12 March 2009.
- Elena Nardini, Andrea Omicini, Mirko Viroli. General-Purpose Coordination Abstractions for Managing Interaction in MAS. *The WI-IAT 2009 Workshops Proceedings*, 15-18 September 2009.
- Ambra Molesini, Marco Prandini, Elena Nardini, Enrico Denti. Risk Analysis and Deployment Security Issues in a Multi-Agent System. *The 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010)*, 2010, 22-24 January 2010.
- Elena Nardini, Mirko Viroli, Emanuele Panzavolta. Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres. *The 25th Annual ACM Symposium on Applied Computing (SAC 2010)*, 22-26 March 2010.



# Agent-based Models, Technologies and Methodologies for the Engineering of Software Systems in the Context of Open Scenarios

Dottorato di Ricerca in Informatica, Elettronica e Telecomunicazioni  
XXIII Ciclo  
Relazione di Fine Secondo Anno

**Elena Nardini<sup>1</sup>**

<sup>1</sup>Alma Mater Studiorum – Università di Bologna  
[elena.nardini@unibo.it](mailto:elena.nardini@unibo.it)

Bologna - January 14, 2010

