Semantic Coordination Through Programmable Tuple Spaces

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

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

Bologna - January 17, 2011
Outline

1. Background and Motivation
2. Research Contribution
3. Application Scenarios
4. Other Research Activities
5. Bibliography
6. Publications
Software systems of today – like Internet-based, pervasive and self-organising systems – are mainly [Omicini and Viroli, 2011]:

- **Distributed** in control, space and time
- **Open** in terms of heterogeneity and dynamism
- **Knowledge-intensive**—most of the activities are knowledge-based

**Coordination** is the activity of managing/costraining the software components interaction space

→ **Coordination models and languages**
  [Papadopoulos and Arbab, 1998, Busi et al., 2001] promoting **uncoupling** among system components play a key role here
The Tuple-Space Model [Rossi et al., 2001]

- Represent the main class of space-based coordination models [Papadopoulos and Arbab, 1998]
  - Communication and coordination occur through a shared data space
  - Generative communication [Gelernter, 1985] as essential environment abstraction supporting openness in distributed systems
  - Associative access fits well with knowledge-intensive systems
Related Work [Omicini and Viroli, 2011]

- The original formulation of tuple spaces is within the LINDA model [Gelernter, 1985]

- A number of implementations and extensions have been developed and proposed in literature—e.g. Sun’s JavaSpaces [Freeman et al., 1999] and GigaSpaces [GigaSpaces, 2007]

- Many other proposals have instead focussed on extending the tuple-based coordination model beyond its original limitations. In particular, two are notable [Omicini and Viroli, 2011]:
  - Those focussing on the programmability of the behaviour of tuple-based communication abstraction
  - Those enhancing tuple-based communication with semantics
What is Needed

- Behaviour programmability leads to **coordination uncoupling**
- Semantic support leads to **semantic uncoupling**

→ Those support in tuple spaces are both essential requirements for open, distributed and knowledge-intensive systems, but...

... none of the tuple-space-based approaches in literature accounts for both
Outline

1. Background and Motivation
2. Research Contribution
3. Application Scenarios
4. Other Research Activities
5. Bibliography
6. Publications
Research Aim

- Design a new coordination abstraction including three key elements:

  - Tuple-based coordination ↔ Communication uncoupling
  - Behaviour programmability ↔ Coordination uncoupling
  - Semantic support ↔ Semantic uncoupling

- No assumptions about the application context and keeping of the conceptual integrity with the original tuple-space model

- Modelled as a runtime first-class abstraction [Ricci and Omicini, 2003] promoting online engineering
  - “Kept alive” through the whole software engineering process
  - Enabling runtime inspection and modification of their current state
Towards Semantic Tuple-Centres

- **Tuple centres** [Omicini and Denti, 2001] are tuple spaces whose behaviour can be determined through a specification language defining how a tuple centre should react to incoming/outcoming communication events.

- Their TuCSoN implementation [Omicini and Zambonelli, 1999] seems to provide the coordination abstraction closest to the research aim:
  - Tuple centres modelled as runtime first-class abstractions
  - A behaviour specification can associate any event possibly occurring in the tuple centre to a set of computational activities called **reactions**
  - Reactions are specified with ReSpecT [Omicini, 2007]—a general-purpose, logic-based specification language

→ **General-purpose semantic tuple-space model**
  - Starting from the tuple centre model
  - Semantically enriching the model
The Semantic Tuple-Centre Model

- An ontology is a formal explicit specification of a shared conceptualisation in terms of concepts, individuals and of relations among them [Gruber, 1995]

- **Ingredients:**
  1. Domain terminology
  2. Semantic tuples
  3. Semantic templates
  4. Semantic primitives
  5. Semantic reactions
  6. Semantic matching

- Semantic tuple-centres formally defined through the conceptual framework of SHOIN(D) [Horrocks et al., 2003]
  - A very expressive Description Logic (DL) [Baader et al., 2003]
  - Represents the counterpart of OWL DL—a kind of OWL, that is the W3C standard for the Semantic Web
Semantic Tuple Centres in TuCSoN—Domain Terminology

- Describes the domain concepts and relations attached to a tuple centre
- Formally defined as SHOIN(D) TBox—describing the terminological axioms: concepts and relations
- OWL DL as domain ontology language for TuCSoN tuple centres
  - OWL is the W3C standard ontology description language for the Semantic Web, and the standard de-facto for semantic applications in general

Example:

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

<owl:Class rdf:ID="CityCar">
  <rdfs:subClassOf rdf:resource="\#Car"/>
</owl:Class>
```
Semantic Tuple Centres in TuCSoN—Semantic Tuples

- Represent domain individuals semantically interpreted by the domain terminology
- Formally defined through a SHOIN(D) ABox—defining the axioms to assert specific domain objects ($C(a)$) and their relations ($R(a,b)$)

**Semantic tuple grammar:**

```
Individual ::= iname '::' C
C ::= cname | cname '(': R ')' 
R ::= rname '::' V | rname 'in' '(' Vset ')' | R ',' R
Vset ::= V | V ',' Vset
V ::= iname | string | int | float
```

**Example:**

1. Car(f550)
2. hasMaker(f550, ferrari)
3. hasMaxSpeed(f550, 285)
4. hasColour(f550, red)
5. hasColour(f550, black)

f550 : 'Car' (hasMaker : ferrari, hasMaxSpeed : 285, hasColour in (red, black))
Semantic Tuple Centres in TuCSoN—Semantic Templates

- Consist in specifications of sets of domain individuals described by the domain terminology
- Formally defined as SHOIN(D) TBox axioms—concept and role descriptions

**Semantic template grammar:**

```
C ::= 'All' | 'None' | cname | C 'and' C | C 'or' C | 'not' C | D |
    | '{' [ iname {',', iname } ] '}' | C '( ' D ')' | '(' C ')' |
    | 'String' | 'Int' | 'Float'
D ::= F | 'exists' F | 'only' F | M
F ::= R 'in' C | R ':' I | R ':' I | R '::' Msymb N |
    | R '::' 'eq' string | R
M ::= '#' R Msymb N
R ::= rname | rname '/' vname
Msymb ::= 'gt' | 'lt' | 'geq' | 'leq' | 'eq'
```
**Semantic Tuple Centres in TuCSoN—Semantic Templates**

**Grammar mapping:**

<table>
<thead>
<tr>
<th>$SHOIN(D)$</th>
<th>$Language Expression$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>All</td>
</tr>
<tr>
<td>$\perp$</td>
<td>None</td>
</tr>
<tr>
<td>$C \sqcap D$</td>
<td>C and D</td>
</tr>
<tr>
<td>$C \sqcup D$</td>
<td>C or D</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>not C</td>
</tr>
<tr>
<td>$\forall R.C$</td>
<td>only R in C</td>
</tr>
<tr>
<td>$\exists R.C$</td>
<td>exists R in C</td>
</tr>
<tr>
<td>$&gt; n R, \leq n R$</td>
<td># $R$ lt n, # $R$ leq n</td>
</tr>
<tr>
<td>$&gt; n R, \geq n R$</td>
<td># $R$ gt n, # $R$ geq n</td>
</tr>
<tr>
<td>$= n R$</td>
<td># $R$ eq n</td>
</tr>
<tr>
<td>${a_1, \ldots, a_n}$</td>
<td>${iname_1, \ldots, iname_n} $</td>
</tr>
<tr>
<td><strong>concrete domains</strong></td>
<td>String, Int, Float</td>
</tr>
<tr>
<td><strong>concrete domain operators</strong></td>
<td>String: eq. Int, Float: eq, lt, leq, gt, geq.</td>
</tr>
</tbody>
</table>

**Example:**

\[ Car \sqcap \exists \text{hasMaker\.ford} \]

‘Car’ and (exists hasMaker : ford)  
‘Car’ and (exists hasMaker / X in ‘Maker’)
Semantic Tuple Centres in TuCSoN—Semantic Primitives

- Represent the language whereby system components can read (rd), consume (in) and write (out) tuples described by the domain terminology.
- Require to revisit the semantic of the basic primitives:
  - Check whether the roles and concepts associated to the individuals and concepts exist in the domain terminology.
  - While an out in a tuple centre always succeeds, in a semantic tuple centre may fail. In particular, in face of the primitive has to be checked:
    - The individual already exists in the tuple centre.
    - The consistency of the tuple to be written with the domain ontology—ABox plus TBox.
    → Such kind of checks can be performed through any DL reasoner.
- The primitive syntax has to be extended in order to:
  - Include both semantic and syntactic primitives.
  - Obtain an individual as a result from a semantic template.

Example:

```
out(specific fiat500: 'CityCar' hasMaker: fiat))
in(specific Result matching ('CityCar' and (exists hasMaker: fiat)))
```
Semantic Tuple Centres in TuCSoN—Semantic Reactions

- Sets of computational activities within a tuple centre defined through a reaction specification language

- A ReSpecT reaction is of kind $\text{reaction}(E, (G, R))$

- In a semantic view:
  - $E$ represents a specification of a coordination primitive related to a concept description ($in/rd$) or to a domain individual ($out$)
  - $E$ should contain a concept description expressed in terms of the semantic template language

Example:

```
reaction( in(semantic ‘Car’), ..., ... )
in(semantic Result matching (‘CityCar’ and (exists hasMaker : fiat)))
reaction( out(semantic ‘Car’ and (exists hasMaker : fiat)), ..., ...)
out(semantic fiat500: ‘CityCar’( hasMaker : fiat))
```
In a semantic view:

- $G$ represents a set of constraints on the communication event $Ev$
- $G$ is extended so that the guard could contain a concept description expressed in terms of the semantic template language

**Example:**

```
reaction( out(spectral 'CityCar'), spectral (exists hasMaker : fiat), ...)
```

- $R$ can read, remove and write tuples from/to the tuple centre
- $R$ can contain semantic primitives

**Example:**

```
reaction( out(spectral 'CityCar'), spectral (exists hasMaker : ford),
   rd (spectral ford 'Maker' and (exists hasCars / N in Int)),
   N1 is N + 1,
   out(spectral ford : 'Maker'(hasCars : N1)) )
```
Semantic Tuple Centres in TuCSoN—Semantic Matching Mechanism

- When a semantic reading/consuming primitive is performed, the matching mechanism should identify and retrieve an individual matching the semantic template.

- Within a reaction, the semantic matching mechanism should work in two ways:
  - If \( E \) describes a writing event, matching \( E \) with \( Ev \) consists in checking:
    - if \( Ev \) is a writing event
    - if the individual in \( Ev \) belongs to the concept described in \( E \) and \( G \)
  - If \( E \) describes a consuming/reading event, matching \( E \) with \( Ev \) consists in checking:
    - if \( Ev \) is a consuming/reading event
    - if the concept description in \( Ev \) is a sub-concept of the concept in \( E \) and \( G \)

- The matching mechanism can be easily obtained by exploiting the reasoning service provided by any DL reasoner: Instance retrieval, Instance checking, and Subsumption checking.
TuCSoN Semantic Tuple-Centre Architecture

- **I SemanticKB** provides: *load ontology, assert individual, delete individual, instance checking, instance retrieval and subsumption checking*

  - **Query generator** is exploited to create a **SPARQL** query [Pérez et al., 2009] for the execution of *instance checking, instance retrieval and subsumption checking*
  - **Pellet reasoner** [Sirin et al., 2007] is exploited to execute **SPARQL** queries and *consistence checking*
Example of Usage

ReSpecT reactions

reaction(out(semantics 'CityCar'),
    semantic(exists hasMaker:ford),
    rd(semantics ford 'Maker' and 
        (exists hasCars/Num in 'Int'),
    N1 is Num+1,
    out(semantics ford:'Maker'(hasCars:N1)))

Tuple space

fiat500: 'CityCar'(hasMaker: fiat)

ka: 'CityCar'(hasMaker: ford)

Car Domain Terminology
Some Evaluation Results

- **Evaluation aim**: Evaluate how much the use of semantic techniques affects the tuple centre behaviour in terms of performance

  → Test the implementation of semantic tuple centres with Pellet against two reference ontologies for DL reasoner benchmarks [Bock et al., 2008]:

    - *lumb* ontology: covers only part of the inference supported by OWL DL. It contains 44 classes with 36 subclass axioms and 6 equivalent class axioms and 32 roles
    - *wine* ontology: more complex because cover OWL DL, in particular it exploits *nominals*—that lead to performance decay. It contains 212 classes with 311 subclass axioms and 96 equivalent class axioms and 30 roles

- The test was executed on an Mac Pro, with two processors Dual-Core Intel Xeon with speed 2.66 GHz and 2 GB of RAM
Some Evaluation Results: Lumb Ontology
Some Evaluation Results: Lumb Ontology
Some Evaluation Results: Lumb Ontology

- In with 1120 individuals in ABox
Basic Papers


Medical-Data Sharing via Semantic Tuple Centres

- 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

- Requirements: communication, coordination and semantic uncopuling

- Semantic tuple centres in order to face with such requirements

- Collaboration financed from COST Action IC0801
  http://www.agreement-technologies.eu/

- Project submitted to the Swiss National Science Foundation (SNSF)
  http://www.snf.ch/E/Pages/default.aspx
SAPERE Project

- **Main aim**: Development of a highly-innovative theoretical and practical framework for the decentralized deployment and execution of self-aware and adaptive services for future and emerging pervasive network scenarios.

- Early research ideas
  - Tuple spaces + chemical reactions as coordination laws [Viroli and Casadei, 2009]
  - Tuples have a concentration (a.k.a. weight, or activity value)
  - Concentration is evolved exactly as in chemistry
  - Some reactions can even fire a tuple from one space to another

- Match between chemical laws and reactants:
  - **Semantic matching** in order to deal with openness requirements
  - Application-dependent **fuzzy match** function $\mu(t, t')$ in order to deal with vagueness of information [Lukasiewicz and Straccia, 2007]
Basic Papers


Outline

1 Background and Motivation
2 Research Contribution
3 Application Scenarios
4 Other Research Activities
5 Bibliography
6 Publications
Agent Oriented Software Engineering (AOSE) and Agent-Oriented Methodologies


Computer Supported Collaborative Learning


Agent Coordination Context

Conclusion

- Model and implementation of semantic tuple centres—a basic brick of coordination infrastructures for open, distributed and knowledge-intensive systems
- Description of model, of the architecture and the implementation, and evaluation of performance
- Ongoing work:
  - Investigation of kinds of TBox and ABox that can be used with TuCSoN tuple centres
  - Evaluating other semantic approaches (or reasoners) that can trade-off speed for expressiveness
  - From the model viewpoint, evaluating of a basic extension concerns the introduction of fuzzyness, relying on approaches like fuzzyDL [Bobillo and Straccia, 2008]
  - The model extension is particularly suitable for complex application scenarios like self-organising pervasive computing domains—like e.g. pervasive and wearable displays [Ferscha and Vogl, 2010]
Outline

1. Background and Motivation
2. Research Contribution
3. Application Scenarios
4. Other Research Activities
5. Bibliography
6. Publications
Bibliography I


GigaSpaces (2007).
Home page.

Toward principles for the design of ontologies used for knowledge sharing.

From shiq and rdf to owl: The making of a web ontology language.

Managing uncertainty and vagueness in description logics for the semantic web.


Ricci, A. and Omicini, A. (2003). Supporting coordination in open computational systems with TuCSoN.


Outline

1. Background and Motivation
2. Research Contribution
3. Application Scenarios
4. Other Research Activities
5. Bibliography
6. Publications
Publications I


Publications II

- 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)*


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. **The paper was selected as a best paper.**


Semantic Coordination Through Programmable Tuple Spaces

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

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

Bologna - January 17, 2011