Matteo Baldoni, Cristina Baroglio
Federico Bergenti, Alfredo Garro (eds.)

From Objects to Agents

XIV Workshop, WOA 2013
Torino, Italy, December 2nd-3rd, 2013

Workshop Notes

WOA 2013 Home Page:
`http://di.unito.it/woa2013`

Preface

Agent-based technologies, developed in the Artificial Intelligence area, have become more and more important especially in more traditional Computer Science areas, like Software Engineering, where the agent abstraction is considered a natural extension of the object abstraction. The importance of these techniques is also witnessed in the industrial sector by their use in the development of tools and applications.

Following the success of WOA 2000 in Parma, WOA 2001 in Modena, WOA 2002 in Milano, WOA 2003 in Villasimius, WOA 2004 in Torino, WOA 2005 in Camerino, WOA 2006 in Catania, WOA 2007 in Genova, WOA 2008 in Palermo, WOA 2009 in Parma, WOA 2010 in Rimini, WOA 2011 in Cosenza, WOA 2012 in Milano, WOA 2013 was hosted in Torino.

This year event, celebrating the forteenth workshop edition, was co-located with the conference of the Italian Association for Artificial Intelligence. On this occasion we took stock of whether the agent technology can still be considered a scion of Artificial Intelligence and to which extent it can still be considered as connected to the object technology. We were honored to have Rafael Heitor Bordini as an invited speaker. His talk was intitled "Jason Comes of Age: 10 Years of Progress in Multi-Agent Oriented Programming".

This volume contains *sixteen* papers, selected by the Programme Committee. Each paper received at least three reviews in order to supply the authors with helpful feedback that could stimulate the research as well as foster discussion.

We would like to thank all authors for their contributions, the members of the Steering Committee for the valuable suggestions and support, and the members of the Programme Committee for their excellent work during the reviewing phase.

November 25th, 2013

Matteo Baldoni
Cristina Baroglio
Federico Bergenti
Alfredo Garro

Program Committee

Matteo Baldoni	Dipartimento di Informatica, Univ. di Torino
Cristina Baroglio	Dipartimento di Informatica, Università di Torino
Federico Bergenti	Universita' degli Studi di Parma
Giacomo Cabri	Università di Modena e Reggio Emilia
Federico Chesani	University of Bologna
Rino Falcone	Istituto di Scienze e Tecnologie della Cognizione, CNR Roma
Nicoletta Fornara	Universita della Svizzera Italiana, Lugano
Giancarlo Fortino	University of Calabria
Alfredo Garro	University of Calabria
Elisa Marengo	Dipartimento di Informatica, Università di Torino
Viviana Mascardi	DIBRIS (Department of Informatics, Bioengineering, Robotics and System Engineering), University of GENOVA, IT
Emanuela Merelli	University of Camerino
Andrea Omicini	Alma Mater StudiorumUniversità di Bologna
Paolo Petta	Austrian Research Institute for Artificial Intelligence
Agostino Poggi	University of Parma
Giovanni Rimassa	Whitestein Technologies AG
Andrea Santi	Universita' di Bologna
Corrado Santoro	
Paola Turci	University of Parma
Eloisa Vargiu	Barcelona Digital Technology Center
Mirko Viroli	Alma Mater Studiorum - Università di Bologna

Additional Reviewers

Giuliani, Alessandro

Keyword Index

Table of Contents

Space-aware Coordination in ReSpecT

Stefano Mariani
DISI, ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: s.mariani@unibo.it

Andrea Omicini
DISI, ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: andrea.omicini@unibo.it

***Abstract*—Spatial issues are essential in new classes of complex software systems, such as pervasive, multi-agent, and self-organising ones. Understanding the basic mechanisms of spatial coordination is a fundamental issue for coordination models and languages in order to deal with such systems, governing situated interaction in the spatio-temporal fabric.**

Along this line, in this paper we make *space-aware coordination media* out of ReSpecT tuple centres, by introducing the few basic mechanisms and constructs that enable the ReSpecT language to face most of the main challenges of *spatial coordination* in complex software systems.

I. INTRODUCTION

Complex socio-technical systems in pervasive computing scenarios are stressing more and more the requirements for coordination middleware [1]. In particular, the availability of a plethora of mobile devices, with motion sensors and motion coprocessors, is pushing forward the needs for *space-awareness* of computations and systems: awareness of the spatial context is often essential to establish which tasks to perform, which goals to achieve, and how.

More generally, spatial issues are fundamental in many sorts of complex software systems, including intelligent, multi-agent, adaptive, and self-organising ones [2]. In most of the application scenarios where *situatedness* plays an essential role, coordination is required to be *space aware*. This is implicitly recognised by a number of proposals in the coordination field trying to embody spatial mechanisms and constructs into the coordination languages – such as TOTA [3], $\sigma\tau$-LINDA [4], GEOLINDA [5], and SAPERE [1] – which, however, are mostly tailored around specific application scenarios.

In this paper we aim at devising out the *basic* mechanisms and constructs required to *generally* enable and promote *space-aware coordination*. Along this line, we introduce the general notion of space-aware coordination medium (Section II), then we show how the ReSpecT coordination media and language can be extended to support space-aware coordination (Section III). After sketching the semantics of the spatial extension (Section IV), Section V shows some meaningful examples of spatial coordination using ReSpecT. Section VI discusses related research, then Section VII provides for final remarks.

II. SPACE-AWARE COORDINATION MEDIA

A. Spatial Issues

Spatial coordination requires spatial *situatedness* and *awareness* of the coordination media, which translates in a number of technical requirements.

a) Situatedness: First of all, *situatedness* requires that a *space-aware coordination abstraction* should at any time be associated to an absolute positioning, both physical (i.e., the position in space of the computational device where the medium is being executed on) and virtual (i.e. the network node on which the abstraction is executed). If not a must-have, geographical positioning is also desirable, and quite a cheap requirement, too, given the widespread availability of mapping services nowadays.

More generally, this concerns both *position* and *motion* – every sort of motion –, which in principle include speed, acceleration, and all variations in the space-time fabric, also depending on the nature of space. In fact, software abstractions may move along a *virtual* space – typically, the network – which is usually *discrete*, whereas physical devices (robots, mobile devices) move through a *physical* space, which is mostly *continuous*; software abstractions, however, may also be hosted by mobile physical devices, and share their motion. As a result, a coordination abstraction may move through either a physical, continuous space, (e.g., I am in a given position of a tridimensional physical space) or a virtual, discrete space (e.g., I am on a given network node).

Physical positioning could be either *absolute* (say, I am currently at latitude X, longitude Y, altitude Z), *geographical* (I am in via Sacchi 3, Cesena, Italy), or *organisational* (I am in Room 5 of the DISI, site of Cesena). Absolute positioning is more or less always available in the days of mobile devices, usually through GPS services—which, coupled with mapping services, typically provide some sort of geographical positioning, too. Virtual positioning is available as a network service, and might be also labelled as either absolute (in terms of the node IP number, for instance) or relative (for instance, as a domain/subdomain localisation via DNS). Organisational location should be instead defined application- or middleware-level, and related to either absolute or virtual positioning.

Furthermore, a notion of *locality* may be available—so as to allow for the *local vs. global* dynamics which is typical of complex distributed systems such as pervasive ones. Locality could be strictly bound to positioning, but not necessarily so: being in the same position is not always the same as being in the same location.

b) Awareness: The main requirement of *spatial awareness* is that the ontology of a space-aware coordination medium should contain some notion of space. This means, first of all, that the position of the coordination medium should be available to the coordination laws it contains in order to make them capable of *reasoning about space*—so, to implement *space-aware coordination laws*. So, generally speaking, a range of predicates / functions should be provided to access spatial information associated to any event occurring in the coordination medium (e.g., where the action causing the event took place, where the coordination medium is currently executing), and to perform simple computations over spatial information.

Also, space has to be embedded into the working cycle of the coordination medium: the event model should include *spatial events*, which affect coordination activity by triggering some space-related computation within the coordination abstraction. In fact, associating spatial information to generic events is not enough: space-related laws like "when at home, switch on the lights" cannot be expressed only referring to actions actually performed, but instead require a specialised notion of spatial event (such as "I am at home") to be triggered. So, a spatial event should be generated within a coordination medium, conceptually corresponding to changes in space—so, related to *motion*, such as starting from / arriving to a place. Spatial events should then be captured by the coordination medium, and used to activate space-aware coordination laws, within the normal working cycle of the coordination abstraction.

B. Spatial Tuple Centres

Tuple centres are introduced in TuCSoN [6] as coordination media meant at encapsulating any computable coordination policy within the coordination abstraction. Technically, a tuple centre is a *programmable* tuple space, i.e., a tuple space whose behaviour in response to events can be programmed so as to specify and enact any coordination policy [7], [8]. Tuple centres can then be thought as general purpose coordination abstractions, which can be suitably forged to provide specific coordination services. So, the core idea behind tuple centres is to have first-class coordination abstractions powerful enough to encapsulate and enforce at execution time the laws required to support coordination in complex distributed systems. This does not happen, for instance, in basic LINDA-like models [9], where complex coordination activities surpassing the limited expressive power of tuple-based coordination force spreading the global logic of coordination among individual agents [10].

In the same way as timed tuple centres empower tuple centres with the ability of embodying timed coordination laws [11], *spatial tuple centres* extend tuple centres so as to address the spatial issues depicted in the previous subsection.

First of all, the location a tuple centre is obtained through the notions of *current place*: which could be, for instance, the absolute position in space of the computational device where the coordination medium is being executed on, or the domain name of the TuCSoN node hosting the tuple centre, or the location on the map. Then, motion is conceptually represented by two sorts of spatial events: moving from a starting place, and stopping at an arrival place—in any sort of space / place.

With respect to the formal model defined in [12], this is achieved by extending the input queue of the environment events to become the multiset *SitE* of time, environment, and spatial events, handled as input events by the *situation* transition ($\longrightarrow_s$)—as shortly discussed in Section IV. Whenever some motion of any sorts occurs (such as the physical device starting / stopping, or the node identifier changes), a spatial event is generated, and put in the tuple centre *SitE* multiset, to be handled by the *situation* transition.

Then, analogously to operation and time events, it is possible to specify reactions triggered by spatial events—the so-called *spatial reactions*. Spatial reactions follow the same semantics of other reactions: once triggered, they are placed in the triggered-reaction set and then executed, atomically, in a non-deterministic order. As a result, a spatial tuple centre can be programmed to react to the motion either in physical or in virtual space, so as to enforce space-aware coordination policies.

Finally, a simple notion of locality is provided by the TuCSoN *node* abstraction: when coordination primitives are invoked with no node specification, they are handled as implicitly referring to the *local interaction space* hosted by the node; when a node identifier is instead associated to the invocation, then the primitive explicitly refers to the *global interaction space* [6].

III. SPATIAL ReSpecT

ReSpecT tuple centres are tuple centres based on first-order logic, adopted both for the communication language (logic tuples), and for the behaviour specification language (ReSpecT) [13]. Basically, reactions in ReSpecT are defined as Prolog-like facts of the form `reaction(`*`Activity`*`,` *`Guards`*`,` *`Goals`*`)` which specify the list of the operations (*`Goals`*) to be executed when a given event occurs (called *triggering event*, caused by an *`Activity`*) and some conditions on the event hold (*`Guards`* evaluate to true). Such operations make it possible to insert / read / remove tuples from the tuple space and the specification space of the tuple centre, but also to observe the properties of the triggering event, as well as to invoke operations over other coordination media. The core syntax of ReSpecT is shown in TABLE I.

According to the abstract model described in Subsection II-B, the ReSpecT language is extended to address spatial issues *(i)* by introducing some spatial predicates to get information about the spatial properties of both the tuple centre and the triggering event, and *(ii)* by making it possible to specify reactions to the occurrence of spatial events. The extension to the ReSpecT is shown in TABLE II.

In particular, the following *observation predicates* are introduced for getting spatial properties of triggering events within ReSpecT reactions:[1]

- `current_place(`*`@S, ?P`*`)` succeeds if *`P`* unifies with the position of the node which the tuple centre belongs to
- `event_place(`*`@S, ?P`*`)` succeeds if *`P`* unifies with the position of the node where the triggering event was originated
- `start_place(`*`@S, ?P`*`)` succeeds if *`P`* unifies with the position of the node where the event chain that led to the triggering event was originated

[1]A Prolog-like notation is adopted for describing the modality of arguments: + is used for specifying input argument, − output argument, ? input/output argument, @ input argument which must be fully instantiated.

⟨*Specification*⟩	::=	{⟨*Reaction*⟩ .}
⟨*Reaction*⟩	::=	`reaction` (⟨*Activity*⟩ [, ⟨*Guards*⟩] , ⟨*Goals*⟩)
⟨*Activity*⟩	::=	⟨*Operation*⟩ | ⟨*Situation*⟩
⟨*Operation*⟩	::=	`out` (⟨*Tuple*⟩) | (`in` | `rd` | `no` | `inp` | `rdp` | `nop`) (⟨*Template*⟩ [, ⟨*Term*⟩])
⟨*Situation*⟩	::=	`time` (⟨*Time*⟩) | `env` (⟨*Key*⟩ , ⟨*Value*⟩)
⟨*Guards*⟩	::=	⟨*Guard*⟩ | (⟨*Guard*⟩ {, ⟨*Guard*⟩})
⟨*Guard*⟩	::=	`request` | `response` | `success` | `failure` | `endo` | `exo` | `intra` | `inter` | `from_agent` | `to_agent` | `from_tc` | `to_tc` | `before` (⟨*Time*⟩) | `after` (⟨*Time*⟩) | `from_env` | `to_env`
⟨*Goals*⟩	::=	⟨*Goal*⟩ | (⟨*Goal*⟩ {, ⟨*Goal*⟩})
⟨*Goal*⟩	::=	[⟨*TupleCentre*⟩ ?] ⟨*Operation*⟩ | ⟨*EnvRes*⟩ (`<-` | `->`) `env` (⟨*Key*⟩ , ⟨*Value*⟩) | ⟨*Observation*⟩ | ⟨*Computation*⟩ | (⟨*Goal*⟩ ; ⟨*Goal*⟩)
⟨*Observation*⟩	::=	⟨*Selector*⟩_⟨*Focus*⟩
⟨*Selector*⟩	::=	`current` | `event` | `start`
⟨*Focus*⟩	::=	(`activity` | `source` | `target`) (⟨*Term*⟩) | `time` (⟨*Term*⟩)

TABLE I. ReSpecT SYNTAX: CORE *(without forgeability, bulk, uniform predicates)*

⟨*Situation*⟩	::=	`time` (⟨*Time*⟩) | `env` (⟨*Key*⟩ , ⟨*Value*⟩) | `from` (⟨*Space*⟩ , ⟨*Place*⟩) | `to` (⟨*Space*⟩ , ⟨*Place*⟩)
⟨*Guard*⟩	::=	`request` | `response` | `success` | `failure` | `endo` | `exo` | `intra` | `inter` | `from_agent` | `to_agent` | `from_tc` | `to_tc` | `before` (⟨*Time*⟩) | `after` (⟨*Time*⟩) | `from_env` | `to_env` | `at` (⟨*Space*⟩ , ⟨*Place*⟩) | `near` (⟨*Space*⟩ , ⟨*Place*⟩ , ⟨*Radius*⟩)
⟨*Focus*⟩	::=	(`activity` | `source` | `target`) (⟨*Term*⟩) | `time` (⟨*Term*⟩) | `place` (⟨*Space*⟩ , ⟨*Term*⟩)
⟨*Space*⟩	::=	`ph` | `ip` | `dns` | `map` | `org`

TABLE II. SPATIAL EXTENSIONS TO ReSpecT *(only affected definitions shown)*

where the node position can be specified as either its absolute physical position (*S*=`ph`), its IP number (*S*=`ip`), its domain name (*S*=`dns`), its geographical location (*S*=`map`) – as typically defined by mapping services like Google Maps –, or its organisational position (*S*=`org`)—that is, a location within an organisation-defined virtual topology.

As an example, the reaction specification tuple

```
reaction( in(q(X)),
   ( operation, completion ),
     ( current_place(ph,DevPos),
       event_place(ph,AgentPos),
       out(in_log(AgentPos,DevPos,q(X))) )).
```

when executed, inserts a new tuple (`in_log/3`) with spatial information each time a TuCSoN agent retrieves a tuple of the form `q(_)` from the tuple centre: this implements a sort of *spatial log*, recording absolute positions of both the querying agent and the device hosting the tuple centre.

Also, the following *guard predicates* are introduced to select reactions to be triggered based on spatial event properties:

- `at(@S,@P)` succeeds when the tuple centre is currently executing at the position *P*, specified according to *S*.
- `near(@S,@P,@R)` succeeds when the tuple centre is currently executing at the position included in the spatial region with centre *P* and radius *R*, specified according to *S*.

So, for instance, `near(dns,'apice.unibo.it',2)` succeeds when the tuple centre is currently executing on a device whose second-level domain is `.unibo.it`.

Reactions to spatial events are specified similarly to ordinary reactions, by introducing the following new event descriptors:

- `from(?S,?P)` matches a spatial event raised when the device hosting the tuple centre starts moving from position *P*, specified according to *S*.
- `to(?S,?P)` matches a spatial event raised when the device hosting the tuple centre stops moving and reaches position *P*, specified according to *S*.

As a result, the following are admissible reaction specification tuples dealing with spatial events:

```
reaction(from(?Space,?Place), Guards, Goals).
reaction(to(?Space,?Place), Guards, Goals).
```

As a simple example, consider the following specification tuples (wherever *Guards* is omitted, it is by default `true`):

```
reaction( from(ph,StartP),
  ( current_time(StartT)
    out(start_log(StartP,StartT)) )).
reaction( to(ph,ArrP),
  ( current_time(ArrT)
    out(stop_log(ArrP,ArrT)) )).
reaction( out(stop_log(ArrP,ArrT)),
  ( internal, completion ),
  ( in(start_log(StartP,StartT))
    in(stop_log(ArrP,ArrT))
    out(m_log(StartP,ArrP,StartT,ArrT)) )).
```

which altogether record a simple *physical motion log*, including start / arrival time and position. In fact, the first reaction stores information about the beginning of a physical motion in a `start_log/2` tuple, the second the end of the motion in a `stop_log/2` tuple, whereas the last removes both sort of tuples and records their data altogether in a `m_log/4` tuple, representing the essential information about the whole trajectory of the mobile device hosting the tuple centre.

IV. Semantics

The basic ReSpecT semantics was first introduced in [13], then extended towards time-aware coordination in [11], reshaped to support the notion of coordination artefact in [8], finally enhanced with situatedness in [12]—which represents the reference semantics for ReSpecT till now.

In order to formalise the semantics for the space-aware extension of ReSpecT, two are the main changes with respect to [12]. First of all, a new, generalised event model should be defined to include both spatial events, and spatial information for any sort of event. Then, the *environment* transition, already handling both time and general environment events, should be extended to include spatial events. All other required extensions (such as the formalisation of each spatial construct's semantics) are technically simple, and trivially extend tables in [12].

A. Event Model

The first fundamental extension to the event model is clearly depicted in TABLE II: a new sort of *spatial* ⟨*Activity*⟩ is introduced. In particular, the notion of ⟨*Situation*⟩ is extended with the two spatial activities `from(` ⟨*Space*⟩ ⟨*Place*⟩ `)`, `to(` ⟨*Space*⟩ ⟨*Place*⟩ `)`, reflecting the initial and final stages of a motion trajectory, respectively—in whatever sort of space.

However, spatial extension of the event model cannot be limited to introducing spatial activities: another issue is represented by *spatial qualification* of events, that is, in short, making *all* ReSpecT events featuring spatial properties—in the same way as temporal properties were introduced for all ReSpecT events in [11]. This is represented by the ⟨*Place*⟩ property featured by ⟨*Cause*⟩ – and ⟨*StartCause*⟩, of course –, as shown in TABLE III, where the extended ReSpecT event model is depicted. Essentially, all ReSpecT events are in principle qualified with both time and space properties—the latter one defined as the position (in whichever sort of space) where the (initial) cause of the event takes place. Of course such properties may be defined or not, depending on the facility available when the event is generated. For instance, if absolute physical positioning is made available by the hosting device, and the device is currently in location *P* when an event is generated, the coordination middleware associates *P* to the event as its physical location—which otherwise would be left undefined.

B. Transition

According to [12], the operational semantics of a ReSpecT tuple centre is expressed by a transition system over a state represented by a labelled triple ${}^{OpE,SitE}\langle Tu, Re, Op\rangle_n^{OutE}$—abstracting away from the specification tuples Σ, which are not of interest in this paper. In particular, Tu is the multiset of the ordinary tuples in the tuple centre; Re is the multiset of the triggered reactions waiting to be executed; Op is the multiset of the requests waiting for a response; OpE is the multiset of incoming ⟨*Operation*⟩ events; $SitE$ is the multiset of incoming ⟨*Situation*⟩ events, including time, spatial, and general environment events; $OutE$ is the outgoing event multiset; n is the local tuple centre time.

$OutE$ is automatically emptied by emitting the outgoing events, with no need for special transitions. In the same way, OpE and $SitE$ are automatically extended whenever a new incoming (either operation or situation) event enters a tuple centre—again, no special transitions are needed for incoming events. In particular, $SitE$ is added new environment events by the associated transducers [12], new time events by the passing of time [11], and – in the spatial extension of ReSpecT presented here – also new spatial events whenever some sort of motion takes place.

So, as described in [12], the behaviour of a ReSpecT tuple centre is modelled by a transition system composed of four different transitions: *reaction* ($\longrightarrow_r$), *situation* ($\longrightarrow_s$), *operation* ($\longrightarrow_o$), *log* ($\longrightarrow_l$). Quite intuitively, spatial events are handled – in the same way as time and environment events – by the *situation* transition, which triggers ReSpecT reactions in response to spatial events. As a result, the *situation* transition is the fundamental, and now finally complete, ReSpecT machinery supporting situatedness in the full acceptation of the term—that is, suitably handling reactiveness of the coordination abstraction to time, space, and general environment events.

V. Examples

A. Spheric Broadcasting

In order to demonstrate the simplicity and effectiveness of the new ReSpecT spatial features, in the following we show the ReSpecT implementation of a sort of communication pattern, widely diffused in nature-inspired systems [14]: *spheric broadcasting*. There, a message has to be *locally spread* in the environment, so to be perceived only by nearby agents. In principle, the distance defining such "diffusion neighbourhood" can be thought of as a "straight-line" radius, identifying a three-dimensional sphere around the point where the message is firstly created. Nevertheless, if this message is, e.g., a digital pheromone left by an ant-like agent, other properties may affect message broadcasting: such as time, for instance, making the message unavailable after a while—which can be programmed in ReSpecT, too, thanks to its timed extension [11].

Since we are mainly concerned with the gain in expressiveness provided by the spatial extension to ReSpecT here proposed, the following ReSpecT specification is focussed on spatial-sensitivity only, and can be logically split into two parts: Reactions `1.1`, `1.2` manage spheric broadcasting requests, whereas Reactions `2.1`, `2.2` enact the spreading process.

```
% 1) Get agent message
% 1.1) Delete garbage tuple
reaction( out(spheric(Msg,Radius)), completion,
  in(spheric(Msg,Radius)) ).
% 1.2) Check range then forward
reaction( out(spheric(Msg,Radius)), completion,
  ( current_place(ph,RecPos),          % Current position
    start_place(ph,SendPos),           % Starting position
    in_range(ph,RecPos,SendPos,Radius), % Prolog computation
    start_source(Sender), start_time(Time),
    out(msg(Msg,Sender,Time)),
```

⟨*Event*⟩	::=	⟨*StartCause*⟩ , ⟨*Cause*⟩ , ⟨*Evaluation*⟩
⟨*StartCause*⟩ , ⟨*Cause*⟩	::=	⟨*Activity*⟩ , ⟨*Source*⟩ , ⟨*Target*⟩ , ⟨*Time*⟩ , ⟨*Space:Place*⟩
⟨*Source*⟩ , ⟨*Target*⟩	::=	⟨*AgentId*⟩ | ⟨*TCId*⟩ | ⟨*EnvResId*⟩ | ⊥
⟨*Evaluation*⟩	::=	⊥ | {⟨*Result*⟩}
⟨*Place*⟩	::=	⟨*GPSCoordinates*⟩ , ⟨*IPAddress*⟩ , ⟨*DomainName*⟩ , ⟨*MapLocation*⟩ , ⟨*VirtualPosition*⟩

TABLE III. EXTENDING ReSpecT EVENTS WITH SPACE

```
    rd(neighbours(Nbrs)),           % List of neighbours
    out(forward(Nbrs,Msg,Radius)) )).
% 2) Forward to every neighbour
% 2.1) Delete garbage tuple
reaction( out(forward(Nbrs,Msg,Radius)),
  ( internal, completion ),
  in(forward(Nbrs,Msg,Radius)) ).
% 2.2) Neighbour list not empty
reaction( out(forward([Nbr|Nbrs],Msg,Radius)),
  ( internal, completion ),
  ( Nbr ? out(spheric(Msg,Radius)),  % Forward
    out(forward(Nbrs,Msg,Radius)) )).% Iterate
```

More specifically,

- Reaction `1.2` reacts to spheric broadcasting requests:
  - by acquiring spatial information about the physical position of the ReSpecT tuple centre currently managing the request, as well as about the entity – either an agent or another tuple centre – sending the request;
  - by checking if current tuple centre is within the given `Radius` w.r.t. the sender's position;
  - if so, it actually stores the message – along with some contextual information – then starts the spreading process;
  - if not so, ReSpecT transactional semantics makes the whole reaction fail, rollbacking all changes made (if any).
- Reaction `1.1` simply consumes the request tuple—no longer needed both in case of `in_range/3` predicate success and failure.
- Reaction `2.2` exploits the *linkability* feature of ReSpecT tuple centres to forward spheric broadcast request to each neighbour, iteratively.
- Reaction `2.1` simply removes the forwarding tuple when all neighbours have been considered.

Given a virtual topology is reified in the tuple `neighbours` – e.g. resembling physical network links between nodes – and the above ReSpecT specification is installed on every tuple centre of a network, we get that any message embedded in a `spheric` tuple is autonomously spread *solely* to "`Radius`-reachable" neighboring tuple centres. This is made possible by the `start_place/2` observation predicate, available in every tuple centre to inspect the place where the *first* spheric broadcast request was issued, thus enabling (physical) range check regardless of the number of tuple centre hops taken to forward the request.

Furthermore, one should note how completely different communication patterns could be obtained through small modifications of the ReSpecT code above:

- by replacing the space qualifier `ph` with `ip` or `dns`, one could achieve, e.g., *subnet broadcasting*, that is, a broadcast communication limited to a given subnet, where such a subnet can be arbitrarily computed through the `in_range/4` predicate.
- by replacing event view predicate `start` with `event`, one could achieve, e.g., *neighborhood-driven, global broadcast*, that is, a broadcast communication covering the whole network, where each node forwards messages to its (spheric) neighborhood solely, but recursively. This is made possible by the fact that `event` considers the *direct cause* of the event, not the original one – as `start` does –, hence predicate `in_range/4` is in turn affected. Nevertheless, suitable ReSpecT reactions should be added to avoid flooding.

This should illustrate the expressive power conveyed by every single predicate: changing even a few of them in a ReSpecT reaction has the potential to strongly impact on the semantics of a ReSpecT specification—and then, on the coordinative behaviour of a ReSpecT tuple centre.

B. *Monitored Motion*

The second example, while focussing on the expressiveness gain given by ReSpecT spatial extension like the previous one, is also meant to whow how the different features of ReSpecT – space-time awareness, situatedness, and programmability – can be combined to cope with complex problems, such as *monitored motion*.

We call "monitored motion" the problem in which a mobile device – e.g. a simple wheel-equipped robot – has to reach a given destination, which implies *(i)* to start moving when asked to do so; *(ii)* to monitor its own position so as to understand when given destination is reached, if obstacles are on its way, etc.; *(iii)* to finally stop when due. In the following ReSpecT specification – where the three reactions resemble the three different stages composing monitored motion – we explicitly monitor arrival to destination only, for the sake of simplicity.

More specifically, looking at the code below

- Reaction `1` reacts to movement requests by:
  - recording current position to compute the "movement vector" to follow, based on given destination;
  - sampling current time to schedule a monitoring reaction when specified in `TStep`—exploiting time awareness;
  - setting environmental properties, in particular, direction of motion and engine state of a `motion_dev` actuator on the hosting device (`Node`)—exploiting situatedness.
- Reaction `2` is responsible of motion monitoring:

- ○ first (predicate `check/2`), it perceives current position and given destination to understand if controlled device is arrived;
  - ○ if `check/2` returns `true`, the motion engine is turned off so to stop;
  - ○ if `check/2` returns `false`, the reaction re-schedules itself to continue monitoring.
- Reaction `3` does some clean-up when destination is reached, and spatial event `to(Dest)` is generated.

```
% 1) Compute motion vector then start moving.
  reaction( out(move(Dest,TStep)), completion,
    ( current_place(ph,Here), current_time(Now),
      Check is Now+TStep,
      direction(Dest,Here,Vec),      % Prolog computation
      out_s( % Reaction #2 ),        % Schedule monitoring
      current_target(_@Node),        % Get node id
      motion_dev@Node <- env(dir,Vec),    % Set direction
      motion_dev@Node <- env(engine,'on') )).  % Move on
% 2) Monitor destination arrival.
  reaction( time(Check), internal, % Time to check
    ( current_place(ph,Here),
      rd(move(Dest,TStep)),
      ( check(Here,Dest),            % Prolog computation
        current_node(Node),
        motion_dev@Node <- env(engine,'off')
        ;                            % 'if-then-else'
        current_time(Now), Check is Now+TStep,
        out_s( % Reaction #2 ) ) )).
% 3) Arrival clean-up.
  reaction( to(Dest),                % Destination reached
    internal, in(move(Dest,_)) ).
```

VI. Related Works

The need for coordination models and languages to support adaptiveness and reactiveness to the contingencies that may occur in the spatio-temporal fabric was recognised by several proposals in the literature, accounting for spatial issues and/or enforcing spatial properties.

$\sigma\tau$-LINDA [4] extends the LINDA model in three ways to enable the emergence of spatio-temporal patterns for tuples configuration in a distributed setting:

- tuples are replaced by "space-time activities", that is, processes manipulating the space-time configuration of tuples in the network;
- space operators `neigh`, `$distance` and `$orientation` are added to allow respectively to send broadcast messages to the neighborhood spaces, measure the distance toward any of them, devise the relative orientation of a target space w.r.t. the current one;
- time operators `next`, `$delay` and `$this` are added allowing (respectively) to delay an activity execution, measure the flow of time, access current coordination space identifier.

GEOLINDA [5], another LINDA extension, deals with spatial aspects by attempting to reflect physical spatial properties in a mobile tuple space setting:

- each tuple and each reading operation is associated to a geometrical volume (*addressing shape*);
- the semantics of reading operations is changed so as to unblock only when the shape of a matching tuple intersects with the addressing shape of the operation;
- each coordination space is associated to a communication range to allow detection of *incoming* and *outgoing* tuples/volumes.

The TOTA middleware [3] considers a distributed tuple space setting in which each tuple is equipped by two additional fields other than its content:

- a *propagation rule*, determining how the tuple should propagate and distribute across the network of linked tuple spaces—e.g. in terms of maximum number of hops, conditional rules upon the presence of other tuples, even how the tuple can change while moving
- a *maintenance rule*, dictating how the tuple should react to the flow of time and/or to spatial events

The combination of these rules makes it possible for TOTA to support self-organising *computational fields*, that is, distributed data structures – such as gradients – enforcing spatial properties in the configuration of tuples, eventually exploited by coordinating agents.

Finally, the SAPERE coordination model [1] is a chemical-inspired model for pervasive applications, enacting spatial computing patterns and gradient-based interaction [15].

Although the above coordination models deal with some spatial-related issues, they are mostly tailored to a specific problem or application domain, and lack of an exhaustive analysis of which are the basic mechanisms required to enable and promote "general-purpose" space-aware coordination.

VII. Conclusion & Future Works

In this paper we take the ReSpecT coordination media and language [8], and extend it with few basic constructs and mechanisms required to build space-aware coordination media able to deal with spatial issues.

Future work will be devoted to explore real-world case studies to put to test both the expressiveness and the effectiveness of the space-aware ReSpecT model in the coordination of complex systems, such as pervasive and adaptive ones. For instance, the recent availability of the TuCSoN middleware [16], exploiting ReSpecT tuple centres, upon Android devices makes it possible to actually experiment with spatial coordination in pervasive scenarios.

Acknowledgments

This work has been partially supported by the EU-FP7-FET Proactive project SAPERE – Self-aware Pervasive Service Ecosystems, under contract no. 256874.

References

[1] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo Serugendo, M. Risoldi, A.-E. Tchao, S. Dobson, G. Stevenson, Y. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally, "Self-aware pervasive service ecosystems," *Procedia Computer Science*, vol. 7, pp. 197–199, Dec. 2011, proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911005667

[2] J. Beal, O. Michel, and U. P. Schultz, "Spatial computing: Distributed systems that take advantage of our geometric world," *ACM Trans. Auton. Adapt. Syst.*, vol. 6, no. 2, pp. 11:1–11:3, Jun. 2011.

[3] M. Mamei and F. Zambonelli, "Programming pervasive and mobile computing applications: The tota approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 18, no. 4, pp. 15:1–15:56, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1538942.1538945

[4] M. Viroli, D. Pianini, and J. Beal, "Linda in space-time: an adaptive coordination model for mobile ad-hoc environments," in *Coordination Languages and Models*, ser. LNCS, M. Sirjani, Ed. Springer-Verlag, 14–15 Jun. 2012, vol. 7274, pp. 212–229, 14th Conference (Coordination 2012), Stockholm (Sweden).

[5] J. Pauty, P. Couderc, M. Banatre, and Y. Berbers, "Geo-Linda: a geometry aware distributed tuple space," in *Advanced Information Networking and Applications, 2007. AINA '07. 21st International Conference on*, may 2007, pp. 370 –377.

[6] A. Omicini and F. Zambonelli, "Coordination for Internet application development," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 3, pp. 251–269, Sep. 1999, special Issue: Coordination Mechanisms for Web Agents. [Online]. Available: http://springerlink.metapress.com/content/uk519681t1r38301/

[7] A. Omicini and E. Denti, "From tuple spaces to tuple centres," *Science of Computer Programming*, vol. 41, no. 3, pp. 277–294, Nov. 2001.

[8] A. Omicini, "Formal ReSpecT in the A&A perspective," *Electronic Notes in Theoretical Computer Science*, vol. 175, no. 2, pp. 97–117, Jun. 2007, 5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 Aug. 2006. Post-proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066107003519

[9] D. Gelernter, "Generative communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, Jan. 1985. [Online]. Available: http://portal.acm.org/citation.cfm?id=2433

[10] E. Denti, A. Natali, and A. Omicini, "Programmable coordination media," in *Coordination Languages and Models*, ser. LNCS, D. Garlan and D. Le Métayer, Eds. Springer-Verlag, 1997, vol. 1282, pp. 274–288, 2nd International Conference (COORDINATION'97), Berlin, Germany, 1–3 Sep. 1997. Proceedings. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-63383-9_86

[11] A. Omicini, A. Ricci, and M. Viroli, "Time-aware coordination in ReSpecT," in *Coordination Models and Languages*, ser. LNCS, J.-M. Jacquet and G. P. Picco, Eds. Springer-Verlag, Apr. 2005, vol. 3454, pp. 268–282, 7th International Conference (COORDINATION 2005), Namur, Belgium, 20–23 Apr. 2005. Proceedings. [Online]. Available: http://www.springerlink.com/content/kbcmbqed8jta590g/

[12] M. Casadei and A. Omicini, "Situated tuple centres in ReSpecT," in *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, S. Y. Shin, S. Ossowski, R. Menezes, and M. Viroli, Eds., vol. III. Honolulu, Hawai'i, USA: ACM, 8–12 Mar. 2009, pp. 1361–1368. [Online]. Available: http://portal.acm.org/citation.cfm?id=1529282.1529586

[13] A. Omicini and E. Denti, "Formal ReSpecT," *Electronic Notes in Theoretical Computer Science*, vol. 48, pp. 179–196, Jun. 2001, declarative Programming – Selected Papers from AGP 2000, La Habana, Cuba, 4–6 Dec. 2000.

[14] A. Omicini, "Nature-inspired coordination models: Current status, future trends," *ISRN Software Engineering*, vol. 2013, 2013, article ID 384903, Review Article. [Online]. Available: http://www.hindawi.com/isrn/se/2013/384903/

[15] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli, "Spatial coordination of pervasive services through chemical-inspired tuple spaces," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 6, no. 2, pp. 14:1–14:24, Jun. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1968513.1968517

[16] TuCSoN, "Home page," `http://tucson.unibo.it/`, 2013.

Tuple-based Coordination of Stochastic Systems with Uniform Primitives

Stefano Mariani
DISI, ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: s.mariani@unibo.it

Andrea Omicini
DISI, ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: andrea.omicini@unibo.it

***Abstract*—Complex computational systems – such as pervasive, adaptive, and self-organising ones – typically rely on simple yet expressive coordination mechanisms: this is why coordination models and languages can be exploited as the sources of the essential abstractions and mechanisms to build such systems. While the features of tuple-based models make them well suited for complex system coordination, they lack the probabilistic mechanisms for modelling the stochastic behaviours typically required by adaptivity and self-organisation.**

To this end, in this paper we explicitly introduce *uniform primitives* as a probabilistic specialisation of standard tuple-based coordination primitives, replacing don't know non-determinism with uniform distribution. We define their semantics and discuss their expressiveness and their impact on system predictability.

I. INTRODUCTION

While computational systems grow in complexity, coordination models and technologies are more and more essential to harness the intricacies of intra- and inter-system interaction [1], [2]. In particular, tuple-based coordination models – derived from the original LINDA [3] – have shown their power in the coordination of pervasive, adaptive, and self-organising systems [4], such as SAPERE [5] and MoK [6].

A foremost feature of computational models for open, adaptive and self-* systems is *non-determinism*. LINDA features *don't know* non-determinism in the access to tuples in tuple spaces, handled with a *don't care* approach: *(i)* a tuple space is a multiset of tuples where multiple tuples possibly match a given template; *(ii)* which tuple among the matching ones is actually retrieved by a *getter* operation (`in`, `rd`) can be neither specified nor predicted (*don't know*); *(iii)* nonetheless, the coordinated system is designed so as to keep on working whichever is the matching tuple returned (*don't care*).

The latter assumption requires that when a process uses a template matching multiple tuples, which specific tuple is actually retrieved is not relevant for that process. This is not the case, however, in many of today adaptive and self-organising systems, where processes may need to implement *stochastic behaviours* like "most of the time do this" or "not always do that"—which obviously do not cope well with don't know non-determinism. For instance, all the nature-inspired models and systems emerged in the last decade – such as chemical, biochemical, stigmergic, and field-based – are examples of the broad class of *self-organising* systems that precisely require such a sort of behaviour [7]—which by no means can be enabled by the canonical LINDA model and its direct derivatives.

To this end, in this paper we define *uniform coordination primitives* (`uin`, `urd`) – first mentioned in [8] – as the specialisation of LINDA *getter* primitives featuring *probabilistic non-determinism* instead of don't know non-determinism. Roughly speaking, uniform primitives allow programmers to both specify and (*statistically*) *predict* the probability to retrieve one specific tuple among a bag of matching tuples, thus making it possible to statistically control non-deterministic systems.

Accordingly, in this paper we first define uniform primitives based on the probabilistic framework from [9] (Section II), then demonstrate their expressive power both formally – by exploiting *probabilistic modular embedding* [10] – and by discussing some examples (Section III). Finally, we compare uniform primitives with other approaches in probabilistic and stochastic coordination (Section IV).

II. UNIFORM PRIMITIVES

LINDA *getter* primitives – that is, data-retrieval primitives `in` and `rd` – are shared by all tuple-based coordination models, and provide them with *don't know non-determinism*: when one or more tuples in a tuple space match a given template, *any* of the matching tuples can be non-deterministically returned.

In a single getter operation, only a *point-wise* property affects tuple retrieval: that is, the conformance of a tuple to the template, independently of the *spatial* context—namely, the other tuples in the same space. Furthermore, in a sequence of getter operations, don't know non-determinism makes any prediction of the overall behaviour impossible: e.g., reading one thousand times the same template in a tuple space with ten matching tuples could possibly lead to retrieving the same tuple all times, or one hundred times each, or whatever admissible combination one could think of—no prediction possible, according to the model. Again, then, only a point-wise property can be ensured even in *time*: that is, only the mere compliance to the model of each individual operation in the sequence.

Instead, uniform primitives enrich tuple-based coordination models with the ability of performing operations that ensure *global* system properties instead of point-wise ones, both in space and in time. More precisely, uniform primitives replace don't know non-determinism with *probabilistic non-determinism* to *situate* a primitive invocation in space – the tuple actually retrieved depends on the other tuples in the space – and to *predict* its behaviour in time — statistically, the distribution of the tuples retrieved will tend to be uniform, over time.

Whereas exploiting probabilistic non-determinism to its full extent would lead to the definition of the complete set of uniform coordination primitives – including, e.g., `uinp` and `urdp` primitives –, here we aim at understanding the fundamental mechanisms making tuple-based models well suited for complex system coordination, by enhancing them with the probabilistic mechanisms for modelling the stochastic behaviours typically required by adaptivity and self-organisation. Accordingly, in this paper we focus only on the two uniform primitives (`uin`, `urd`) that specialise the basic LINDA getter primitives. In the remainder of this section, first (Subsection II-A) we define them informally, then (Subsection II-B) we provide them with a formal semantic specification according to the probabilistic framework defined in [9].

A. Informal semantics

The main motivation behind uniform primitives is to introduce a *simple* yet *expressive* probabilistic mechanism in tuple-based coordination: simple enough to work as a specialisation of standard LINDA operations, expressive enough to model the most relevant stochastic behaviours of complex computational systems such as adaptive and self-organising ones.

Whereas expressiveness is discussed in Section III, simplicity is achieved by defining uniform primitives as specialised versions of standard LINDA primitives: so, first of all, `uin` and `urd` are compliant with the standard semantics of `in` and `rd`. In the same way as `in` and `rd`, `uin` and `urd` ask tuple spaces for one tuple matching a given template, suspend when no matching tuple is available, return a matching tuple chosen non-deterministically when one or more matching tuples are available in the tuple space. As a straightforward consequence, any tuple-based coordination system working with `in` and `rd` would also work by using instead `uin` and `urd`, respectively—and any process using `in` and `rd` could adopt `uin` and `urd` instead without any further change.

On the other hand, the nature of the specialisation lays precisely in the way in which a tuple is non-deterministically chosen among the (possibly) many tuples matching the template. While in standard LINDA the choice is performed based on don't know non-determinism, uniform primitives exploit instead *probabilistic non-determinism* with *uniform distribution*. So, if a standard getter primitive requires a tuple with template T, and m tuples $t_1, .., t_m$ matching T are in the tuple space when the request is executed, any tuple $t_{i \in \{1..m\}}$ could be retrieved, but nothing more could be said—no other assertion is possible about the result of the getter operation. Instead, when a uniform getter primitive requires a tuple with template T, and m tuples $t_1, .., t_m$ matching T are available in the tuple space when the request is served, one assertion is possible about the result of the getter operation: that is, each of the m matching tuples $t_{i \in \{1..m\}}$ has exactly the same probability $1/m$ to be returned. So, for instance, if 2 `colour(blue)` and 3 `colour(red)` tuples occur in the tuple space when a `urd(colour(X))` is executed, the probability of the tuple retrieved to be `colour(blue)` or `colour(red)`) is exactly 40% or 60%, respectively.

Operationally, uniform primitives behave as follows. When executed, a uniform primitive takes a *snapshot* of the tuple space, "freezing" its state at a certain point in time—and space, being a single tuple space the target of basic LINDA primitives. The snapshot is then exploited to assign a probabilistic value $p_i \in [0, 1]$ to any tuple $t_{i \in \{1..n\}}$ in the space—where n is the total number of tuples in the space. There, non-matching tuples have value $p = 0$, matching tuples have value $p = 1/m$ (where $m \leq n$ is the number of matching tuples), and the overall sum of probability values is $\sum_{i=1..n} p_i = 1$. The choice of the matching tuple to be returned is then statistically based on the computed probabilistic values.

As a consequence, while standard getter primitives exhibit point-wise properties only, uniform primitives feature *global* properties, both in space and time. In terms of spatial context, in fact, standard getter primitives can return a matching tuple independently of the other tuples currently in the same space—so, they are "context unaware". Instead, uniform getter primitives return matching tuples based on the overall state of the tuple space—so, their behaviour is *context aware*. In terms of time, too, sequences of standard getter operations present no meaningful properties. Instead, by definition, sequences of uniform getter operations tend to globally exhibit a uniform distribution over time. So, for instance, performing N `urd(colour(X))` operations over a tuple space containing 10 `colour(white)` and 100 `colour(black)` tuples, leads to a sequence of returned tuples which, while N grows, would tend to contain ten times more `colour(black)` tuples than `colour(white)` ones.

B. Formal semantics

In order to define the semantics of (getter) uniform primitives, we rely upon a simplified version of the process-algebraic framework in [9], dropping multi-level priority probabilities. In detail, we exploit closure operator $\uparrow$, handles h, and closure term G as follows:

(i) handles coupled to actions (open transitions) represent tuple templates associated with primitives;

(ii) handles listed in closure term G represent tuples offered (as synchronisation items) by the tuple space (modelled as a process);

(iii) closure term G associates handles (tuples) with their cardinality in the tuple space;

(iv) closure operator $\uparrow$ *(a)* matches admissible synchronisations between processes and the tuple space, and *(b)* computes their associated probability distribution based upon handle-associated values.

It is worth to note that closure operator $\uparrow$ could be seen as following our statistical interpretation of a uniform primitive: it takes a snapshot of the tuple space state – *matching*, step *(a)* – then samples it probabilistically — *sampling*, step *(b)*.

1) Semantics of `uin` (uniform consumption): Three transition rules define the operational semantics of the `uin` primitive for *uniform consumption*:

[SYNCH-C] Open transition representing the request for process-space synchronisation upon template T, which leads to the snapshot:

$$\texttt{uin}_T.P \mid \langle t_1,..,t_n\rangle \xrightarrow{T} \texttt{uin}_T.P \mid \langle t_1,..,t_n\rangle \uparrow \{(t_1,v_1),..,(t_n,v_n)\}$$

where $v_{i=1..n} = \mu(T,t_i)$, and $\mu(\cdot,\cdot)$ is the standard matching function of LINDA, hence $\forall i, v_i ::= 1 \mid 0$.

[CLOSE-C] Closed unlabelled transition (reduction) representing the internal computation assigning probabilities to synchronisation items (uniform distribution computation):

$$\texttt{uin}_T.P \mid \langle t_1,..,t_n\rangle \uparrow \{(t_1,v_1),..,(t_n,v_n)\} \hookrightarrow \texttt{uin}_T.P \mid \langle t_1,..,t_n\rangle \uparrow \{(t_1,p_1),..,(t_n,p_n)\}$$

where $p_j = \frac{v_j}{\sum_{i=1}^{n} v_i}$ is the absolute probability of retrieving tuple t_j, with $j = 1..n$.

[EXEC-C] Open transition representing the probabilistic response to the requested synchronisation (the sampling):

$$\texttt{uin}_T.P \mid \langle t_1,..,t_n\rangle \uparrow \{..,(t_j,p_j),..\} \xrightarrow{t_j}_{p_j} P[t_j/T] \mid \langle t_1,..,t_n\rangle \backslash t_j$$

where $[\cdot/\cdot]$ represents term substitution in process P continuation, and $\backslash$ is multiset difference, expressing removal of tuple t_j from the tuple space.

2) Semantics of `urd` *(uniform reading):* As for standard LINDA getter primitives, the only difference between *uniform reading* (`urd`) and uniform consumption (`uin`) is the non-destructive semantics of the reading primitive `urd`. This is reflected by EXEC-R open transition:

[EXEC-R] The same as EXEC-C, except for the fact that it does not remove matching tuple

$$\texttt{urd}_T.P \mid \langle t_1,..,t_n\rangle \uparrow \{..,(t_j,p_j),..\} \xrightarrow{t_j}_{p_j} P[t_j/T] \mid \langle t_1,..,t_n\rangle$$

whereas other transitions are left unchanged.

3) Example: As an example, in the following system state

$$\texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle$$

where $\mu(T,tx)$ holds for $x = a,b,c$, the following synchronisation transitions are enabled:

(a) $\texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \xrightarrow{ta}_{0.5} P[ta/T] \mid \langle ta,tb,tc\rangle$

(b) $\texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \xrightarrow{tb}_{0.25} P[tb/T] \mid \langle ta,ta,tc\rangle$

(c) $\texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \xrightarrow{tc}_{0.25} P[tc/T] \mid \langle ta,ta,tb\rangle$

For instance, if transition *(a)* wins the probabilistic selection, then the system evolves according to the following trace—simplified by summing up cardinalities and probabilities in order to enhance readability:

$$\texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \xrightarrow{T} \texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \uparrow \{(ta,2),(tb,1),(tc,1)\}$$
$$\hookrightarrow \texttt{uin}_T.P \mid \langle ta,ta,tb,tc\rangle \uparrow \{(ta,\tfrac{1}{2}),(tb,\tfrac{1}{4}),(tc,\tfrac{1}{4})\} \xrightarrow{ta}_{\frac{1}{2}} P[ta/T] \mid \langle ta,tb,tc\rangle$$

III. EXPRESSIVENESS

In [11], authors demonstrate that LINDA-based languages cannot implement probabilistic models: a LINDA process calculus, although Turing-complete, is not expressive enough to express probabilistic choice [11]. In our specific case, the gain of expressiveness is formally proven in [12], where uniform primitives are formally proven to be strictly more expressive than standard LINDA coordination primitives by exploiting *probabilistic modular embedding* (PME) [10], an extension to *modular embedding* [13] explicitly meant to capture the expressiveness of stochastic systems.

In particular, if we denote with uLINDA the LINDA coordination model where standard getter primitives `rd` and `in` are replaced with uniform getter primitives `urd` and `uin`, then uLINDA is proven to be strictly *more expressive* than LINDA according to PME, since uLINDA *probabilistically embeds* ($\succeq_p$) LINDA, but not the other way around—so that formally, according to PME, LINDA and uLINDA are not observationally equivalent ($\not\equiv_o$):

$$\text{uLINDA} \succeq_p \text{LINDA},\ \text{LINDA} \not\succeq_p \text{uLINDA} \implies \text{uLINDA} \not\equiv_o \text{LINDA}$$

Since formally asserting a gap in expressiveness does not necessarily make it easy for the reader to fully appreciate how much this can make the difference for adaptive and self-organising systems, in the remainder of this section we discuss two examples showing how uniform primitives make it possible to *(i)* have some self-organising property appear by emergence (Subsection III-A), and *(ii)* straightforwardly design stochastic systems reproducing some simple yet meaningful nature-inspired behavioural pattern, such as pheromone-based coordination (Subsection III-B).

```
LogicTuple templ;
while(!die){
  templ = LogicTuple.parse("ad(S)");
  // Pick a server probabilistically
  op = acc.urd(tid, templ, null);
  // Plain Linda version
  // op = acc.rd(tid, templ, null);
  if (op.isResultSuccess()) {
    service = op.getLogicTupleResult();
    // Submit request
    req = LogicTuple.parse(
      "req("+service.getArg(0)+","+reqID+")"
      );
    acc.out(tid, req, null);
  }
}
```

Fig. 1. Java code for clients looking for services.

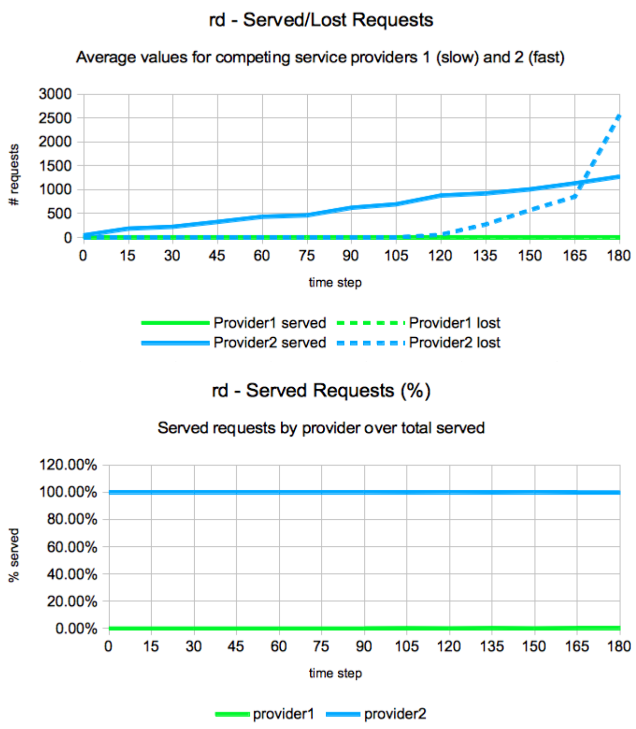


Fig. 2. Clients using `rd` primitive: service provider 1 is under-exploited.

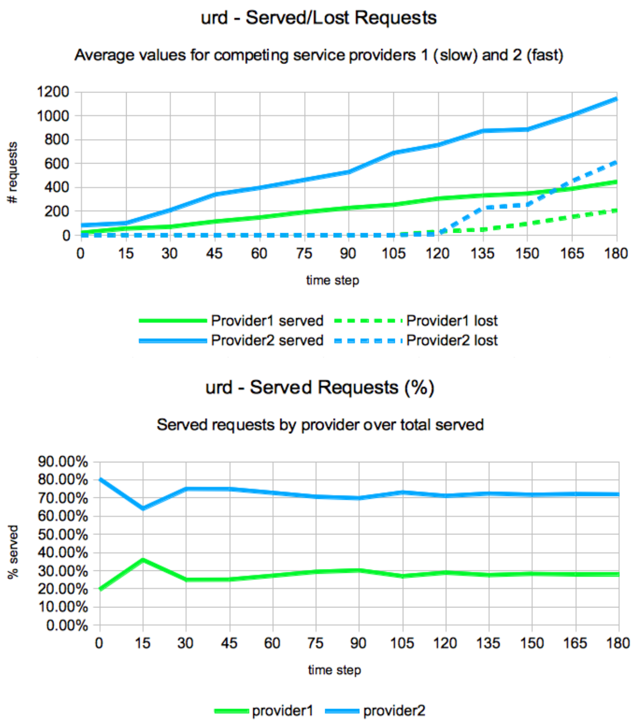


Fig. 3. Clients using `urd` primitive: a certain degree of *fairness* is guaranteed, based on self-organisation.

A. Load Balancing

In order to better explain what the "basic mechanisms enabling self-organising coordination" actually are – that is, a minimal construct able (alone) to impact the observable properties of a coordinated system – we discuss the following scenario: two service providers are both offering the same service to clients – through proper "advertising tuples" –; the first is slower than the second, that is, it needs more time to process a request—thus modelling differences in computational power.

Their working cycle is quite simple: a worker thread gets requests from a shared tuple space, then puts them in the master thread (the actual service provider) bounded queue. The master thread continuously polls the queue looking for requests to serve: when one is found, it is served, then the master emits another advertising tuple; if none is found, the master does something else, then re-polls the queue—no advertising is done. The decoupling enforced by the queue is useful to model the fact that service providers should not block on the space waiting for incoming requests, so as to be free of performing other jobs meanwhile—e.g. reporting, resource clean-up, etc. The queue is bounded to model memory constraints.

In this setting, clients (whose Java code is listed in Fig. 1) search for available services first via `rd` primitive (Fig. 2), then via `urd` (Fig. 3). All charts' values are not single runs, but average values resulting from different runs—e.g., value plotted at time step 60 is not that of a single run, but the average of the number of requests observable at time step 60 of a number of runs (actually, 30).

By using the `rd` primitive we *blindly commit* to the actual implementation of the LINDA model currently at hand. For instance, Fig. 2 gives some hints about the implementation used for our simulation – the TuCSoN coordination middleware [14], [15] –: since provider 1 is almost unused, we understand that `rd` is implemented as a FIFO queue, always matching the first tuple among many ones—provider 2 advertising tuple, in this case. The point here is that such a prediction was not possible prior to the simulation, and with no information about the actual LINDA implementation used.

By using primitive `urd` instead (Fig. 3), we know – and can predict – how much each service provider will be exploited by clients: since we know by design that after successfully serving a request a provider emits an advertising tuple, and that such tuples are those looked for by clients, we know that the faster provider will produce more tuples, hence it will be more frequently found than the slower one. From Fig. 3 charts, in fact, we can see how the system of competing service providers self-organises by splitting incoming requests. Furthermore, such split is not statically designed or superimposed, but results by *emergence* from a number of run-time factors, such as clients interactions, service providers computational load, computational power, and memory. It should also be noted that such form of *load balancing* is not the only benefit gained when using `urd` over `rd`: actually, the `urd` simulation successfully serves $\simeq$ 1600 requests – distributed among providers 1 and 2 according to uniform primitive semantics – losing $\simeq$ 600, whereas the `rd` simulation serves successfully $\simeq$ 1250 – leaving provider 1 unused – losing over 2500.

B. Pheromone-based coordination

In *pheromone-based coordination* used by ants to find optimal paths – as well as by many ant-inspired computational systems, such as [16], [17] – each agent basically wanders

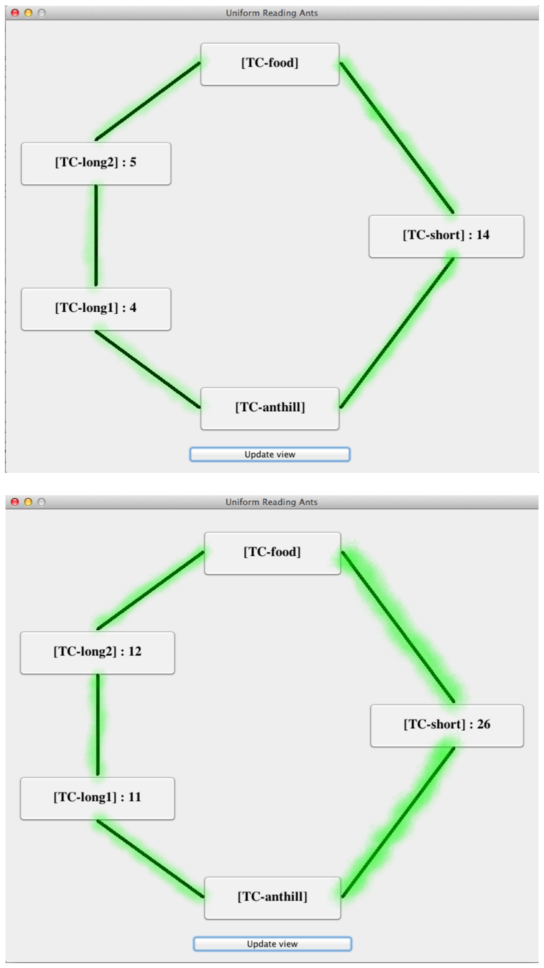


Fig. 4. Digital ants search food *(top box)* wandering randomly from their anthill *(bottom box)*.

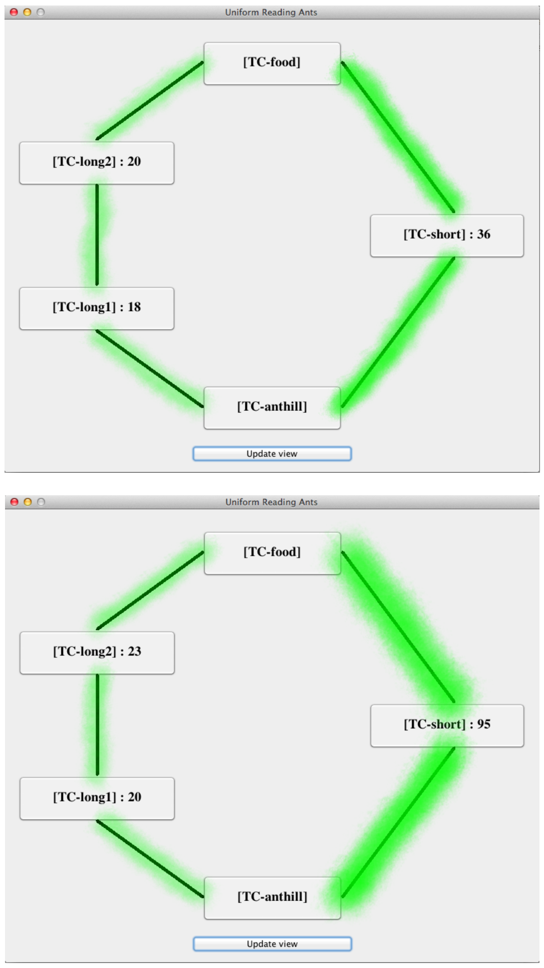


Fig. 5. By `urd`-ing digital pheromones left while carrying food, digital ants stochastically find the optimal path toward the food source.

randomly through the network until it finds a pheromone trail, which the agent is *likely* to follow based on the trail "strength".

Here, aspects such as pheromone release, scent, and evaporation [16] are not relevant: instead, the above-mentioned notions of "randomness" and "likelihood" are on the one hand *essential* for pheromone-based coordination, on the other hand *require* uniform primitives to be designed using a tuple-based coordination model. In particular, we consider a network of n nodes representing places p_i, with $i = 1..n$, through which ant agents walk. The default tuple space in node p_i contains at least one *neighbour* tuple $\mathtt{n(}p_j\mathtt{)}$ for each neighbour node p_j and the neighbourhood relation is reflexive—so, if node p_i and p_j are neighbours, p_i tuple space contains tuple $\mathtt{n(}p_j\mathtt{)}$ and p_j tuple space contains tuple $\mathtt{n(}p_i\mathtt{)}$. Pheromone deposit in node p_i is modelled by the insertion of a new tuple $\mathtt{n(}p_i\mathtt{)}$ in every neighbour node p_i.

Thus, ants wandering through places and ants following trails can both be easily modelled using uniform primitives: ant agents just need to look locally for neighbour tuples through a `urd(n(P))`. If no pheromone trail is to be detected nearby, every neighbour place is represented by a single tuple, so all neighbour places have the same probability to be chosen—thus leading to random wandering of ants. In case some of the neighbours contains a detectable trail, the corresponding neighbour tuple occurs more than once in the local tuple space: so, by using uniform primitives, the tuple corresponding to a neighbour place with a pheromone trail has a greater probability to be chosen than the others.

For instance, say `p1`, `p2`, `p3` are neighbour places. Without a pheromone trail, an ant in `p1` moves to either `p2` or `p3` with the same probability, starting from the following system state:

$$\mathtt{urd(n(X))}.P \mid \langle \mathtt{n(p2)}, \mathtt{n(p3)} \rangle$$

There, the enabled synchronisation transitions are

(a) $\mathtt{urd(n(X))}.P \mid \langle \mathtt{n(p2)}, \mathtt{n(p3)} \rangle \xrightarrow{\mathtt{n(p2)}}_{0.5} P[\mathtt{p2/X}] \mid \langle \mathtt{n(p2)}, \mathtt{n(p3)} \rangle$

(b) $\mathtt{urd(n(X))}.P \mid \langle \mathtt{n(p2)}, \mathtt{n(p3)} \rangle \xrightarrow{\mathtt{n(p3)}}_{0.5} P[\mathtt{p3/X}] \mid \langle \mathtt{n(p2)}, \mathtt{n(p3)} \rangle$

that is, an ant agent in `p1` has the same probability (50%) to move to either `p2` or `p3`—which exactly models random ant wandering.

Instead, if a pheromone trail involves p3 – so that for instance p1 contains 2 tuples n(p3) – the initial system state would be

$$\texttt{urd(n(X))}.P \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle$$

There, the enabled synchronisation transitions are

(c) $\texttt{urd(n(X))}.P \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle \xrightarrow{\texttt{n(p2)}}_{0.\overline{3}} P[\texttt{p2/X}] \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle$

(d) $\texttt{urd(n(X))}.P \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle \xrightarrow{\texttt{n(p3)}}_{0.\overline{6}} P[\texttt{p3/X}] \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle$

which exactly models the fact that the ant agent in p1 is more likely to move to p3 than to p2, thus (probabilistically) following the pheromone trail.

A crucial point, here, is to understand the issue of system *predictability* with / without uniform primitives. Reachable states for the system above would not change by replacing urd with rd: the transitions above would work in the same way *apart from* probabilistic labelling. This essentially means that a standard LINDA coordinated system would potentially reach the same states as the one with uniform primitives: the point is, nevertheless, that quantitative information would be available for the latter system, not for the former.

In particular, in the second example above, the reachable states are *(c)* $P[\texttt{p2/X}] \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle$ and *(d)* $P[\texttt{p3/X}] \mid \langle \texttt{n(p2),n(p3),n(p3)} \rangle$. Using urd, we know that states *(c)* and *(d)* would be reached with probability $.\overline{3}$ and $.\overline{6}$, respectively: so, both a probabilistic prediction on the single system run, and a statistic prediction over multiple system runs are made possible by the use of uniform primitives. The usage of rd, instead, allows for nothing similar: we just know that both states *(c)* and *(d)* could be reached, but no quantitative predictions of any sort are possible.

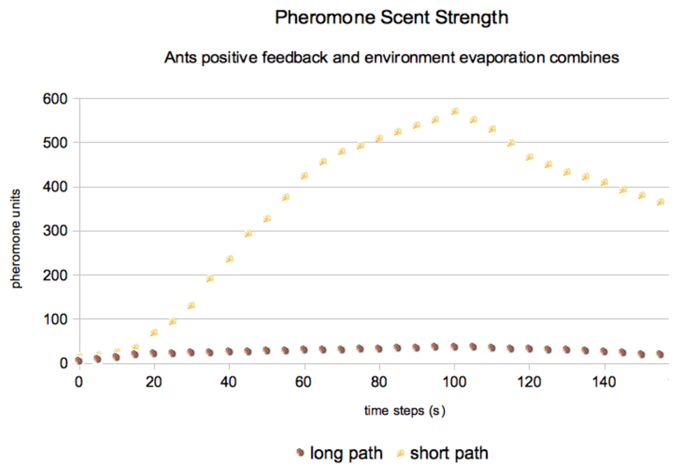


Fig. 6. Pheromones strength across time. Descending phase corresponds to food depletion in food tuple centre: no new pheromones added, evaporation makes strength decline.

Our experiments are conducted in a toy scenario involving digital ants and pheromones programmed in ReSpecT [18] upon the TuCSoN coordination middleware [14]. The experiment involves ten digital ants starting from the anthill with the goal of finding food, and follows the "canonical" assumptions of ant systems. So, at the beginning, any path has equal probability of being chosen, thus modelling random walking of ants in absence of pheromone. As ants begin to wander around, eventually they find food, and release pheromone on their path while coming back home. As a consequence, the shortest path eventually gets more pheromone since it takes less time to travel on it rather than on the longest path. Pheromones as well as connections between tuple centres are modelled as described above, with "neighbour" tuples: the more neighbour tuples of a certain type, the more likely ants will move to that neighbour tuple centre with their next step.

Fig. 4 and Fig. 5 depict a few screenshots of our toy scenario: there, five distributed tuple centres (the larger boxes) model a topology connecting the anthill *(bottom box)* to a food source *(top box)*: the leftmost path is longer, whereas the rightmost is shorter. The green "spray-like" effect on paths (black lines) models the strength of the pheromone scent: the greater and greener the path, the more pheromones lay on it.

By plotting pheromones strength evolution over time, Fig. 6 simply shows how our expectations about digital ants behaviour are met: in fact, despite starting from the situation in which any path is equi-probable (the amount of pheromones on the shortest path is the same as on the longest path), eventually the system detects the shortest path, which becomes the most exploited—and contains in fact more pheromone units.

In the Java code describing the behaviour of ants (Fig. 7), in particular in method smellPheromone() (line 10), usage of the uniform primitive urd is visible on line 27, whereas line 29 shows the tuple template given as its argument, that is, n(NBR): at runtime, NBR unifies with a TuCSoN tuple centre identifier, making it possible for the ant to move there. Quite obviously, the idea here is not just showing a new way to model ant-like systems. Instead, the example above is meant to point out how a non-trivial behaviour – that is, dynamically solving a shortest path problem – can be achieved by simply substituting uniform primitives to traditional LINDA getter primitives—which instead would not allow the system to work as required. Furthermore, the solution is adaptive, fully distributed, and based upon local information solely – thus, it appears by *emergence* –, and robust against topology changes—a ReSpecT specification implementing evaporation was used, although not shown for the lack of space.

IV. RELATED WORKS

Uniform primitives were first used in [19] as a tool for solving a specific coordination problem, called *collective sort*: however, neither there, nor in subsequent papers [20], [8], they were given but a few lines of informal definition, and their general role in the coordination of complex computational systems was not yet clarified.

In [21], similar primitives are presented and formally defined to forge the *biochemical tuple space* notion, leading a tuple space to act as a chemical simulator. There, tuples are enriched with an *activity/pertinency value* – similarly to the quantitative information defined in [11] – to resemble chemical concentrations, therefore LINDA primitives are necessarily refined with the ability to consider such numerical label. So, the main point of difference w.r.t. therein defined primitives is that *(i)* here we rely on tuples multiplicity to model probability, leaving the LINDA tuples structure untouched, *(ii)* uniform primitives are scheduled and executed as LINDA classical getter primitives, while in [21] their primitives have a *stochastic rate of execution* equipped.

To the best of our knowledge, proposals presented to extend LINDA with probabilities follow two main approaches [22]:

- *data-driven* models, where the quantitative information required to model probability is associated with the data items – the tuples – in the form of *weights*. This approach is adopted in `ProbLinCa` [11], the probabilistic version of a LINDA-based process calculus.
- *schedule-driven* models, where the quantitative information is added to the processes using special "probabilistic schedulers". This is the approach taken by [22] to define a probabilistic extension of the KLAIM model named PKLAIM.

Instead, our approach belongs to a third, novel category – which we call *interaction-driven* – where probabilistic behaviour is *(i)* associated to communication primitives – thus, neither to processes (or schedulers), nor to tuples – and *(ii)* enacted during the interaction between a process and the coordination medium—that is, solely through such primitives.

Also, uniform primitives can be seen as complementary to both the approaches taken in `ProbLinCa` and PKLAIM, where the basic LINDA model is changed quite deeply. Uniform primitives, instead, extend LINDA by *specialising* standard LINDA primitives, without changing neither tuple structure nor scheduling policy. Furthermore, uniform primitives could be used to emulate both approaches: tuple weights could be reified by their multiplicity in the space, whereas probabilistic scheduling could be obtained by properly synchronising processes upon probabilistic consumption of shared tuples. Moreover, uniform coordination primitives could be used in place of LINDA standard ones without affecting the model, merely refining don't care non-determinism as probabilistic non-determinism: as a result, all the expressiveness results and all the applications based on the canonical LINDA model do still hold using `uin` and `urd` instead of `in` and `rd`.

More complex coordination models exist in literature for which uniform primitives could play a key role in providing the probabilistic mechanisms required for the engineering of stochastic systems like adaptive and self-organising ones.

STOKLAIM [23] is an extension to KLAIM in which process actions are equipped with *rates* affecting execution probability, and execution *delays* as well—that is, time needed to carry out an action. By reifying action rates as tuples in the space, with multiplicity proportional to rates, uniform-reading such tuples would allow to probabilistically schedule actions' execution *à la* STOKLAIM. Furthermore, delays could be emulated, too, by uniform-reading a set of "time tuples", where a higher value corresponds to a lower action rate.

SAPERE [5] is a biochemically-inspired model for the engineering of complex self-organising and adaptive *pervasive service ecosystems*, where agents share *LSAs* (Live Semantic Annotation), which could be thought of as a special kind of tuples, representing them in shared contexts, and allowing them to interact and pursue their own goals. LSAs are managed through *eco-laws*, which are some sort of chemical-like rules, scheduled according to their rates Hence, uniform primitives could play in SAPERE the same role as in STOKLAIM—once eco-laws are reified as tuples with a multiplicity proportional to execution rate. Furthermore, from the pool of all LSAs which can participate in a eco-law, the ones actually consumed by the law – as *chemical reactants* – are selected probabilistically. Once again, such behaviour could be enabled by uniform consumption of reactant LSAs in eco-laws.

V. CONCLUSION

In this paper we formally define *uniform primitives* as simple specialisation of standard LINDA coordination primitives, exploiting probabilistic non-determinism in place of don't know non-determinism. We argue that uniform primitives introduce a simple yet powerful mechanism enhancing tuple-based coordination with the ability to express and predict stochastic behaviours, thus to design complex coordinated systems featuring adaptiveness and self-organisation.

```
while (!stopped) {
  if (!carryingFood) {
    // If not carrying food
    isFood = smellFood();
    if (isFood) {
      // pick up food if any
      pickFood();
    } else {
      // or stochastically follow pheromone
      direction = smellPheromone();
      move(direction);
    }
  } else {
    // If carrying food
    if (isAnthill()) {
      // drop food if in anthill
      dropFood();
    } else {
      // or move toward anthill
      direction = smellAnthill();
      move(direction);
    }
  }
}

private LogicTuple smellPheromone() {
  ITucsonOperation op = acc.urd(
    tcid,
    LogicTuple.parse("n(NBR)"),
    TIMEOUT
  );
  if (op.isResultSuccess()) {
    return op.getLogicTupleResult();
  }
}
```

Fig. 7. Java code for ants.

Acknowledgments

This work has been supported by the EU-FP7-FET Proactive project SAPERE – Self-aware Pervasive Service Ecosystems, under contract no. 256874.

References

[1] D. Gelernter and N. Carriero, "Coordination languages and their significance," *Communications of the ACM*, vol. 35, no. 2, pp. 97–107, 1992. [Online]. Available: http://dx.doi.org/10.1145/129630.129635

[2] P. Wegner, "Why interaction is more powerful than algorithms," *Communications of the ACM*, vol. 40, no. 5, pp. 80–91, May 1997. [Online]. Available: http://dx.doi.org/10.1145/253769.253801

[3] D. Gelernter, "Generative communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, Jan. 1985. [Online]. Available: http://dx.doi.org/10.1145/2363.2433

[4] A. Omicini and M. Viroli, "Coordination models and languages: From parallel computing to self-organisation," *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 53–59, Mar. 2011, special Issue 01 (25th Anniversary Issue). [Online]. Available: http://dx.doi.org/10.1017/S026988891000041X

[5] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo, M. Risoldi, A.-E. Tchao, S. Dobson, G. Stevenson, Y. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally, "Self-aware pervasive service ecosystems," *Procedia Computer Science*, vol. 7, pp. 197–199, Dec. 2011, proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11). [Online]. Available: http://dx.doi.org/10.1016/j.procs.2011.09.006

[6] S. Mariani and A. Omicini, "Molecules of Knowledge: Self-organisation in knowledge-intensive environments," in *Intelligent Distributed Computing VI*, ser. Studies in Computational Intelligence, G. Fortino, C. Bădică, M. Malgeri, and R. Unland, Eds., vol. 446. Springer, 2013, pp. 17–22, 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24-26 Sep. 2012. Proceedings. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32524-3_4

[7] A. Omicini, "Nature-inspired coordination models: Current status, future trends," *ISRN Software Engineering*, vol. 2013, 2013, article ID 384903, Review Article. [Online]. Available: http://dx.doi.org/10.1155/2013/384903

[8] L. Gardelli, M. Viroli, M. Casadei, and A. Omicini, "Designing self-organising MAS environments: The collective sort case," in *Environments for MultiAgent Systems III*, ser. LNAI, D. Weyns, H. V. D. Parunak, and F. Michel, Eds. Springer, May 2007, vol. 4389, pp. 254–271, 3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71103-2_15

[9] M. Bravetti, "Expressing priorities and external probabilities in process algebra via mixed open/closed systems," *Electronic Notes in Theoretical Computer Science*, vol. 194, no. 2, pp. 31–57, 16 Jan. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2007.11.003

[10] S. Mariani and A. Omicini, "Probabilistic modular embedding for stochastic coordinated systems," in *Coordination Models and Languages*, ser. LNCS, C. Julien and R. De Nicola, Eds. Springer, 2013, vol. 7890, pp. 151–165, 15th International Conference (COORDINATION 2013), Florence, Italy, 3–6 Jun. 2013. Proceedings. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38493-6_11

[11] M. Bravetti, R. Gorrieri, R. Lucchi, and G. Zavattaro, "Quantitative information in the tuple space coordination model," *Theoretical Computer Science*, vol. 346, no. 1, pp. 28–57, 23 Nov. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2005.08.004

[12] S. Mariani and A. Omicini, "Probabilistic embedding: Experiments with tuple-based probabilistic languages," in *28th ACM Symposium on Applied Computing (SAC 2013)*, Coimbra, Portugal, 18–22 Mar. 2013, pp. 1380–1382, Poster Paper. [Online]. Available: http://dx.doi.org/10.1145/2480362.2480621

[13] F. S. de Boer and C. Palamidessi, "Embedding as a tool for language comparison," *Information and Computation*, vol. 108, no. 1, pp. 128–157, 1994. [Online]. Available: http://dx.doi.org/10.1006/inco.1994.1004

[14] A. Omicini and F. Zambonelli, "Coordination for Internet application development," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 3, pp. 251–269, Sep. 1999, special Issue: Coordination Mechanisms for Web Agents. [Online]. Available: http://dx.doi.org/10.1023/A:1010060322135

[15] TuCSoN, "Home page," 2013. [Online]. Available: http://tucson.unibo.it

[16] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, Jul. 2004. [Online]. Available: http://mitpress.mit.edu/books/ant-colony-optimization

[17] H. V. D. Parunak, S. Brueckner, and J. Sauter, "Digital pheromone mechanisms for coordination of unmanned vehicles," in *1st International Joint Conference on Autonomous Agents and Multiagent systems*, C. Castelfranchi and W. L. Johnson, Eds., vol. 1. New York, NY, USA: ACM, 15–19 Jul. 2002, pp. 449–450. [Online]. Available: http://dx.doi.org/10.1145/544741.544843

[18] A. Omicini and E. Denti, "From tuple spaces to tuple centres," *Science of Computer Programming*, vol. 41, no. 3, pp. 277–294, Nov. 2001. [Online]. Available: http://dx.doi.org/10.1016/S0167-6423(01)00011-9

[19] M. Casadei, L. Gardelli, and M. Viroli, "Simulating emergent properties of coordination in Maude: the collective sort case," in *5th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2006)*, ser. Electronic Notes in Theoretical Computer Science, C. Canal and M. Viroli, Eds. CONCUR 2006, Bonn, Germany: Elsevier Science B.V., 31 Aug. 2006, pp. 59—80. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2007.05.022

[20] M. Casadei, M. Viroli, and L. Gardelli, "On the collective sort problem for distributed tuple spaces," *Science of Computer Programming*, vol. 74, no. 9, pp. 702–722, 2009, Special Issue on the 5th International Workshop on Foundations of Coordination Languages and Architectures (FOCLASA '06). [Online]. Available: http://dx.doi.org/10.1016/j.scico.2008.09.018

[21] M. Viroli and M. Casadei, "Biochemical tuple spaces for self-organising coordination," in *Coordination Languages and Models*, ser. LNCS, J. Field and V. T. Vasconcelos, Eds. Lisbon, Portugal: Springer, Jun. 2009, vol. 5521, pp. 143–162, 11th International Conference (COORDINATION 2009), Lisbon, Portugal, Jun. 2009. Proceedings. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02053-7_8

[22] A. Di Pierro, C. Hankin, and H. Wiklicky, "Probabilistic Linda-based coordination languages," in *3rd International Conference on Formal Methods for Components and Objects (FMCO'04)*, ser. LNCS, F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever, Eds. Berlin, Heidelberg: Springer, 2005, vol. 3657, pp. 120–140. [Online]. Available: http://dx.doi.org/10.1007/11561163_6

[23] R. De Nicola, D. Latella, J.-P. Katoen, and M. Massink, "StoKlaim: A stochastic extension of Klaim," Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" (ISTI), Tech. Rep. 2006-TR-01, 2006. [Online]. Available: http://www1.isti.cnr.it/~Latella/StoKlaim.pdf

Parameter Engineering vs. Parameter Tuning: the Case of Biochemical Coordination in $\mathcal{M}o\mathcal{K}$

Stefano Mariani
DISI, ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: s.mariani@unibo.it

***Abstract*—To cope with nowadays MAS complexity, nature-inspired coordination models and languages gained increasing attention: in particular, biochemical coordination models. Being intrinsically stochastic and self-organising, the effectiveness of their outcome likely depends on a correct parameter tuning stage. In this paper, we focus on chemical reactions rates, showing that simply imitating chemistry "as it is" may be not enough for the purpose of effectively engineer complex, self-organising coordinated systems such as $\mathcal{M}o\mathcal{K}$.**

I. INTRODUCTION

Nowadays MAS are demanding new paradigms and abstractions to deal with their increasing complexity [1]. Such complexity is mostly due to the number and nature of interactions happening within and between MAS [2], [3]. Coordination models and languages – whose main goal is to govern such interactions [4] – have historically drawn inspiration from *self-organising coordination* in natural phenomena—e.g., pheromone-based [5] and chemical [6] coordination. Among the many, *biochemical coordination* has been shown to be particularly effective [7], [8]. Here, there is no central authority ruling the interaction space. Instead, a number of *local*, *stochastic* coordination rules, to which all the interacting agents "implicitly" obey – as they are the "laws of nature" – drive MAS coordination. Thus, it happens *by emergence* as a consequence of the self-organisation between MAS coordinated entities. Being a self-organising process, coordination effectiveness likely depends on a correct *parameter tuning* stage, often performed in loop with a *simulation* stage [9].

In the case of biochemical coordination, being the "laws of nature" the (artificial) chemical reactions installed in the coordination medium, such parameters are, e.g., the *rate* of application of a chemical reaction, the *concentration* of the chemicals participating the reaction and their *stoichiometry*—the "extent" to which chemicals participate. In this paper, we focus on rates, aiming at a twofold goal:

- on one hand, showing that the *law of mass action* for rate expressions may be not enough to effectively engineer a biochemical coordination middleware
- on the other hand, highlighting that designing arbitrary *functional rates* demands for a disciplined and principled approach different from "parameter tuning", which we call *parameter engineering*.

In particular, such goals are defined w.r.t. the $\mathcal{M}o\mathcal{K}$ model and the BioPEPA tool, used as the *subject* and the *means* of investigation, respectively. Nevertheless, a generalisation of such goals suitable for any kind of nature-inspired MAS is possible. First of all, the fact that a given natural system works properly relying on a given set of parameters, each of which has a given set of functional dependencies with others, doesn't necessarily mean that the same sets of parameters and functional dependencies will work for an artificial system drawing inspiration from the natural one. Then, to proficiently identify the relevant parameters and engineer their (possibly reciprocal) functional dependencies, a proper methodology is needed—which will likely rely on simulations.

Accordingly, the remainder of the paper is organized as follows: Section II explains what biochemical coordination is (Subsection II-A) and reminds the importance of simulation tools for self-organising systems development (Subsection II-B), also describing the $\mathcal{M}o\mathcal{K}$ model (Subsection II-A1) as well as the BioPEPA tool (Subsection II-B1); Section III conveys the main contribution of the paper, introducing and motivating the notion of parameter engineering through a number of functional rates engineering examples; finally, Section IV concludes also giving some hints about further works.

II. BACKGROUND

A. Biochemical Coordination

The chemical metaphor appears particularly appealing for MAS coordination due to the simplicity of its foundation [6]. The idea is to coordinate any MAS entity (agents as well as information) as "molecules" floating in a chemical "solution", whose evolution is driven by chemical "reactions" continuously and spontaneously consuming and producing molecules. As many chemical reactions can occur at a given time, chemical solution evolution is driven by race conditions among their *rates*, which means certain reactions are *stochastically* executing over others—as in chemistry actually is [10].

Biochemical tuple spaces [7] enhance such metaphor by adding a spatial abstraction: the *compartment*. A compartment is a tuple space equipped with biochemical reactions, driving the evolution of the molecules floating in it. Compartments may be networked in "neighbourhoods" as in chemistry happens through membranes, so as to shape more complex spatial structures—such as tissues and organs. Computationally, biochemical tuple spaces are a stochastic extension of the LINDA model [11]: the idea is to equip each tuple with a "concentration" value, representing a measure of the *pertinency/activity* of the tuple (molecule) within the space (compartment)—the higher it is, the more likely and frequently the tuple will influence system coordination [7]. Such concentration is evolved by biochemical rules installed into the compartment, affecting concentration values over time *exactly* in the same way chemical substances evolve into chemical solutions [10]—that is, according to the law of mass action [12], [10].

The *law of mass action* is[1] a mathematical model that *explains* and *predicts* the behaviour of solutions in dynamic equilibrium. It can be described with respect to two aspects: *(i)* the equilibrium aspect, concerning the composition of a reaction mixture at equilibrium and *(ii)* the kinetic aspect, concerning the *rate equations* for elementary reactions. The law states that the rate of an elementary reaction (r_f) – a reaction that proceeds through only one transition state, that is one mechanistic step – is proportional (k_f) to the product of the concentrations of the participating molecules (R^1, R^2):

$$r_f = k_f[R^1][R^2]$$

k_f is called *rate constant* and, in chemistry, is a function of participating molecules affinity—to learn more, please refer to [12] and therein cited bibliography.

The $\mathcal{MoK}$ model, briefly described in next section, models MAS coordinated entities as well as coordination processes by *(i)* adopting the chemical metaphor abstractions and *(ii)* borrowing (to some extent) from biochemical tuple spaces the computational model.

1) The $\mathcal{MoK}$ Model: $\mathcal{M}$olecules *o*f $\mathcal{K}$nowledge [13] ($\mathcal{MoK}$ for short) is a model for knowledge self-organisation in MAS. The main goals of $\mathcal{MoK}$ are:

- to let information chunks autonomously aggregate into heaps of knowledge
- to let knowledge autonomously flow toward the interested agents—rather than be searched

Here follows a brief summary of $\mathcal{MoK}$ model components—consider reading [13] and [14] for $\mathcal{MoK}$ formalisation and early application respectively:

- $\mathcal{MoK}$ atoms — produced by a given source to convey an "atomic piece of information", atoms should also store some metadata to ease semantic characterisation
- $\mathcal{MoK}$ molecules — "heaps" for information aggregation, they cluster together semantically related atoms
- $\mathcal{MoK}$ enzymes — enzymes reify knowledge-oriented (inter-)actions made by agents and are meant to influence molecules' concentration[2]
- $\mathcal{F}_{\mathcal{MoK}}$ function — as a knowledge-driven model, $\mathcal{MoK}$ must have a way to determine the semantic correlation between information, therefore, the $\mathcal{MoK}$ function $\mathcal{F}_{\mathcal{MoK}}$ should be defined, taking two molecules and returning a value $m \in [0, 1]$.
- $\mathcal{MoK}$ reactions — the behaviour of a $\mathcal{MoK}$ system is determined by biochemical reactions, which stochastically – according to their rate – drive molecules aggregation, reinforcement, decay, and diffusion:
    - ○ Aggregation[3] — bounds together semantically related molecules
    - ○ Reinforcement — consumes an enzyme to reinforce the related molecule

$$\texttt{enzyme}(Molecule^1) + Molecule^1_c \longmapsto^{r_{reinf}} Molecule^1_{c+1}$$

    - ○ Decay — enforcing time situatedness, molecules should fade away as time passes

$$Molecule_c \longmapsto^{r_{decay}} Molecule_{c-1}$$

    - ○ Diffusion — space situatedness is inspired by biology, therefore based upon diffusion to neighbouring compartments (tuple spaces)

$$\{Molecule^1 \bigcup Molecules^1\}_{\sigma_i} + \{Molecules^2\}_{\sigma_{ii}} \longmapsto^{r_{diffusion}} \{Molecules^1\}_{\sigma_i} + \{Molecules^2 \bigcup Molecule^1\}_{\sigma_{ii}}$$

These four biochemical reactions are the *minimum set* of coordination mechanisms believed (at the moment) to be *necessary* and *sufficient* to properly drive a $\mathcal{MoK}$-coordinated MAS toward the desired behaviour regarding knowledge self-organisation. Nevertheless, this set may be refined and extended if its lack of expressiveness w.r.t. $\mathcal{MoK}$ desiderata becomes evident. Anyway, having a well-defined set of *primitives* is a necessary step to start distinguishing what sorts of self-organising behaviours can and cannot be achieved with $\mathcal{MoK}$.

In fact, once such primitives are fixed, we can focus on the issue of properly engineering their rate expressions—r_{reinf}, r_{decay}, $r_{diffusion}$. In particular: is it sufficient to stick with the law of mass action to achieve $\mathcal{MoK}$ goals, or should we build our "custom" functional dependencies? If so, which parameters and which kind of dependencies (direct, inverse, etc.) are to be used in each rate expression? And how can MAS designers make such decisions?

Section III answers these questions through a number of examples exploiting the BioPEPA simulation tool – briefly described in next section – to analyse different alternatives regarding $\mathcal{MoK}$ reactions rate expressions.

B. Biochemical Simulation

Simulation has been widely recognized as a fundamental development stage in the process of designing and implementing both MAS as well as biochemical processes [16], [9]. This is mostly due to the high number of system parameters needed, the huge number of local interactions between components, the influence of randomness and probability on system evolution. A number of different simulation tools capable of modeling biochemical-like processes exist, either born in the biochemistry field (see [17] for a survey) or in the (Multi-)Agent Based Simulation research area (survey in [18]). Among the many, ALCHEMIST [19], PRISM [20], and BioPEPA [12] at least, are worth to be mentioned. Our choice fell on the latter for its appealing features – briefly described in next section – which perfectly suit the purpose of the paper.

[1]From http://en.wikipedia.org/wiki/Law_of_mass_action.

[2]Please, notice that an atom is a "singleton molecule", hence the term "molecule" will be used also for "atom" from now on.

[3]Aggregation reaction formalisation is not shown here because it has been left out from BioPEPA simulations for the lack of expressiveness of the tool. To learn more, please refer to the technical report [15].

1) The Bio-PEPA Tool: BioPEPA [12] is a language for modeling and analysis of biochemical processes. It is based on PEPA [21], a process algebra originally aimed at performance analysis of software systems, extending it to deal with some features of biochemical networks, such as stoichiometry and different kinds of kinetic laws—including the law of mass action. The most appealing features of BioPEPA are:

- custom kinetic laws represented by means of *functional rates*
- definition of stoichiometry ("how many" molecules of a given kind participate) and role played by the species (reactant, product, enzyme, ...) in a given reaction
- theoretical roots in CTMC semantics—behind any BioPEPA specification lies a stochastic labelled transition system modeling a CTMC

In BioPEPA, rate expressions are defined as mathematical equations involving reactants' concentrations (denoted with the reactant name and dynamically computed at run-time) and supporting mathematical operators (e.g. `exp` and `log` functions) as well as built-in kinetic laws (e.g. the law of mass action, denoted with the keyword `fMA`) and time dependency (through the variable `time`, changing value dynamically according to the current simulation time step)[4]. The BioPEPA Eclipse plugin[5] is the tool used in next section to investigate $\mathcal{MoK}$ reactions' rates influence on system behaviour.

III. Parameter Engineering in Rate Expressions

As far as nature-inspired MAS are concerned, simulation tools are usually exploited to study how *those parameters inherited from the natural metaphor* influence the overall MAS behaviour. This is done with the aim to fine-tune such parameters value so as to get the better run-time "performances"—whatever this means (often, a behaviour closer to that exhibited in nature).

But, what about the question of wether the natural system's parameters are well suited also for the artificial one? In particular, w.r.t. biochemical coordination (thus, $\mathcal{MoK}$ also): what about shaping our own rate expressions for biochemical reactions rather than blindly relying on the law of mass action to define their functional dependencies? Do we gain any improvement w.r.t. the overall coordinated MAS behaviour? Furthermore: can the same improvement be achieved by simply fine-tuning the natural system's parameters *as they are* in nature (e.g. the law of mass action *constant rate*)?

Through the following experiments, we aim at answering this kind of questions, hopefully achieving our twofold goal:

- showing that the law of mass action is too weak to effectively express a number of self-organising behaviours—such as $\mathcal{MoK}$'s
- highlighting that shaping custom functional dependencies for rate expressions is a complex task demanding a well-engineered approach—indeed, *parameter engineering* prior to parameter tuning

By generalisation, our first goal aims at showing there is the need to consider re-engineering natural system's parameters, as well as their functional dependencies, so as to better cope with the problem at hand—as done with other natural metaphors: most notably, the ACO approach to distributed optimisation, in which the original "ant" metaphor is indeed just a metaphor, not the actual implementation [22].

Therefore, for each of the following experiments, we *(i)* identify which are the desiderata for the MAS run-time behaviour, *(ii) engineer* rates by designing functional dependencies which are likely to pursue the chosen goal, *(iii)* include a pure parameter tuning stage to fine-tune the MAS behaviour (if needed). All of this is done one reaction (coordination policy) at a time, thus one functional rate at a time, incrementally accumulated until composing the whole MAS behaviour. This approach is what we call *parameter engineering*.

Furthermore, a principle we believe to be extremely important for engineering self-organsing systems will be kept in mind: keeping the number of *external* parameters as small as possible. For "external" we mean parameters which are *ad hoc* added to the coordination model – $\mathcal{MoK}$ in our case – to better design functional rate expressions—e.g., the law of mass action constant rate. On the contrary, *internal* parameters are those already present in the coordinated MAS—e.g., in the case of $\mathcal{MoK}$, the concentration of the reactants or the time flowing. The advantage of using internal parameters as opposed to external ones, lies in the fact that a system using more internal parameters than external ones is much more adaptive and self-regulating, since it only relies on "within-system" information rather than on "outside-system" data to dynamically adjust its behaviour—in the case of $\mathcal{MoK}$, reactions' rates.

Technical Notes on Experiments: Each of the following experiments has been performed by using Gillespie's stochastic simulation algorithm in 30 independent replications. Each of the following plots has been directly generated from BioPEPA as a result of the correspondent experiment—hence, of the 30 Gillespie runs. In each chart, the x-axis plots the time steps of the simulation, whereas the y-axis the concentration level of the reactants expressed in units of molecules.

A. Injection Rate

Althought injection of atoms into a $\mathcal{MoK}$ compartment is not yet part of $\mathcal{MoK}$'s core set of formalised reactions, its influence on the system is so important to deserve its own analysis. Basically, injection can be described as follows:

Injection — Produces atoms out of *sources* without consuming them

$$\texttt{source}(\mathit{Molecule}^1) + \mathit{Molecule}^1_c \longmapsto^{r_{inj}} \texttt{source}(\mathit{Molecule}^1) + \mathit{Molecule}^1_{c+1}$$

Two contrasting needs have to be addressed: on one hand, atoms should be *perpetually* injected into the MAS, since there is no way to know a-priori *when* some information will be useful; on the other hand, we would likely avoid flooding the system without any control on how many atoms are in play. Thus, three options are viable:

[4]To learn more about BioPEPA syntax, please refer to [12].

[5]Instructions on how to install at http://homepages.inf.ed.ac.uk/s9552712/bio-pepa/download.html, manual at http://homepages.inf.ed.ac.uk/stg/research/biopepa/eclipse/manual/manual.pdf

1) make injection rates decreasing as time passes
2) enforce some kind of "saturation" to stop injection
3) a combination of the two

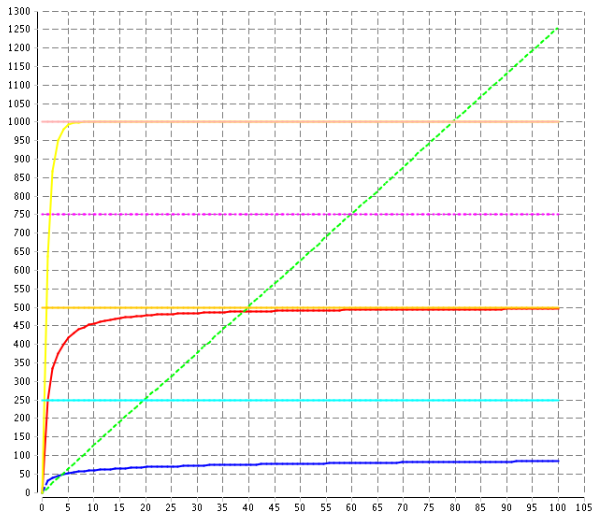


Fig. 1. Comparison of functional rates for atoms injection. Horizontal lines represent correspondent sources' concentration: purple dashed for option (1), pink for option (2), orange for option (3), light-blue for option (4).

Fig. 1 shows option (1) in blue, option (2) in yellow and option (3) in red. The green dashed line plots the law of mass action rate, whereas horizontal lines are the sources. Fig. 2 shows the BioPEPA functional rates specification used.

```
// option (1)
injE = [source_economics/atom_economics * (1 / (1 + time))];
// option (2)
injS = [source_sports - atom_sports];
// option (3)
injC = [(1 / (1 + time)) * (source_crime - atom_crime)];
// option (4)
injP = [fMA(0.05)];
```

Fig. 2. The `fMA` keyword calls a built-in function to compute the law of mass action. Its only parameter is the rate constant. The fMA implicitly consider reactants involved in the reaction exploiting its correspondent functional rate—for the full BioPEPA specification, please refer to [15].

Clearly, using rate expressions based on the law of mass action is out of question: its behaviour follows none of $\mathcal{M}o\mathcal{K}$ injection reaction desiderata. Once discarded also option (1), whose trend is clearly too slow in reaching saturation, options (2) and (3) may seem almost identical. Actually they are not:

- option (2) is "saturation-driven" only, thus if at some point in time `atom_sports` will suddenly decrease in concentration – e.g. due to agents consuming them – they will go back to saturation-level as fast as possible, no matter how long their sources are within the system
- option (3) instead, makes the saturation process time-dependant. In particular, the longer `source_sports` are within the system, the slower saturation will be

Choosing among the two depends on the application-specific context in which the $\mathcal{M}o\mathcal{K}$ model is used. In $\mathcal{M}o\mathcal{K}$-`News` [14], e.g., option (3) is better, since in the news management scenario information (on average) loses relevance as time passes.

B. *Decay Rate*

$\mathcal{M}o\mathcal{K}$ decay reaction is an effective way to resemble the relationship between information relevance and time flow. Furthermore, decay enforces a kind of *negative feedback* which, together with the *positive feedback* provided by $\mathcal{M}o\mathcal{K}$ enzymes, enables the *feedback loop* peculiar of natural systems.

Time dependency alone is not enough for a meaningful decay behaviour: by using, e.g., a fixed rate we end-up simply slowing down the saturation process provided by injection reaction. Hence, Fig. 3 shows three different combinations of time dependency and concentration dependency for $\mathcal{M}o\mathcal{K}$ decay reaction—a fourth one (yellow line), based on the law of mass action, is given for comparison purpose:

1) linear time dependency + relative concentration dependency (blue dashed line)
2) logarithmic time dependency + relative concentration dependency (red line)
3) linear time dependency + built-in law of mass action (green dashed line)

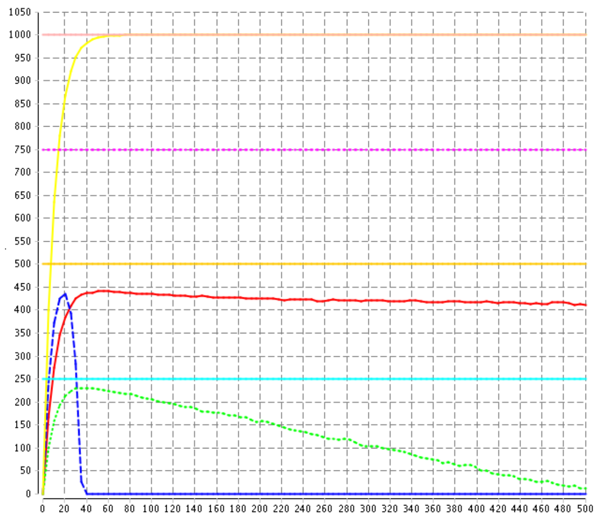


Fig. 3. Comparison of functional rates for atoms decay. Again, horizontal lines represent correspondent sources' concentration: purple dashed for option (1), orange for option (2), light-blue for option (3), pink for option (4).

Fig. 4 shows the BioPEPA functional rates specification used[6]. Again, the law of mass action is unsatisfactory, as well as option (1). Options (2) and (3) are both viable solutions instead. The choice is mostly driven by how fast are the dynamics of the scenario in which $\mathcal{M}o\mathcal{K}$ has to be deployed, thus how fast information should lose relevance—e.g., in $\mathcal{M}o\mathcal{K}$-`News`, choice (2) has been preferred. Nevertheless, please notice that option (3) has an additional parameter w.r.t. option (2): the law of mass action "rate constant". Furthermore, even if such parameter is made dynamic – e.g. the ratio between sources and atoms concentrations as done in options (1), (2) – the

[6]Actually, the Heaviside function has been also used to counter BioPEPA setting which allows rates to become negative—see [12].

```
// option (1)
decayE = [source_economics / atom_economics *
            time];
// option (2)
decayC = [source_crime / atom_crime *
            log(1+time)];
// option (3)
decayP = [fMA(0.05) * time];
// option (4)
decayS = [fMA(0.05)];
```

Fig. 4. BioPEPA specification of rate expressions for $\mathcal{M}o\mathcal{K}$ decay reaction.

trend still would not match our desiderata for $\mathcal{M}o\mathcal{K}$ decay reaction—compare with yellow line of Figure 4 in [15].

C. Reinforcement Rate

To properly engineer $\mathcal{M}o\mathcal{K}$ reinforcement reaction rate, we have to keep in mind what enzymes are meant for, that is, *(i)* representing a *situated interest* manifested by an agent w.r.t. a piece of knowledge – an atom or a molecule – *(ii)* to be exploited to reinforce such knowledge "relevance" within the MAS. With the word "situated" we mean that reinforcement should take into account the *situatedness* of agents (inter-)actions along a number of dimensions: time, space, type—a "search" action, a "read" action, etc. For these reasons, $\mathcal{M}o\mathcal{K}$ reinforcement reaction rate should:

- be prompt, that is rapidly increase molecules concentration—despite decay
- limited both in time and space, to resemble relevance relationship with situatedness of (inter-)actions
- depend on the (inter-)action type—e.g. a "read" action could inject more enzymes and/or reinforce atoms with greater stoichiometry w.r.t. a "search" action

Fig. 6 clearly shows that our desiderata are fulfilled only by a reinforcement reaction having a functional dependency on the ratio between the reinforced molecule's concentration and its source own—option (1) in Fig. 5. Once again, sticking

```
// option (1)
feedS = [(source_sports / atom_sports)];
// option (2)
feedE = [fMA(source_economics / atom_economics)];
// option (3)
feedC = [fMA(0.05)];
```

Fig. 5. BioPEPA specification of rate expressions for $\mathcal{M}o\mathcal{K}$ reinforcement reaction.

with the law of mass action alone is out of question: option (2) – dashed blue line –, even if adopting a dynamic rate constant, exhibits an exceedingly high and fast peak, option (3) – red line –, using a fixed rate constant (as in the law of mass action typically is), almost completely ignores the feedback—enzymes are too slowly consumed (orange line, plotting enzymes concentration).

Furthermore, Fig. 7 and Fig. 8 show, respectively, how *concentration* and *stoichiometry* can influence $\mathcal{M}o\mathcal{K}$ reinforcement reaction behaviour, effectively modeling situatedness—in particular, what we called the "type" of (inter-)actions. In fact, *(i)* in Fig. 7 the initial concentration of "red" enzymes

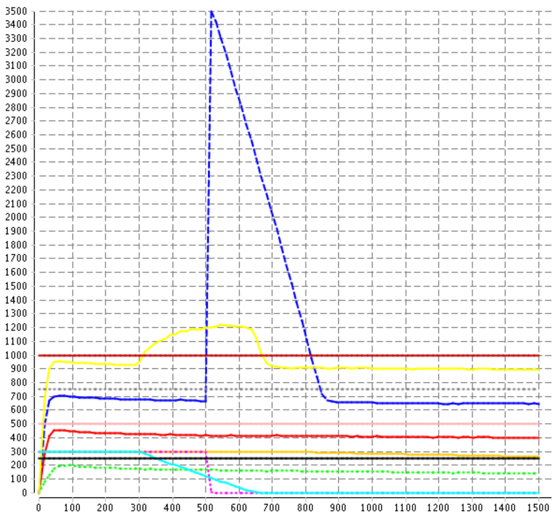


Fig. 6. Comparison of functional rates for atoms reinforcement. Lines worth to be considered are: the yellow one, plotting option (1), the dashed blue one, plotting option (2), the red one, plotting option (3).

(red line) is doubled w.r.t. "yellow" enzymes (yellow line) in Fig. 6: as a result, the "duration" of the feedback is doubled as well; *(ii)* in Fig. 8 the stoichiometry of "red" atoms (red line) in reinforcement reaction is doubled w.r.t. "yellow" atoms (yellow line) in Fig. 6: as a result, the "intensity" of the feedback is more than doubled.

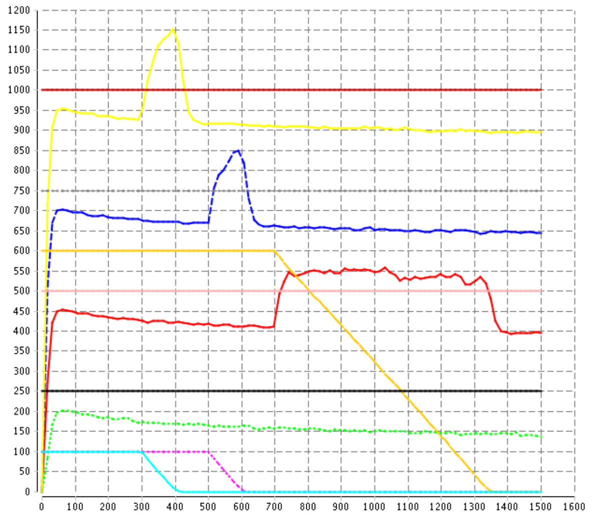


Fig. 7. Enzymes concentration increment effect on reinforcement.

Notice also: *(i)* Fig. 7 shows the opposite holds too, that is, halving the initial concentration halves the duration of the feedback (yellow and blue lines); *(ii)* Fig. 8 shows that no interference happens between concentration and stoichiometry parameters, in fact, reinforcement lasts as long as in Fig. 7.

D. Diffusion Rate

As regards $\mathcal{M}o\mathcal{K}$ diffusion reaction, the topology depicted in Fig. 9 has been taken as a reference. Namely, four $\mathcal{M}o\mathcal{K}$ compartments are imagined to be connected one to each other, allowing in principle any molecule to move anywhere.

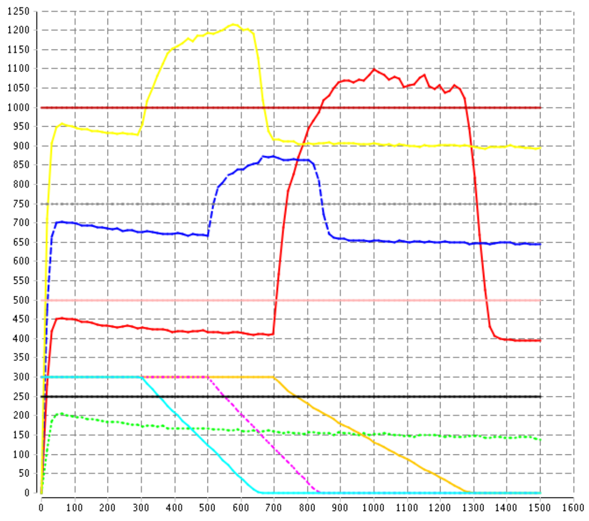


Fig. 8. Atoms stoichiometry increment effect on reinforcement.

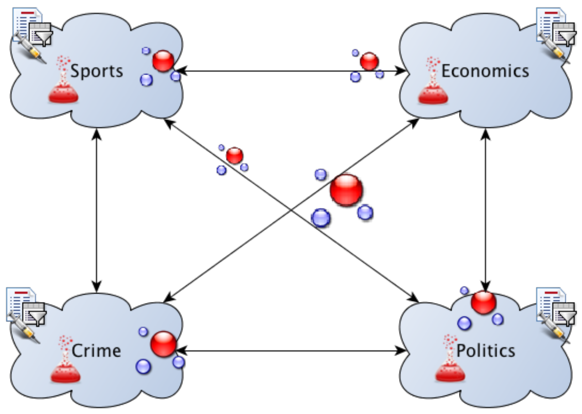


Fig. 9. $\mathcal{MoK}$ topology to experiment with diffusion reaction.

Our main desiderata regarding $\mathcal{MoK}$ diffusion reaction are similar to those of $\mathcal{MoK}$ injection reaction: on one hand, we would like to perpetually spread information around, because agents working in other compartments may be interested in it; on the other hand, we would also like to keep some degree of control about "how much" information is moved around. Such "degree of control" can be achieved by reusing the concept of "saturation", as shown by Fig. 11: in particular, it seems reasonable to allow only a fraction of molecules to move from their "origin" compartment—see Fig. 10. In practice, we can

```
// diffusion weight
DW = 0.75;
// diffusion functional rates (a@x => a@y)
diffSE = [DW * as@sports - as@economics]; // blue line
diffSC = [DW/2 * as@sports - as@crime]; // red line
diffSP = [DW/3 * as@sports - as@politics]; // green line
```

Fig. 10. Notation $r@c$ refers to the concentration of reactant r in compartment c. Previous listings did not follow such notation because there was only a single compartment—$\mathcal{MoK}$ diffusion was not considered.

arbitrarily decrease/increase the saturation-level of the origin compartment in the destination compartment. Furthermore,

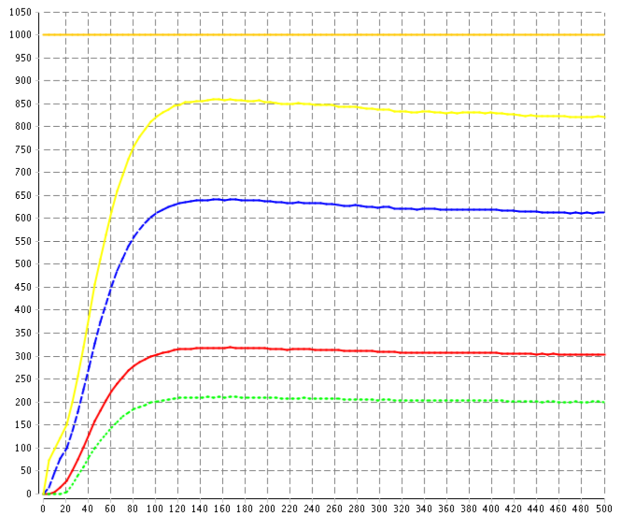


Fig. 11. $\mathcal{MoK}$ diffusion reaction trend. The yellow line plots the concentration level of the atoms in their "origin" compartment (the orange horizontal line represents their source).

they are functionally related. As a side note, notice a diffusion reaction featuring the law of mass action is not depicted. The motivation is that it exhibits an unexpected "malfunctioning" affecting also other reactions. More on this "interference problem" in next section.

E. On the Problem of Interference Between Reactions

All the experiments in the paper have been conducted incrementally, that is, each $\mathcal{MoK}$ reaction has been added to the BioPEPA specification one at a time. As reported in [15], when adding diffusion to other $\mathcal{MoK}$ behaviours, BioPEPA plots highlighted some *interference* between reactions. E.g., Fig. 12 depicts what happened when reinforcement has been added to injection, decay and diffusion.

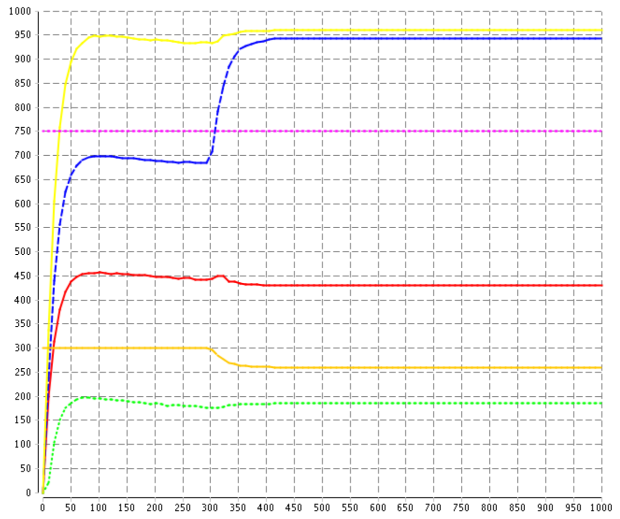


Fig. 12. $\mathcal{MoK}$ reinforcement reaction addition to injection, decay and diffusion. Not only enzymes are not fully depleted, but also undesirable and unexpected interferences with other reactions are clearly highlighted.

A number of unexpected behaviours can be seen:

- first of all, our desiderata for $\mathcal{M}o\mathcal{K}$ reinforcement reaction are not met (dashed blue line). In particular, it seems atoms cannot go beyond their original compartment concentration level (yellow line)
- second, enzymes are not fully depleted (orange line)
- last but not least, other atoms are affected by a successful application of $\mathcal{M}o\mathcal{K}$ reinforcement reaction (yellow, red and green lines): in particular, in the time interval during which enzymes are consumed *all* other trends experiment some fluctuations

The reason at the root of all these issues is still unknown: being chemical-like reactions scheduling essentially based on race conditions between the correspondent functional rates – evaluated at a given point in time –, understanding what exactly happens within the system at a given time step is not trivial at all—or even impossible, depending on the debugging services the simulation tool adopted provides. Nevertheless, the satisfactory BioPEPA specification shown in Fig. 13 has been found. In particular, $\mathcal{M}o\mathcal{K}$ reinforcement reaction rate has been added a "feed factor" parameter, used to weight the influence of the atoms to be reinforced w.r.t. the concentration of the corresponding source in the compartment the latter belongs to. Fig. 14 shows that our desiderata are now met

```
// feed factor > 1
FF = 2;
// option (1)-revised
feedEC = [se@economics / (ae@crime * FF)];
```

Fig. 13. Adjusted BioPEPA specification of rate expressions for $\mathcal{M}o\mathcal{K}$ reinforcement reaction used together with $\mathcal{M}o\mathcal{K}$ diffusion reaction.

successfully. Although not shown here for the lack of space, also the functional dependencies on enzymes concentration and atoms stoichiometry shown in Fig. 7 and Fig. 8 are preserved.

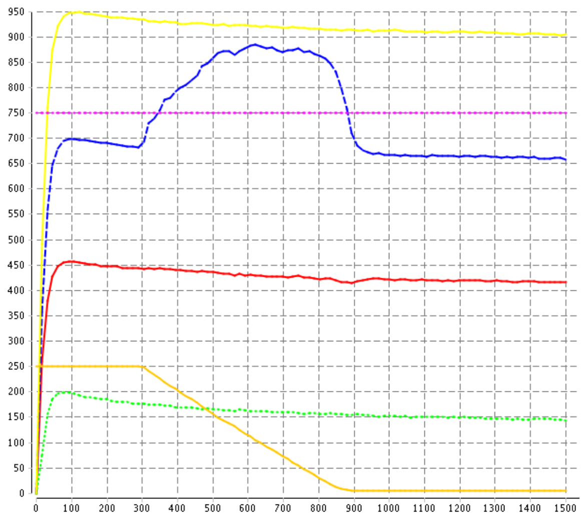


Fig. 14. Adjusted $\mathcal{M}o\mathcal{K}$ reinforcement reaction: enzymes are now completely depleted and other reactions no longer affected.

This clearly demonstrates the intricacies behind rates design in biochemical coordination, therefore motivating the principled and disciplined – namely, *engineered* – approach to parameter tuning we called parameter engineering.

IV. Conclusion & Further Works Roadmap

In this paper, we showed that simply imitating nature *as is* may be not the optimal approach while engineering nature-inspired MAS. Indeed, once a suitable natural metaphor has been found, MAS designers should ask themselves if the natural system's parameters are the optimal ones also for the artificial system they aim to build. If it is not the case, they should clearly state which goals their MAS is pursuing then detects, preferably within the MAS itself, which parameters better suits their needs as well as which (if any) functional dependencies between such parameters better cope with the problem their MAS aims to solve.

In particular, we focussed on the case of biochemical coordination in $\mathcal{M}o\mathcal{K}$, showing that sticking with the law of mass action for rate expressions is not enough to model interesting behaviours. Furthermore, designing more complex rate expressions demands for a principled approach going beyond parameter tuning, which we call parameter engineering, likely to be supported by incremental simulation of each single basic "law of nature" in play.

To the best of our knowledge, no closely related works exists to date except, to some extent, [9]. Nevertheless, we believe our work to be complementary to that in [23] about self-organising design patterns as well as to [9]: in fact, once a design pattern has been recognized as a potential solution to a given problem, a simulation stage is out of doubts useful, therefore a parameter engineering phase necessary.

As a last note, further works will be devoted to analyze the tradeoff between designing more complex expressions and sticking with the law of mass action at the cost employing more (dual, complementary and/or opposite) reactions to reach the same "complex trend" by composition.

Acknowledgments

This work has been supported by the EU-FP7-FET Proactive project SAPERE – Self-aware Pervasive Service Ecosystems, under contract no. 256874.

References

[1] A. Omicini and P. Contucci, "Complexity & interaction: Blurring borders between physical, computational, and social systems. Preliminary notes," in *Computational Collective Intelligence. Technologies and Applications*, ser. LNCS, C. Bădică, N. T. Nguyen, and M. Brezovan, Eds., vol. 8083. Springer Berlin Heidelberg, 2013, pp. 1–10, 5th International Conference (ICCCI 2013). Craiova, Romania, 11–13 Sep. 2013, Proceedings. Invited Paper. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40495-5_1

[2] P. Wegner, "Why interaction is more powerful than algorithms," *Communications of the ACM*, vol. 40, no. 5, pp. 80–91, May 1997. [Online]. Available: http://dx.doi.org/10.1145/253769.253801

[3] A. Omicini, A. Ricci, and M. Viroli, "The multidisciplinary patterns of interaction from sciences to Computer Science," in *Interactive Computation: The New Paradigm*, D. Q. Goldin, S. A. Smolka, and P. Wegner, Eds. Springer, Sep. 2006, pp. 395–414. [Online]. Available: http://dx.doi.org/10.1007/3-540-34874-3_15

[4] D. Gelernter and N. Carriero, "Coordination languages and their significance," *Communications of the ACM*, vol. 35, no. 2, pp. 97–107, 1992. [Online]. Available: http://dx.doi.org/10.1145/129630.129635

[5] P.-P. Grassé, “La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d’interprétation du comportement des termites constructeurs,” *Insectes Sociaux*, vol. 6, no. 1, pp. 41–80, Mar. 1959. [Online]. Available: http://dx.doi.org/10.1007/BF02223791

[6] J.-P. Banâtre, P. Fradet, and D. Le Métayer, “Gamma and the chemical reaction model: Fifteen years after,” in *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View*, ser. LNCS, C. S. Calude, G. Păun, G. Rozenberg, and A. Salomaa, Eds. Springer, 2001, vol. 2235, pp. 17–44. [Online]. Available: http://dx.doi.org/10.1007/3-540-45523-X_2

[7] M. Viroli and M. Casadei, “Biochemical tuple spaces for self-organising coordination,” in *Coordination Languages and Models*, ser. LNCS, J. Field and V. T. Vasconcelos, Eds. Lisbon, Portugal: Springer, Jun. 2009, vol. 5521, pp. 143–162, 11th International Conference (COORDINATION 2009), Lisbon, Portugal, Jun. 2009. Proceedings. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02053-7_8

[8] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo, M. Risoldi, A.-E. Tchao, S. Dobson, G. Stevenson, Y. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally, “Self-aware pervasive service ecosystems,” *Procedia Computer Science*, vol. 7, pp. 197–199, Dec. 2011, proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11). [Online]. Available: http://dx.doi.org/10.1016/j.procs.2011.09.006

[9] L. Gardelli, M. Viroli, and A. Omicini, “On the role of simulations in engineering self-organising MAS: The case of an intrusion detection system in TuCSoN,” in *Engineering Self-Organising Systems*, ser. LNAI, S. A. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, Eds. Springer, 2006, vol. 3910, pp. 153–168, 3rd International Workshop (ESOA 2005), Utrecht, The Netherlands, 26 Jul. 2005. Revised Selected Papers. [Online]. Available: http://dx.doi.org/10.1007/11734697_12

[10] D. T. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, Dec. 1977. [Online]. Available: http://dx.doi.org/10.1021/j100540a008

[11] D. Gelernter, “Generative communication in Linda,” *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, Jan. 1985. [Online]. Available: http://dx.doi.org/10.1145/2363.2433

[12] F. Ciocchetta and J. Hillston, “Bio-PEPA: A framework for the modelling and analysis of biological systems,” *Theoretical Computer Science*, vol. 410, no. 33–34, pp. 3065 – 3084, 2009, concurrent Systems Biology: To Nadia Busi (1968–2007). [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2009.02.037

[13] S. Mariani and A. Omicini, “Molecules of Knowledge: Self-organisation in knowledge-intensive environments,” in *Intelligent Distributed Computing VI*, ser. Studies in Computational Intelligence, G. Fortino, C. Bădică, M. Malgeri, and R. Unland, Eds., vol. 446. Springer, 2013, pp. 17–22, 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24-26 Sep. 2012. Proceedings. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32524-3_4

[14] ——, “Self-organising news management: The *Molecules of Knowledge* approach,” in *1st International Workshop on Adaptive Service Ecosystems: Natural and Socially Inspired Solutions (ASENSIS 2012)*, J. L. Fernandez-Marquez, S. Montagna, A. Omicini, and F. Zambonelli, Eds., SASO 2012, Lyon, France, 10 Sep. 2012, pp. 11–16, pre-proceedings. [Online]. Available: http://dx.doi.org/10.1109/SASOW.2012.48

[15] S. Mariani, “Analysis of the Molecules of Knowledge model with the Bio-PEPA Eclipse plugin,” ALMA MATER STUDIORUM–Università di Bologna, Bologna, Italy, AMS Acta Technical Report 3783, 20 Sep. 2013. [Online]. Available: http://amsacta.unibo.it/3783/

[16] E. Merelli, G. Armano, N. Cannata, F. Corradini, M. d’Inverno, A. Doms, P. Lord, A. Martin, L. Milanesi, S. Möller, M. Schroeder, and M. Luck, “Agents in bioinformatics, computational and systems biology,” *Briefings in Bioinformatics*, vol. 8, no. 1, pp. 45–59, 2007. [Online]. Available: http://dx.doi.org/10.1093/bib/bbl014

[17] R. Alves, F. Antunes, and A. Salvador, “Tools for kinetic modeling of biochemical networks,” *Nature biotechnology*, vol. 24, no. 6, pp. 667–672, 2006. [Online]. Available: http://dx.doi.org/10.1038/nbt0606-667

[18] C. Nikolai and G. Madey, “Tools of the trade: A survey of various agent based modeling platforms,” *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 2, p. 2, 2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/2/2.html

[19] D. Pianini, S. Montagna, and M. Viroli, “Chemical-oriented simulation of computational systems with Alchemist,” *Journal of Simulation*, 2013. [Online]. Available: http://dx.doi.org/10.1057/jos.2012.27

[20] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: Verification of probabilistic real-time systems,” in *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[21] S. Gilmore and J. Hillston, “The PEPA workbench: A tool to support a process algebra-based approach to performance modelling,” in *Computer Performance Evaluation Modelling Techniques and Tools*, ser. Lecture Notes in Computer Science, G. Haring and G. Kotsis, Eds. Springer Berlin Heidelberg, 1994, vol. 794, pp. 353–368. [Online]. Available: http://dx.doi.org/10.1007/3-540-58021-2_20

[22] M. Dorigo and M. Birattari, “Ant colony optimization,” in *Encyclopedia of Machine Learning*, C. Sammut and G. Webb, Eds. Springer US, 2010, pp. 36–39. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-30164-8_22

[23] J. Fernandez-Marquez, G. Marzo Serugendo, S. Montagna, M. Viroli, and J. Arcos, “Description and composition of bio-inspired design patterns: a complete overview,” *Natural Computing*, pp. 1–25, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11047-012-9324-y

A Multi-Agent Solution for the Interoperability Issue in Health Information Systems

Paolo Sernani, Andrea Claudi, Luca Palazzo, Gianluca Dolcini, Aldo Franco Dragoni
Dipartimento di Ingegneria dell'Informazione (DII)
Università Politecnica delle Marche
Via Brecce Bianche 60131 Ancona Italy
{p.sernani, a.claudi, l.palazzo, g.dolcini, a.f.dragoni}@univpm.it

***Abstract*—To achieve high quality and efficiency standards, interoperability between different information systems in healthcare is strongly required. Distribution, high modularity, robustness are features of agent oriented architectures, making Multi-Agent Systems (MAS) ideal for Health Information Systems (HIS), as the healthcare domain is characterized by system and data heterogeneity. This paper presents an agent oriented architecture to address this kind of issues, capable to access geographically distributed data to allow health professionals to retrieve/update any patient's record efficiently and reliably. The proposed architecture is composed by three layers, to allow local data storage keeping clinical information available by authorized facilities and physicians. Furthermore MAS technology integrates with legacy systems, wrapping them with agents.**

I. Introduction

The healthcare domain is facing a growing number of challenges: the incidence of medical errors is rising; many medical facilities are understaffed, and serve increasingly large areas; healthcare costs are rising more and more; healthcare facilities are under pressure to provide better services with less resources [1]. The median age of population is increasing, resulting in a rise in the number of chronic diseases and thus of health-related emergencies [2]. Health Information Systems (HIS) can provide a better coordination among medical professionals and facilities, reducing the number and incidence of medical errors [3]. They are considered as a solution to assist physicians in tracking patient medical history, interventions, encounters and lab test results [4]. In the same time, they can reduce healthcare costs and may provide a means to improve the management of hospitals [5], [6]. Unfortunately, due to the inherent complexity of their application domain, HIS are fragmented in various systems that hardly make use of communication standards, process definition protocols and homogeneous data representations. Thus international boards and local governments are defining general requirements for HIS and supporting the adoption of health information technology ([7]), to provide sustainable and effective healthcare services. Italian Health Ministry, following European Union (EU) directives, defines the requirements for the "Basic Infrastructure for Electronic Healthcare" [8]:

- *localization and availability of health records*. Patients' clinical information should be available 24 hours a day and 7 days a week, wherever data are stored.
- *Federated architecture*. Healthcare facilities and services are distributed and federated by nature, and clinical data should be maintained in the facility where they are produced, ensuring that information will be updated when necessary.
- *Security and privacy*. Due to the importance and the strictly personal nature of clinical data, information should be processed by mean of secure architectures, addressing privacy laws.
- *Scalability, modularity and reliability*. The infrastructure should be modular, to avoid a quick obsolescence, and scalable, to support the growing number of medical records; a HIS should be designed to achieve a safety critical degree of reliability.
- *Integration with legacy systems*. HIS architecture should integrate with existing systems in order to preserve past investments and to make its adoption practicable for local facilities.
- *Use of open standards*. It is, in fact, a mandatory requirement for those systems, as HIS, addressing interoperability issues.

Multi-Agent System (MAS) paradigm, being characterized by decentralization and parallel execution of activities based on autonomous entities [9] with social ability [10], could be ideal to implement HIS that respond to the needs of the healthcare domain: fields such as information access, decision support systems, internal hospital tasks would gain the greatest advantages from the typical distribution of agent technology and the existing standardization of communication between agents [11].

A. Our Contribution

In this paper we propose an agent oriented architecture capable to access geographically distributed data to allow health professionals to retrieve/update any patient's record efficiently and reliably. Such architecture meets the interoperability requirements among different health facilities and, at the same time, integrates with existing legacy systems (including local databases), being a new software layer on top of existing ones: this allows to protect the investments made by facilities and institutions as required by ministerial directives [8], in addition to address interoperability issues.

The main advantages of such architecture are:

- *Distribution*. A key concept of agent technology is flexibility: the complex issues of interoperability and integration with existing systems is broken down to minor tasks assigned to individual agents: cooperation is the solution to the original question. Retrieving data is possible from any point in the territory just through communication of distributed agents, and expensive infrastructures - as happens with cloud solutions - are not required.

- *High modularity*. Thanks to standardization activities made by the MAS community - FIPA IEEE -, simply adding new agents in the architecture (registering their services and sharing the same ontology) is enough in order to extend the capabilities of the system.
- *Robustness*. An agent oriented infrastructure provides many recovery techniques to better achieve fault tolerance goals.
- *Integration with existing systems*. With the aid of wrapper agents, each one designed for a particular instance of legacy information systems, the architecture represents a higher fully interoperable software layer. Communication at this level is readily able to use well established standard ontologies for messaging (HL7), definition of clinical documents (HL7 CDA), scheduled workflows (IHE) and health care terminologies (such as LOINC and SNOMED CT).

B. *Paper Structure*

The rest of this paper is organized as follow: section II describes related works, about HIS and adopted technologies to implement them; section III details the multi-agent system architecture; section IV illustrates an implementation related to an emergency-response scenario; section V points out some qualitative evaluations about the proposed architecture; finally, section VI draws the conclusions from the described work.

II. Related Works

In recent years, two different technologies have been the subject of much of the research relating to HIS: cloud computing and multi-agent systems. A mobile system that enables electronic healthcare data storage, update and retrieval using Cloud Computing is proposed in [12], in which a mobile application based on an Android client enables the users to retrieve remotely health information and images. In [13] a wireless sensor network is used to automate the data collection process. The collected information are distributed through a Cloud Computing solution to medical staff. An alternative approach is proposed in [14], where data and service interoperability is obtained through a distributed and agent-oriented system. [15] and [16] use the multi-agent system technology to support the home-care monitoring and treatment of patients. In [17] software agents are developed as personal assistants for physicians and administrative staff, trying to free them from routine work.

Also researches about HIS impact have been carried out. In [4] several papers concerning HIS and their implementations are examined in order to understand factors and influencers from previous experiences. In [18] scientific literature is investigated in order to provide a conceptual basis to understand and address HIS success and failure. The work in [19] analyses positive and negative findings in HIS research, remarking the lack of reports about negative results, necessary to assess benefits of HIS. Finally formal ways to design and develop HIS are necessary since the wide introduction of health information technologies can lead to new types of errors (see for example [20], [21]).

III. Infrastructure

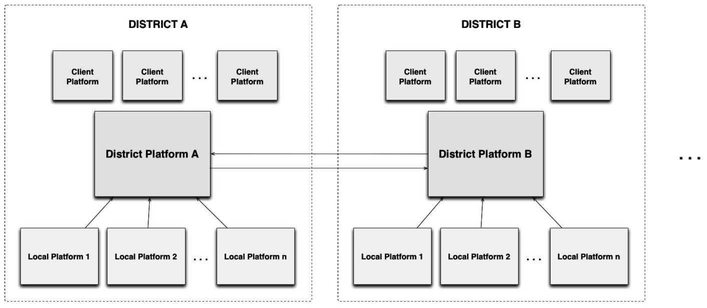


Fig. 1. The global architecture is structured in three logic layers.

The agent oriented architecture is expressed by three levels of abstraction, named local platform, district platform and client platform (Fig.1-2): each one is characterized by its specific agents and resources as described in the next subsections. The discriminating factor between the first two layers is of administrative nature: there is a local platform for each health facility in the territory (e.g. a hospital); facilities refers to administrative districts, which constitute the second layer of the architecture; finally, the client level is represented by any software agent which needs to login to the infrastructure to retrieve documents or insert/update a patient's health record.

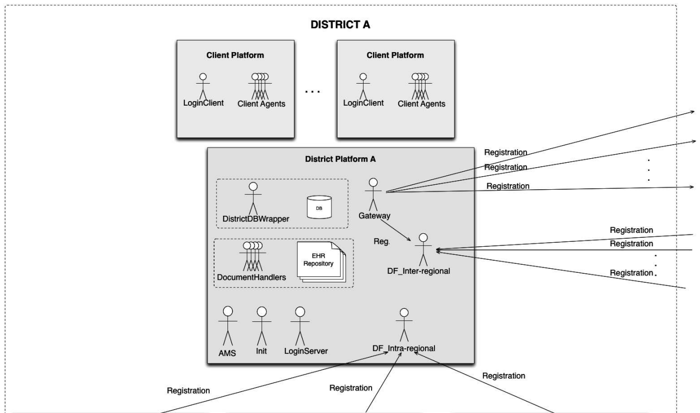


Fig. 2. Relation between local platforms and their referring district platform.

A. *Local Platform*

There is a local platform (Fig. 3) for each health facility. It has the role to interface with any information system, currently

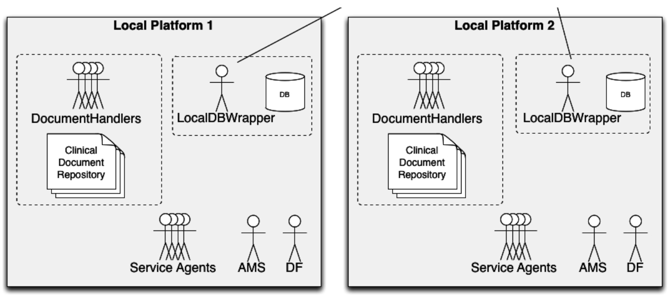


Fig. 3. Local platform agents.

present in the structure, committed to the management of clinical documents (create, edit, search, access) and the scheduling of different departments in the facility. Every local platform needs to know the address of its referring district platform in order to have access to the entire agent infrastructure.

LocalDBWrapper.: The task of such agents is to interface with the databases of a certain local healthcare institution. The advantages in the use of wrapping agents are the following:

- All the legacy systems would not be modified or replaced, but in fact encapsulated within such agents. In this way, any external agent, which needs to access to data contained by a local database, will be able to obtain them simply by communicating with the referring LocalDBWrapper agent, thus avoiding direct interaction with legacy systems.
- It makes possible to abstract the actual data representation within the different information systems available in the various facilities. With this solution, we don't need to address issues like information conflicts (such as homonymy and synonymy) or data schema inconsistencies by burdensome techniques of renaming, restructuring or even system redesign; it is sufficient to design a wrapping agent for each different legacy system able to translate the internal data representation in the ontology shared by all the agents in the infrastructure.

Hence, using agents to wrap local databases allows to keep data in the local facilities where the medical records are generated, differently from outsourced cloud solutions that store data in remote servers and have to deal with privacy concerns [22]. Furthermore agents have proven useful when directly acting as Web Services, providing agent-based services [23]. An agent that needs data from the LocalDBWrapper is able to obtain the service with a message exchange in the FIPA Agent Communication Language. In order to add a local platform to the entire agent oriented architecture, the LocalDBWrapper agents must register to DF_Intra-District agent of their referring district platform: this makes it available from distributed and remote agents, which need to retrieve data contained by the local structure.

DocumentHandler.: This kind of agents are able to access the content of a specific clinical document produced within the facility, such as clinical reports, laboratory tests, prescriptions, etc. In general, a DocumentHandler is contacted by a client agent to get health records managed by it: the DocumentHandler agent locates the requested document through its unique identifier, obtains it from the clinical repository and translates the information in an outgoing message towards the requesting client agent. Hence, the latter will be able to get the contents of clinical data requested.

Service Agents.: This set consists of agents for the management of different departments of the healthcare structure (e.g. radiology, cardiology, analysis laboratory, etc.). This paper does not provide further information on this field, but it is possible to find details about an agent oriented implementation of the Radiology Scheduled Workflow provided by Integrated the Healthcare Enterprise (IHE) consortium in [24].

B. District Platform

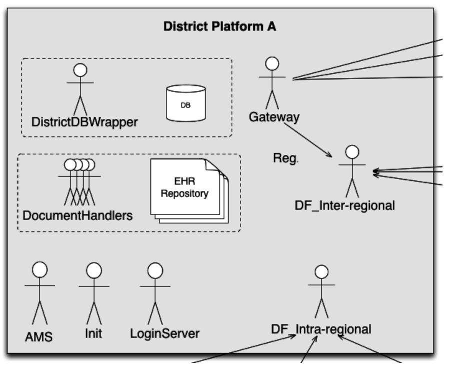


Fig. 4. District platform agents.

The main task of a district platform (Fig. 4) is to encapsulate all the local platforms that administratively belong to it. Basically, the district platforms represent the logic layer which composes the final architecture and allows to achieve the interoperability goal of our distributed system: every district platforms, therefore, must know each other their address.

DistrictDBWrapper.: These agents have similar functions with local wrappers: they manage data within district databases. The gateway agent contacts wrappers in order to store or retrieve any reference to a patient's clinical records, which have been produced by every local platform in the territory or by general practitioners.

DocumentHandler.: DocumentHandler agents manage those kind of documents which are of administrative competence of a district, such as Electronic Health Record (EHR) and Patient Summary [25]. They may refer to health records which are distributed in different local platforms: the Gateway agent has the role to look for and gather this information.

Gateway.: The Gateway agent catches the client requests and makes queries to local and district wrappers to retrieve data about any distributed health record of a citizen (Fig. 5). It returns the addresses of DocumentHandler agents which the client must contact to get the required documents. To accomplish this task, the gateway performs two basic activities:

- When it retrieves the distributed data required to fulfil a client request, it must integrate them into a data structure, so that the client can handle a single dataset.
- When a clinical record is produced within a district for a patient belonging to another district, the former

gateway must inform the latter one to make its referring DistrictDBWrapper agent register such event in its own district database.

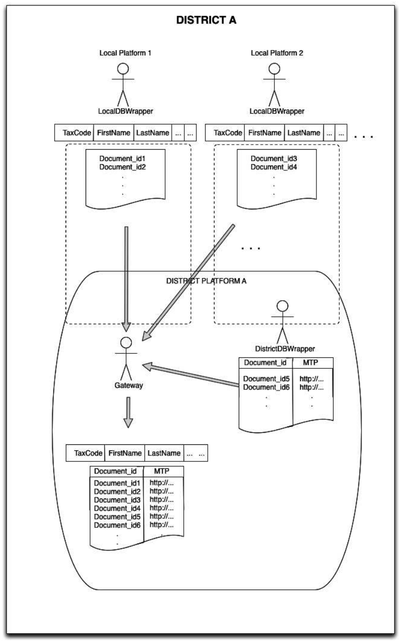


Fig. 5. The Gateway agent retrieves the location of health records within the system.

Init.: During the starting phase of the district platform, the Init agent registers the same platform Gateway to all the active DF_Inter-district agents of the remote district platforms in the territory.

DF_Inter-district.: As we just said, it is the Directory Facilitator in which all the remote Gateways are registered. This allows a single Gateway to communicate with any other distributed gateway in the entire infrastructure.

DF_Intra-district.: This Directory Facilitator contains all the LocalDBWrapper agents registrations of the local platforms belonging to the same district.

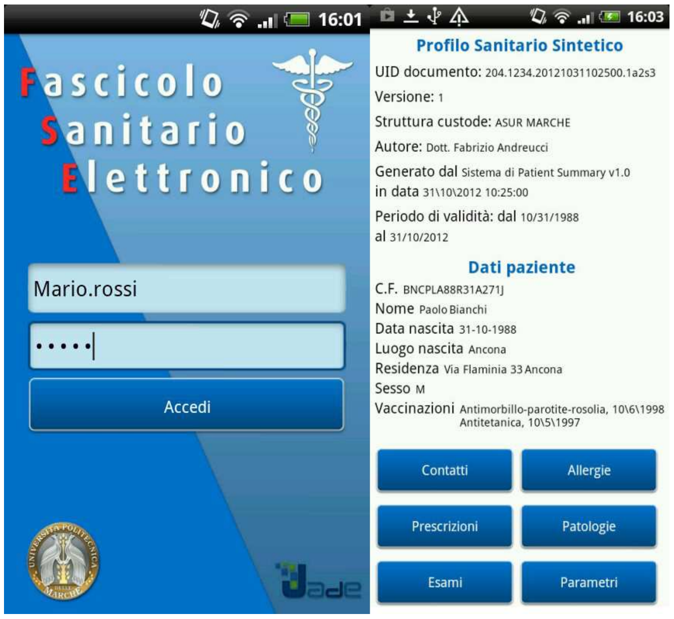
LoginServer.: Its task is to establish a secure connection with the client that wants to access to the infrastructure to retrieve data in a specified district.

C. Client Platform

This logic platform contains client applications, which may be any agent oriented software that is able, after a login phase, to access data through the connection with a district gateway agent. Examples of client applications could be: software to access EHR, both by medical staff and citizens, mobile applications to retrieve the Patient Summary for emergency situations, software to update health records by general practitioners, etc.

IV. Scenario

To show the capabilities of this architecture we assumed a scenario where an emergency doctor urgently needs to consult a patient's health records, in particular his patient


16:01

Fascicolo Sanitario Elettronico

Mario.rossi

•••••

Accedi

16:03

Profilo Sanitario Sintetico

UID documento: 204.1234.20121031102500.1a2s3

Versione: 1

Struttura custode: ASUR MARCHE

Autore: Dott. Fabrizio Andreucci

Generato dal Sistema di Patient Summary v1.0
in data 31\10\2012 10:25:00

Periodo di validità: dal 10/31/1988
al 31/10/2012

Dati paziente

C.F. BNCPLA88R31A271J
Nome Paolo Bianchi
Data nascita 31-10-1988
Luogo nascita Ancona
Residenza Via Flaminia 33 Ancona
Sesso M
Vaccinazioni Antimorbillo-parotite-rosolia, 10\6\1998
Antitetanica, 10\5\1997

Fig. 6. Login and main screen of mobile application for the Patient Summary.

summary. According to the EU definition, a patient summary is a clinical document that is digitally stored in repositories with cumulative indexing systems and secure access by authorised people. It is an HL7 CDA compliant document, contained in the patient's EHR, whose purpose is to summarize a patient's clinical history and his current situation.

In short, the main Patient Summary's use cases can be summed up in [26]:

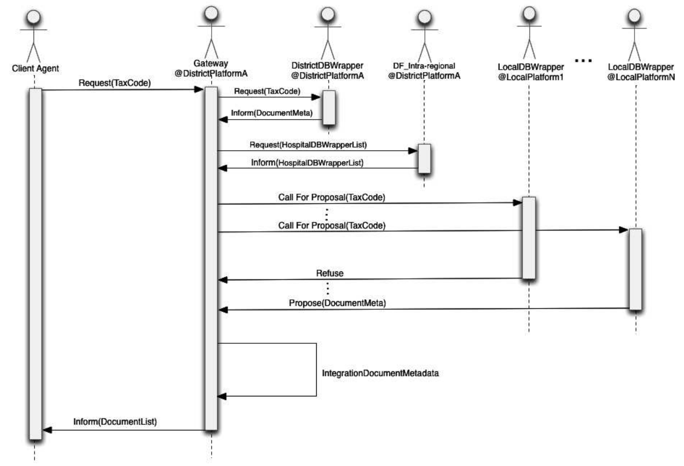
- Emergency situations in which the patient may not give an exhaustive description about his clinical history (problems, allergies, current medicines, etc.).
- Reliability of the information flows between family doctor and health facilities.
- Patients affected by chronic diseases managed by several specialists or elderly in home care regime.
- Diagnostic process support, telemedicine, etc.

Finally, the Patient Summary contains both mandatory and optional fields, and it is expressed through XML markup language.

To build such scenario we used:

- JADE Framework [23] to develop local and district agents in some desktop computers.
- An android smartphone application (Fig. 6) to simulate the client agent, developed with JADE LEAP add-on.
- Ministerial directives to compose a Patient Summary for our experiment, an XML parser and an agent ontology based on HL7 concepts.

The operating mode is very simple (Fig. 7). First of all, the mobile client application log in to the district platform entering its username and password: a secure connection is established with the platform using TSL protocol to ensure secure access to patients' personal and sensitive data. Then, the client asks for a citizen's Patient Summary and its relative health records by typing his tax code: the Gateway agent will query the different distributed entities to find the location of required data and inform the client where it can retrieve health records. Finally, the client application gather this data asking directly to DocumentHandler agents of the platforms which hold the patient's records.



Client Agent
Gateway @DistrictPlatformA
DistrictDBWrapper @DistrictPlatformA
DF_Intra-regional @DistrictPlatformA
LocalDBWrapper @LocalPlatform1
...
LocalDBWrapper @LocalPlatformN
Request(TaxCode)
Request(TaxCode)
Inform(DocumentMeta)
Request(HospitalDBWrapperList)
Inform(HospitalDBWrapperList)
Call For Proposal(TaxCode)
Call For Proposal(TaxCode)
Refuse
Propose(DocumentMeta)
IntegrationDocumentMetadata
Inform(DocumentList)

Fig. 7. The client agent queries the infrastructure for a citizen's health records.

V. Evaluation

Since the proposed MAS has to be understood as an infrastructure to address the interoperability issue in HIS domain and we implemented a simple proof-of-concept for a specific use case scenario, a quantitative evaluation is impossible at this stage of the work. Nevertheless a qualitative evaluation about the benefits produced by the adoption of MAS can be outlined.

The standardization of agent communication permits to integrate legacy system and new services wrapping them with agents, achieving interoperability and modularity. The JADE framework, adopted for our emergency scenario, adheres to FIPA standards allowing agents to register their services in a Directory Facilitator (DF) agent. Hence, other agents can query the DF to obtain the services they need. Also reliability and robustness can be achieved with MAS approach: keeping the focus on JADE, the framework offers the Main Container Replication Service (MCRS) and the DF persistence; in this way the container responsible for agent management is not a single point of failure and offered services are always traceable.

Some domain experts[1] have concerns about the actual applicability of our Multi-Agent approach in present HIS for which Service Oriented Architectures (SOA) are widely adopted. In our opinion an integration between MAS and SOA is both possible and desirable. Furthermore W3C specifications about Web Services confirm that software agents are the running programs that drive Web services both to implement them and to access them [27].

[1]as Klaus-Peter Adlassnig, during the final panel of the 1st International Workshop on Artificial Intelligence and NetMedicine (NETMED'12)

VI. Conclusion

In health information systems, the importance of addressing interoperability issues among existing systems is widely recognized. A crucial aspect is to allow health professionals to get any information they need about a patient in a pervasive and reliable way, even if these data are distributed in technically and geographically different health information systems.

To meet these requirements, in this paper we proposed an architecture based on MAS technology that takes advantage of the adoption of established standards for the management of clinical documents. Our goal was to show how MAS features can contribute in HIS in terms of interoperability, reliability, modularity and robustness; and how health professionals - and thus citizens - could benefit from this efficient distributed system. The adoption of a Multi-Agent architecture responds to requirements prescribed by international boards and local governments; differently from cloud computing solutions that propose to centralize data in the cloud, MAS are more suitable to respect the distributed and federated nature of healthcare services. The proposed architecture ensures that clinical data are stored in the same place where they are produced, and seems better fit what prescribed by privacy laws. Furthermore legacy systems can be integrated simply wrapping them with agents that have to share the same ontology used by existing agents.

As future work the inclusion of proactive agents within the architecture will be investigated, mapping out possible improvements deriving from the adoption of goal-oriented behaviours with respect to the interoperability issue.

References

[1] U. Varshney, "Pervasive healthcare," *Computer*, vol. 36, no. 12, pp. 138–140, 2003.

[2] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. Müller, "Ambient intelligence in assisted living: enable elderly people to handle future interfaces," in *Universal access in human-computer interaction. Ambient interaction*. Springer, 2007, pp. 103–112.

[3] J. S. Ash, M. Berg, and E. Coiera, "Some unintended consequences of information technology in health care: the nature of patient care information system-related errors," *Journal of the American Medical Informatics Association : JAMIA*, vol. 11, no. 2, pp. 104–12, 2004.

[4] D. A. Ludwick and J. Doucette, "Adopting electronic medical records in primary care: lessons learned from health information systems implementation experience in seven countries," *International journal of medical informatics*, vol. 78, no. 1, pp. 22–31, 2009.

[5] R. Haux, "Health information systems - past, present, future." *International journal of medical informatics*, vol. 75, no. 3-4, pp. 268–81, 2006.

[6] M. E. Frisse, K. B. Johnson, H. Nian, C. L. Davison, C. S. Gadd, K. M. Unertl, P. A. Turri, and Q. Chen, "The financial impact of health information exchange on emergency department care," *Journal of the American Medical Informatics Association*, vol. 19, no. 3, pp. 328–333, 2012.

[7] D. Blumenthal, "Stimulating the adoption of health information technology," *New England Journal of Medicine*, vol. 360, no. 15, pp. 1477–1479, 2009.

[8] P. Ferronato, S. Lotti, and D. Berardi, "Strategia architetturale per la Sanità Elettronica," Tech. Rep., 2006.

[9] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979 – 991, 2009.

[10] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[11] J. Nealon and A. Moreno, "Agent-based applications in health care," in *Applications of Software Agent Technology in the Health Care Domain*, ser. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhuser Basel, 2003, pp. 3–18.

[12] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing cloud computing and android os," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 1037–1040.

[13] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, "A Cloud Computing Solution for Patient's Data Collection in Health Care Institutions," in *2010 Second International Conference on eHealth, Telemedicine, and Social Medicine*, no. ii. IEEE, 2010, pp. 95–99.

[14] V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Building Interoperable Health Information Systems Using Agent and Workflow Technologies," in *Medical Informatics in a United and Healthy Europe*, 2009, pp. 180–184.

[15] D. Capozzi and G. Lanzola, "An Agent-Based Architecture for Home Care Monitoring and Education of Chronic Patients," in *2010 Complexity in Engineering*, vol. 9. IEEE, 2010, pp. 138–140.

[16] M. Lyell and X. Liu, "Software agent application to support the patient-at-home," in *2012 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2012, pp. 97–103.

[17] M. T. Nguyen, P. Fuhrer, and J. Pasquier-Rocha, "Enhancing e-health information systems with agent technology," *International journal of telemedicine and applications*, vol. 2009, 2009.

[18] R. Heeks, "Health information systems: failure, success and improvisation," *International journal of medical informatics*, vol. 75, no. 2, pp. 125–37, 2006.

[19] M. B. Buntin, M. F. Burke, M. C. Hoaglin, and D. Blumenthal, "The benefits of health information technology: A review of the recent literature shows predominantly positive results," *Health Affairs*, vol. 30, no. 3, pp. 464–471, 2011.

[20] D. M. Lopez and B. Blobel, "Architectural approaches for hl7-based health information systems implementation," *Methods of Information in Medicine*, vol. 49, no. 2, p. 196, 2010.

[21] A. W. Kushniruk, D. W. Bates, M. Bainbridge, M. S. Househ, and E. M. Borycki, "National efforts to improve health information system safety in canada, the united states of america and england," *International Journal of Medical Informatics*, vol. 82, no. 5, pp. 149 – 160, 2013.

[22] H. Takabi, J. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *Security Privacy, IEEE*, vol. 8, no. 6, pp. 24–31, 2010.

[23] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, 2007.

[24] L. Palazzo, A. F. Dragoni, A. Claudi, and G. Dolcini, "A multi-agent approach for health information systems domain," in *Proc. of the 1st Workshop on Artificial Intelligence and NetMedicine (NetMed)*, 2012.

[25] M. Ciampi, G. De Pietro, C. Esposito, M. Sicuranza, and P. Donzelli, "On federating Health Information Systems," in *International Conference in Green and Ubiquitous Technology*, no. March 2001, 2012, pp. 139–143.

[26] Italian Health Ministry, "Specifiche tecniche per la creazione del "Profilo Sanitario Sintetico" secondo lo standard HL7-CDA Rel. 2, Tavolo Permanente per la Sanità Elettronica," Tech. Rep., 2010.

[27] W3C Working Group, "Web Services Architecture. W3C Working Group Note 11 February 2004," 2004. [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

Integrated Analysis and Synthesis of Pedestrian Dynamics: First Results in a Real World Case Study

Sultan D. Khan, Luca Crociani, Giuseppe Vizzari
Complex Systems and Artificial Intelligence research center
Università degli Studi di Milano–Bicocca, Milano, Italy
{sultan.khan, luca.crociani, vizzari}@disco.unimib.it

***Abstract*—The paper introduces an agent-based model for the simulation of crowds of pedestrians whose main innovative element is the representation and management of an important type of social interaction among the pedestrians: members of groups, in fact, carry out of a form of interaction (by means of verbal or non-verbal communication) that allows them to preserve the cohesion of the group even in particular conditions, such as counter flows, presence of obstacles or narrow passages. The paper formally describes the model and presents its application to a real world scenario in which an analysis of the impact of groups on the overall observed system dynamics was performed. The simulation results are compared to empirical data and they show that the introduced model is able to produce quantitatively plausible results in situations characterised by the presence of groups of pedestrians.**

I. Introduction

The simulation of pedestrians and crowds is a consolidated and successful application of research results in the more general area of computer simulation of complex systems. Relevant contributions to this area come from disciplines ranging from physics and applied mathematics to computer science, often influenced by anthropological, psychological, sociological studies. The quality of the results provided by simulation models was sufficient to lead to the design and development of commercial software packages, offering useful functionalities to the end user (e.g. CAD integration, CAD-like functionalities, advanced visualisation and analysis tools) in addition to a simulation engine[1].

The last point is a crucial and critical element of this kind of research effort: computational models represent a way to formally and precisely define a computable form of theory of pedestrian and crowd dynamics. However, these theories must be validated employing field data, acquired by means of experiments and observations of the modeled phenomena, before the models can actually be used for sake of prediction. This paper represents a step in this direction, since it presents the application of methods from computer vision field for performing automated analysis on pedestrian dynamics, which are mainly aimed at the validation of an agent-based model for its simulation. The paper breaks down as the following. Description of the state-of-art of modelling and analysis of crowd dynamics is presented in Sec.II. Experimental methods for the automated analysis are described in Sec. III), while the real world case study used for their application is described in Sec. V-A. Then, the agent-based model for the simulation is presented in Sec. IV). First results of the analysis methods are described in Sec. V, while a discussion on the possibilities to exploit these data for sake of validation of the simulation results are presented in Sec. VI. Conclusions and future developments end the paper.

[1]See http://www.evacmod.net/?q=node/5 for a large list of pedestrian simulation models and tools.

II. Related Works

A. Synthesis

Pedestrian models can be roughly classified into three main categories that respectively consider pedestrians as *particles subject to forces*, particular *states of cells* in which the environment is subdivided in Cellular Automata (CA) approaches, or *autonomous agents* acting and interacting in an environment. The most widely adopted particle based approach is represented by the *social force model* [1], which implicitly employs fundamental proxemic concepts like the tendency of a pedestrian to stay away from other ones while moving towards his/her goal. *Cellular Automata* based approaches have also been successfully applied in this context: in particular, the floor-field model [2], in which the cells are endowed with a discretised gradient guiding pedestrians towards potential destinations. Finally, works like [3] essentially extend CA approaches, separating the pedestrians from the environment and granting them a behavioural specification that is generally more complex than what is generally represented in terms of a simple CA transition rule, but they essentially adopt similar methodologies. The resulting models are *agent–based*, since pedestrians are not merely states of cell. Along this direction of endowing models of more complicated behavioural models, relevant innovative studies regard social aspects and the transfer of emotions in crowds (see, e.g., [4]).

A recent survey of the field by [5] and by a report commissioned by the Cabinet Office by [6] made clear that, even after the substantial research that has been carried out in this area, there is still much room for innovations in models improving their performances both in terms of *effectiveness* in modelling pedestrians and crowd phenomena, in terms of *expressiveness* of the models (i.e. simplifying the modelling activity or introducing the possibility of representing phenomena that were still not considered by existing approaches), and in terms of *efficiency* of the simulation tools. Research on models able to represent and manage phenomena still not considered or properly managed is thus still lively and important. One of the aspects of crowds of pedestrians that has only been recently considered is represented by the implications of the presence of groups. A small number of recent works represent a relevant effort towards the modeling of groups, respectively in particle-based [7] (extending the social force model), in CA-based [8] (with ad-hoc approaches) and in agent-based approaches [9], [10] (introducing specific behavioral rules for managing group oriented behaviors): in all these approaches, groups are modeled by means of additional contributions to the overall pedestrian behaviour representing the tendency to stay close to other group members. However, the above approaches only mostly deal with small groups in relatively low density conditions; those dealing with relatively large groups (tens of pedestrians) were not validated against real data.

B. Automated Analysis

1) Dominant Flows Motion Detection: Crowd flow segmentation has multiple benefits: 1) enables clutter free visualization of moving groups 2) independence from detection and tracking 3) provide input for the pedestrian simulation models. Automatic analysis of the crowd has become the center of focus for most of researchers in computer vision. Detecting pedestrians and tracking are traditional ways of crowd analysis. Most algorithms developed for object detection and tracking work well in low density crowds where the number of people are less than twenty but in density crowds where the amount of people exceeds hundreds of thousand, detection and tracking of individuals are almost impossible due to multiple occlusions. Therefore, the research has focused on gathering global motion information at higher scale. Global analysis of dense group of moving people is often based on optical flow analysis.

A survey about the crowd analysis methods employed in computer vision is presented in [11]. An interdisciplinary framework for crowd analysis to improve simulation models of pedestrian flows is also presented in [12].

In [13] lagrangian coherent structures are detected by calculating finite-time scalar Lyapunov Exponent (FTLE) field over the phase space; these coherent structures represent different crowd motion patterns generating by moving in different directions. In [14] SIFT feature were instead used to detect dominant motion flows: flow vectors of SIFT features are calculated and then motion flow map is divided into small regions of equal size; in each region, dominant motion flows are estimated by clustering flow vectors. Crowd flow is estimated using multiple visual features reported in [15] where flow is estimated by the number of persons passing through a virtual trip wire and accumulate the total number of foreground pixels. Novel region growing scheme is adopted in [16] for crowd flow segmentation where translation flow is used to approximate the motion of crowd and region growing scheme is employed to segment the crowd flow. Min-cut/max flow algorithm is used in [17] for crowd flow segmentation. Histogram based crowd flow segmentation is reported in [18] where angle matrix of foreground pixels is segmented instead of optical flow foreground. The derivative curve of histogram is used to segment the flow.

2) People Counting in High Density Crowds: Estimating Crowd density and counting people is an important factor in crowd management. The increase of number of people in small areas may create problems like physical injury and fatalities. Hence early detection of the crowd can avoid these problems. Counting of the people moving in the crowd can provide information about the blockage at some point or even stampede. [19] proposed Bayesian model based segmentation to segment and count people but this method is not appropriate for high density crowds. [20] proposed blob features of moving objects to eliminate background and shadow from the image. [21] showed classification accuracy of 95% when crowd density is classified into four classes by using wavelet descriptors. [22] used texture descriptors called advanced local binary pattern descriptors to estimate crowd density estimation. [23] proposed a system that calculate the directional movement of the crowd and count the people as they cross some virtual line. [24] used specialized imaging system using infra-red imaging to count the people in the crowd. [25] have discussed in detail the concept of crowd monitoring using image processing through visual cameras. [26] used simple background subtraction from the static images to estimate the crowd density. Some other researchers [27], [28], [29] have also used the concept of background removal to estimate the crowd area. To estimate the crowd density using image processing, many researchers have used the information of texture, edges or some global or local features [30], [31].

III. Experimental Method for Automated Analysis

A. Motion Flow Characterisation

In this paper, we use Horn & Schunck [32] to calculate the dense optical flow field. The dense optical field calculated also contains the background information. We remove the background optical flow vectors by setting up a threshold. The optical flow vectors that are coherent and are a part of same flow are clustered. After clustering, some blobs appear which are removed by blob absorption method.

1) Motion Flow Composition: The motion flow field is a set of independent flow vectors in each frame and each flow vector is associated with its respective spatial location. The motion flow field is calculated by using optical flow methods. Given two images, F_t and F_{t+1} as input, we use Horn and Schunck [32] to compute dense optical flow. Consider a feature point i in F_t, its flow vector Z_i includes its location $X_i = (x_i, y_i)$ and its velocity $V_i = (v_{x_i}, v_{y_i})$, i. e. $Z_i = (X_i, V_i)$. We denote by $R_i(Z_i)$ as the magnitude of a flow vector and θ_i its angle or direction. The vector $M_i = (Z_i, R_i, \theta_i)$ summarises all the information associated to a feature i. Then $\{M_1, M_2, \ldots, M_k\}$ is the motion flow field of all the points of an image comprising $r \times c$ features such that $r \times c = k$, with r the number of rows and c the number of columns.

When computing dense optical flow we calculate the movement of all the pixels of an image, so it is usually the best

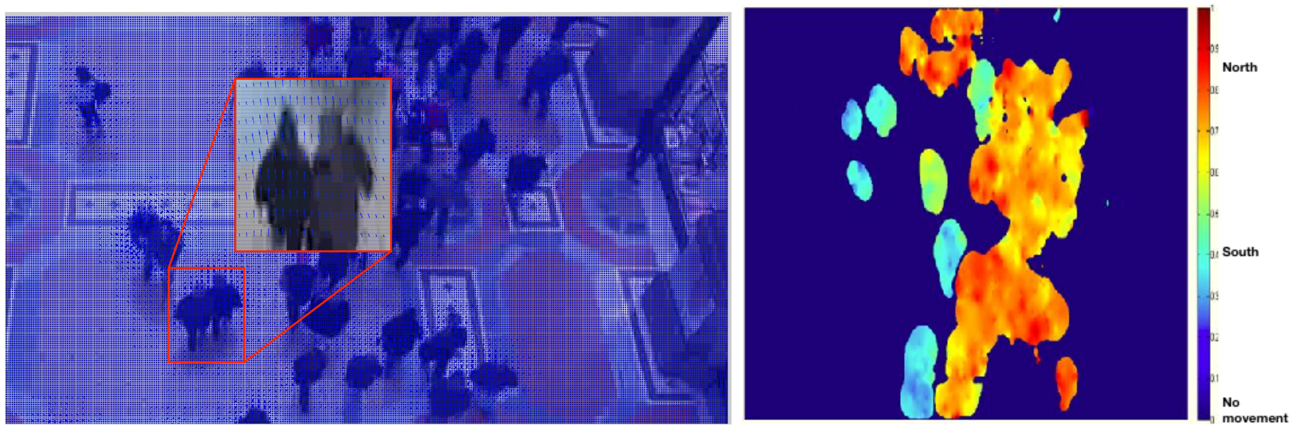


Fig. 1. Optical flow computed using the method described in [32], on the left, and compact representation of the movement direction of all pixels, on the right.

choice to remove background, to reduce computational costs without losing much information. To do so, a magnitude threshold is set to eliminate features characterised by a low magnitude that are considered background noise and that are not taken into account. This technique can also be used in computing coarse optical flow.

2) Motion Flow Field Segmentation: The motion flow field $\{M_1, M_2, \ldots, M_n\}$ is a matrix where each flow vector represents motion in specific direction as shown in Figure 1. Figure 1 does not show dominant motion patterns, so we can not infer any meaningful information about flow. Therefore, we need a method that automatically analyses the similarity among the flow vectors. We compute similarity among the flow vectors by applying similarity measure approach [33]. Similar vectors are grouped together to represent specific motion pattern by using clustering techniques [34]. This process of grouping vectors that represent specific motion pattern is called segmentation. After segmentation process, motion field is divided into small segments. Flow vectors that have similarity among them are clustered.

These blobs represent small clusters and resulted due to following reasons. First, if the objects move slowly, the inside and outside flow vectors of the objects are not same and as a result are classified into two different flows. Second, if the two opposite optical flow intersect, the optical flow at intersection point is ambiguous. Usually these small clusters or blobs are not the part of dominant motion flows. We adopt blob absorption approach, where these blobs are either absorbed by dominant cluster or by background. Let blob B_i of color C_i is the blob to be absorbed. Then find the edges of the blob by using Canny et al. [33] edge detector. For any point $p(x, y)$ on the edge of blob, we search 2×2 neighborhood of edge point. If any of neighborhood points has different color C_j that represents the dominant motion or background, then change the color of blob B_i to C_j, the color that represents dominant flow or background.

B. People Counting in a High Density Situations

In this paper, we have proposed a framework to count people in the extremely dense crowd where people are moving at different speeds. Foreground segmentation is done by various methods of background subtraction namely, approximate median, and frame difference and mixture of Gaussian method. Time complexity is calculated for these techniques and approximate median technique is selected which fast and accurate. Blob analysis is done to count the people in the crowd and blob area is optimized to get the best counting accuracy. In this paper, we extract the foreground by using Gaussian mixture model and optical flow. After getting foreground objects we use blob analysis method and optimize the blobs area by comparing it with ground truth data. Experimental results shows 90% accuracy of our results.

1) Motion Segmentation: Motion segmentation is the most important pre-processing step for detecting the moving objects from the video. Traditionally in video surveillance with a fixed camera, researchers tend to find some sort of motion in the video. There are two part of such of videos, background and foreground part. The object in motion is the foreground part of the video and the rest static part is the background. Motion detection is used to extract foreground part from the video. Such kind of extraction is useful for detecting, tracking and understanding the behavior of the object. Traditionally, background subtraction method is used for extracting moving objects from the video frame where pixels in the currents frame that deviate significantly from the background are considered as part of moving objects. Such kinds of methods are usually prone to errors due to unpredicted and changing behavior of the pixels. In addition, this method cannot accurately detect fast moving or slow moving as well as multiple objects. Also these methods are affected by change in illumination in the video frame. Sometime change in illumination in static background will be detected as part of moving object. Such errors and noise must be removed from the foreground objects before applying blob analysis. In order to extract valid and accurate foreground objects, we employed both Gaussian mixture model and Horn and Schank optical flow[32].

2) Blob Area Optimisation: Blobs are the connected regions in a binary image. For blob detection, image is first converted to binary image. Then next step is finding the connected components in the binary image. After finding connected components in binary image, the next step is to measure the properties of each connected component (object) in a binary image. In this paper, we are interested in measuring the 'Area' of each connected components. Area is the number of pixels in the region. Each binary image has a lot of connected components of variable size. We are interested in finding those connected components having area greater than some specific value. Area of the connected component differs depending upon the distance of camera from the scene. If the distance between the camera and crowd is less, greater will be number of pixels in a connected component and hence greater will be the blob size of the object. Hence the first step in people counting is to decide the optimal area of connected component. For this purpose, we have used four initial frames whose ground truth is available. In the iterative approach, we change the area of the blob size and count the people. This count is then compared with the ground truth of the frame (actual number of people in the frame). For each frame, optimal area is found for which the people count error was minimum.

IV. Pedestrian Simulation Model

In this section the formalisation of the agent-based computational model will be discussed, by focusing on the definition of its three main elements: *environment*, *update mechanism* and *pedestrian behaviour*.

A. Environment

The environment is modelled in a discrete way by representing it as a grid of squared cells with $40\ cm^2$ size (according to the average area occupied by a pedestrian [35]). Cells have a state indicating the fact that they are vacant or occupied by obstacles or pedestrians: $State(c) : Cells \rightarrow \{Free,\ Obstacle,\ OnePed_i, TwoPeds_{ij}\}$.

The last two elements of the definition point out if the cell is occupied by one or two pedestrians respectively, with their own identifier: the second case is allowed only in a controlled way to simulate overcrowded situations, in which the density is higher than $6.25\ m^{-2}$ (i.e. the maximum density reachable by our discretisation).

The information related to the scenario[2] of the simulation are represented by means of *spatial markers*, special sets of cells that describe relevant elements in the environment. In particular, three kinds of spatial markers are defined: (i) $start$ areas, that indicate the generation points of agents in the scenario. Agent generation can occur in *block*, all at once, or according to a user defined *frequency*, along with information on type of agent to be generated and its destination and group membership; (ii) $destination$ areas, which define the possible targets of the pedestrians in the environment; (iii) $obstacles$, that identify all the non-walkable areas as walls and zones where pedestrians can not enter.

[2]It represents both the structure of the environment and all the information required for the realization of a specific simulation, such as crowd management demands (pedestrians generation profile, origin-destination matrices) and spatial constraints.

Space annotation allows the definition of virtual grids of the environment, as containers of information for agents and their movement. In our model, we adopt the *floor field* approach [2], that is based on the generation of a set of superimposed grids (similar to the grid of the environment) starting from the information derived from spatial markers. Floor field values are spread on the grid as a gradient and they are used to support pedestrians in the navigation of the environment, representing their interactions with static object (i.e., destination areas and obstacles) or with other pedestrians. Moreover, floor fields can be *static* (created at the beginning and not changed during the simulation) or *dynamic* (updated during the simulation). Three kinds of floor fields are defined in our model: (i) *path field*, that indicates for every cell the distance from one destination area, acting as a potential field that drives pedestrians towards it (static). One path field for each destination point is generated in each scenario; (ii) *obstacles field*, that indicates for every cell the distance from neighbour obstacles or walls (static). Only one obstacles field is generated in each simulation scenario; (iii) *density field*, that indicates for each cell the pedestrian density in the surroundings at the current time-step (dynamic). Like the previous one, the density field is unique for each scenario.

Chessboard metric with $\sqrt{2}$ variation over corners [36] is used to produce the spreading of the information in the path and obstacle fields. Moreover, pedestrians cause a modification to the density field by adding a value $v = \frac{1}{d^2}$ to cells whose distance d from their current position is below a given threshold. Agents are able to perceive floor fields values in their neighbourhood by means of a function $Val(f, c)$ (f represents the field type and c is the perceived cell). This approach to the definition of the objective part of the perception model moves the burden of its management from agents to the environment, which would need to monitor agents anyway in order to produce some of the simulation results.

B. Pedestrians and Movement

Formally, our agents are defined by the following triple: $Ped = \langle Id,\ Group,\ State\rangle$; where $State = \langle position, oldDir,\ Dest\rangle$, with their own numerical identifier, their group (if any) and their internal state, that defines the current position of the agent, the previous movement and the final destination, associated to the relative path field.

Before describing agent behavioural specification, it is necessary to introduce the formal representation of the nature and structure of the groups they can belong to, since this is an influential factor for movement decisions.

1) Social Interactions: To represent different types of relationships, two kinds of groups have been defined in the model: a *simple group* indicates a family or a restricted set of friends, or any other small assembly of persons in which there is a strong and simply recognisable cohesion; a *structured group* is generally a large one (e.g. team supporters or tourists in an organised tour), that shows a slight cohesion and a natural fragmentation into subgroups, sometimes simple.

Between members of a simple group it is possible to identify an apparent tendency to stay close, in order to guarantee the possibility to perform interactions by means of verbal or non-verbal communication [37]. On the contrary, in large groups people are mostly linked by the sharing of a common goal, and the overall group tends to maintain only a weak compactness, with a following behaviour between members. In order to model these two typologies, the formal representation of a group is described by the following: $Group : \langle Id, [SubGroup_1, \dots, SubGroup_m], [Ped_1, \cdots, Ped_n]\rangle$.

In particular, if the group is simple, it will have an empty set of subgroups, otherwise it will not contain any direct references to pedestrians inside it, which will be stored in the respective leafs of its three structure. Differences on the modelled behavioural mechanism in simple/structured groups will be analysed in the following section, with the description of the utility function.

2) Agent Behaviour: Agent behaviour in a single simulation turn is organised into four steps: *perception, utility calculation, action choice* and *movement*. The *perception* step provides to the agent all the information needed for choosing its destination cell. In particular, if an agent does not belong to a group (from here called *individual*), in this phase it will only extract values from the floor fields, while in the other case it will perceive also the positions of the other group members within a configurable distance, for the calculation of the *cohesion* parameter. The choice of each action is based on an utility value assigned to every possible movement according to the function $U(c) = \frac{\kappa_g G(c)+\kappa_{ob}Ob(c)+\kappa_s S(c)+\kappa_c C(c)+\kappa_i I(c)+\kappa_d D(c)+\kappa_{ov}Ov(c)}{d}$.

Function $U(c)$ takes into account the behavioural components considered relevant for pedestrian movement, each one is modelled by means of a function that returns values in range $[-1;+1]$, if it represents an *attractive* element (i.e. its goal), or in range $[-1;0]$, if it represents a *repulsive* one for the agent. For each function a κ coefficient has been introduced for its calibration: these coefficients, being also able to actually modulate tendencies based on objective information about agent's spatial context, complement the objective part of the perception model allowing agent heterogeneity. The purpose of the function denominator d is to constrain the diagonal movements, in which the agents cover a greater distance ($0.4 * \sqrt{2}$ instead of 0.4) and assume higher speed with respect to the non-diagonal ones.

The first three functions exploit information derived by local floor fields: $G(c)$ is associated to goal attraction whereas $Ob(c)$ and $S(c)$ respectively to geometric and social repulsion. Functions $C(c)$ and $I(c)$ are linear combinations of the perceived positions of members of agent group (respectively simple and structured) in an extended neighbourhood; they compute the level of attractiveness of each neighbour cell, relating to group cohesion phenomenon. Finally, $D(c)$ adds a bonus to the utility of the cell next to the agent according to his/her previous direction (a sort of *inertia* factor), while $Ov(c)$ describes the *overlapping* mechanism, a method used to allow two pedestrians to temporarily occupy the same cell at the same step, to manage high-density situations.

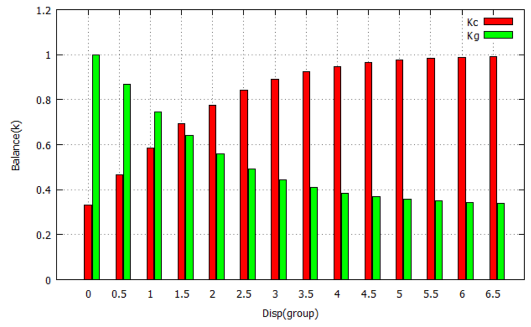


Fig. 2. Graphical representation of $Balance(k)$, for $k = 1$ and $\delta = 2.5$.

As we previously said, the main difference between simple and structured groups resides in the cohesion intensity, which in the simple ones is significantly stronger. Functions $C(c)$ and $I(c)$ have been defined to correctly model this difference. Nonetheless, various preliminary tests on benchmark scenarios show us that, used singularly, function $C(c)$ is not able to reproduce realistic simulations. Human behaviour is, in fact, very complex and can react differently even in simple situation, for example by allowing temporary fragmentation of simple groups in front of several constraints (obstacles or opposite flows). Acting statically on the calibration weight, it is not possible to achieve this dynamic behaviour: with a small cohesion parameter several permanent fragmentations have been reproduced, while with an increase of it we obtained no group dispersions, but also an excessive and unrealistic compactness.

In order to face this issue, another function has been introduced in the model, to adaptively balance the calibration weight of the three attractive behavioural elements, depending on the fragmentation level of simple groups:

$$Balance(k) = \begin{cases} \frac{1}{3}\cdot k + (\frac{2}{3}\cdot k \cdot DispBalance) & \text{if } k = k_c \\ \frac{1}{3}\cdot k + (\frac{2}{3}\cdot k \cdot (1 - DispBalance)) & \text{if } k = k_g \vee k = k_i \\ k & \text{otherwise} \end{cases}$$

where $DispBalance = tanh(\frac{Disp(Group)}{\delta})$, $Disp(Group) = \frac{Area(Group)}{|Group|}$, k_i, k_g and k_c are the weighted parameters of $U(c)$, δ is the calibration parameter of this mechanism and $Area(Group)$ calculates the area of the convex hull defined using positions of the group members. Fig. 2 exemplifies both the group dispersion computation and the effects of the $Balance$ function on parameters. The effective utility computation, therefore, employs calibration weights resulting from this computation, that allows achieving a dynamic and adaptive behaviour of groups: cohesion relaxes if members are sufficiently close to each other and it intensifies with the growth of dispersion.

After the utility evaluation for all the cells in the neighbourhood, the choice of action is stochastic, with the probability to move in each cell c as (N is the normalization factor): $P(c) = N \cdot e^{U(c)}$. On the basis of $P(c)$, agents move in the resulted cell according to their set of possible actions, defined as list of the eight possible movements in the Moore neighbourhood, plus the action to keep the position (indicated as X): $A = \{NW, N, NE, W, X, E, SW, S, SE\}$.

C. Time and Update Mechanism

Time is also discrete: an initial definition of the duration of a time step was set to $0.31\ s$. This choice, considering the side of the cell (40 cm), generates a linear pedestrian speed of about $1.3\ m/s$, which is in line with the data from the literature representing observations of crowd in normal conditions [35].

Regarding the update mechanism, three different strategies are usually considered in this context [38]: *ordered sequential*, *shuffled sequential* and *parallel* update. The first two strategies are based on a sequential update of agents, respectively managed according either to a *static* list of priorities that reflects their order of generation or a *dynamic* one, shuffled at each time step. On the contrary, the parallel update calculates the choice of movement of all the pedestrians at the same time, actuating choices and managing collisions in a latter stage. In the model we adopted the parallel update strategy, that is usually considered more realistic due to consideration of conflicts arisen for the movement in a shared space [39], [40].

With this update strategy, the agents life-cycle must consider that, before carrying out the *movement* execution, potential conflicts[3] must be solved. The overall simulation step therefore follows a three step procedure: (i) *update of choices* and *conflicts detection* for each agent; (ii) *conflicts resolution*, that is the resolution of the detected conflicts between agent intentions; (iii) *agents movement*, that is the update of agent positions exploiting the previous conflicts resolution, and *field update*, that is the computation of the new density field according to the updated positions of the agents.

The resolution of conflicts employs an approach essentially based on the one introduced in [40], based on the notion of friction. Let us first consider that conflicts can involve two or more pedestrians: in case more than two pedestrians involved in a conflict for the same cell, the first step is to block all but two of them, randomly chosen, reducing the problem to a simple case. To manage a simple conflict, another random number $\in [0; 1]$is generated and compared to two thresholds, $frict_l$ and $frict_h$, with $0 < frict_l < frict_h \leq 1$: the outcome can be that all agents are blocked when the extracted number is lower than $frict_l$, only one agent moves (chosen randomly) when the extracted number is between $frict_l$ and $frict_h$ included, or even two agents move when the number is higher than $frict_h$ (in this case pedestrian overlapping occurs). For our tests, the values of the thresholds make it quite relatively unlikely the resolution of a simple conflict with one agent moving and the other blocked, and much less likely their overlapping.

[3]essentially related to the simultaneous choice of two (or more) pedestrians to occupy the same cell

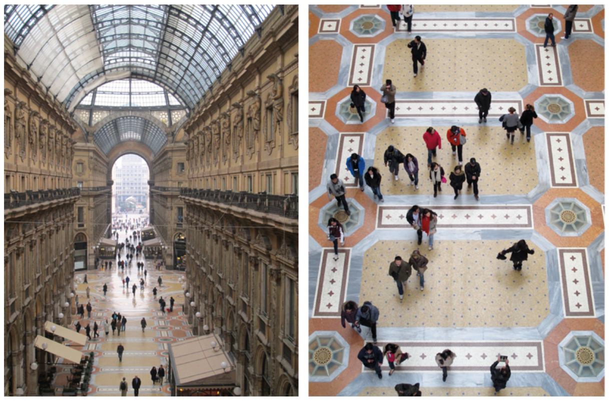


Fig. 4. From the left: an overview of the Vittorio Emanuele II gallery and the quasi-zenithal of passerby within the walkway.

V. Experimental Results

This section discuss about the qualitative analysis of the results obtained from experiments. We carried out our experiments on a PC of 2.6 GHz (Core i5) with 4.0 GB memory and video by Bandini et al [41], whose scenario is described with the following.

A. The Analysed Scenario

The survey was performed the last 24th of November 2012 from about 2:50 pm to 4:10 pm. It consisted in the observation of the bidirectional pedestrian flows within the Vittorio Emanuele II gallery (see Fig. 4), a popular commercial-touristic walkway situated in the Milan city centre (Italy). The gallery was chosen as a crowded urban scenario, given the large amount of people that pass through it during the weekend for shopping, entertainment and visiting touristic-historical attractions in the centre of Milan.

The team performing the observation was composed of four people. Several preliminary inspections were performed to check the topographical features of the walkway. The balcony of the gallery, that surrounds the inside volume of the architecture from about ten meters in height, was chosen as location thanks to possibility to (i) position the equipment for video footages from a quasi-zenithal point of view and (ii) to avoid as much as possible to influence the behaviour of observed subjects, thanks to a railing of the balcony partly hiding the observation equipment. The equipment consisted of two professional full HD video cameras with tripods. The existing legislation about privacy was consulted and complied in order to comply with ethical issues about the privacy of people recorded within the pedestrian flows.

B. Automated Analysis Experimental Results

We computed optical flow at a coarser resolution to reduce computational time as shown in Figure 1(a); let the output of optical flow be a binary image F_b. The gaussian mixture model was applied to the same sample frame to compute foreground; let the output of the Gaussian mixture model be called F_{gm}. Later on, F_{gm} and F_b were put into logical AND

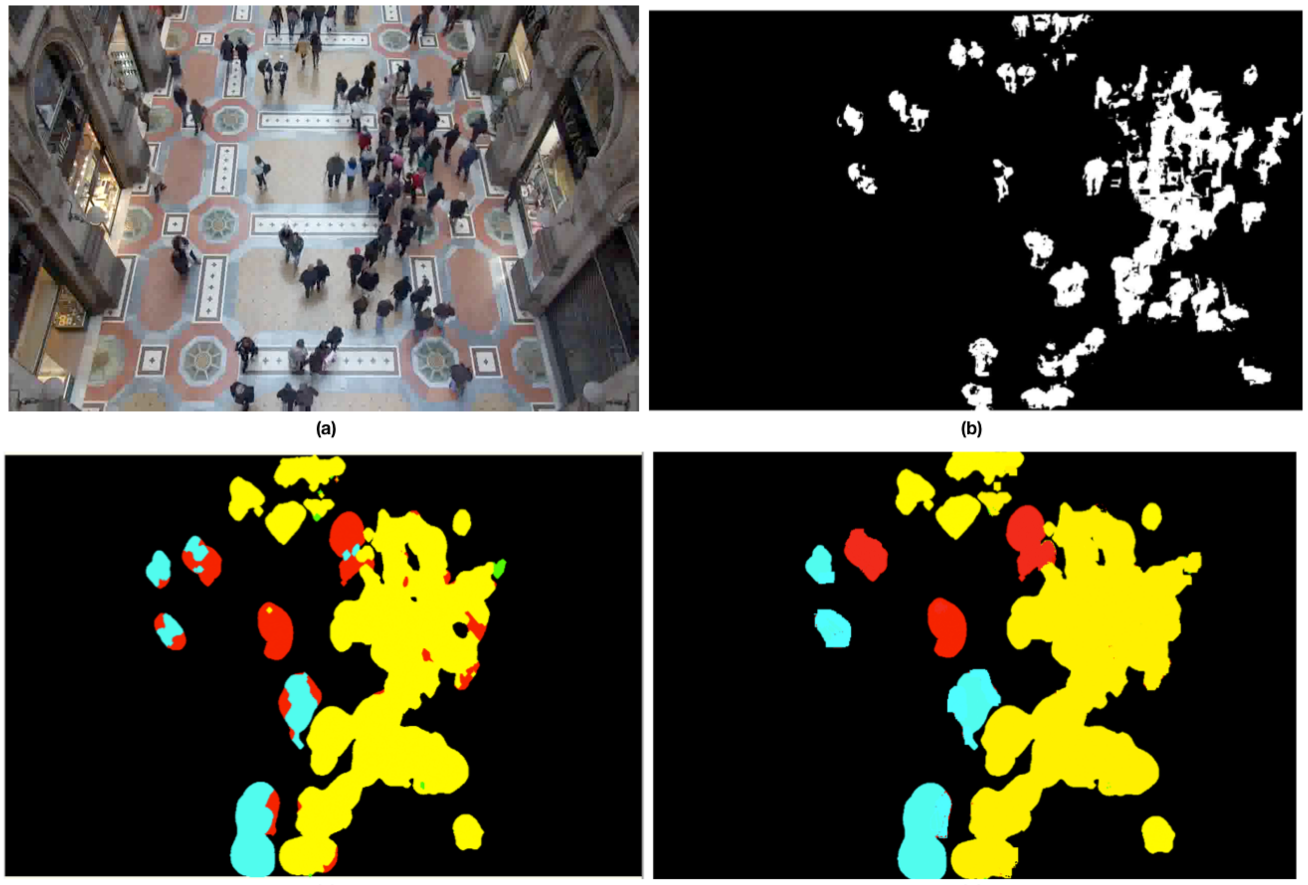


Fig. 3. Sample frame (a); its foreground image (F_f), computed by GMM and optical flow (b); the result of clustering flow vectors (c) and the result after applying blob absorption (d).

to extract foreground image F_f as shown in Figure 3(b). After computing optical flow and generating foreground image, similarity among the flow vectors in F_b was determined by using a similarity measure. Similar flow vectors were clustered to represent a specific motion pattern. Blob absorption method was applied to remove small clusters. Blob analysis method was applied on foreground image (F_f) to count the number of people.

Figure 3(c) and (d) show clustering result, flow vectors which satisfy similarity measures are combined into one cluster. Each cluster in the figure is color coded representing different motion patterns. Some blobs appear due to problems discussed in section III-A which are removed by the blob absorption method. Figure 3(d) shows a more refined version of motion patterns. As we can see from the Figure 3(c) and (d), there is a dominant motion towards North, a minor but still significant motion towards South, little motion towards West and almost negligible motion towards East.

Such little motion makes small clusters, which are usually absorbed by the blob absorption method, as shown in Figure 3(d). The results shows that large number of people moving towards North while there is less movement in other directions which is further illustrated in Figure 5.

Figure 5(a) shows ground truth calculated manually by using the Ground Truth Annotation (GTA) tool: 62 people were manually counted. After blob analysis, instead, 52 people were automatically detected in the whole scene as shown in Figure 5(b). Figure 5(c) shows instead the count of people moving in different directions: while studying dominant direction of crowd, analysis of speeds of crowd is important to understand overall crowd dynamics. Figure 5(d) shows the speed magnitude of all flow vectors. We use color codes to represent the magnitude of speeds. The bar scale represents different speeds where dark color or magnitude of 1 represent high speed while blue region shows zero magnitude. Figure 5(d) leads us to an important observation that the people moving alone are moving with high speed while others moving in groups move relatively slower. This kind of observations is important and provides a useful input to pedestrian simulation models.

Our proposed algorithm detects dominant motion patterns in the scene. Once motion pattern are detected, we can find the source and sink of each pattern. Sources refer to locations where objects appear and sinks are the location where objects disappear. Most of the scenes contain multiple sources and sinks: for example, a market place where multiple groups of

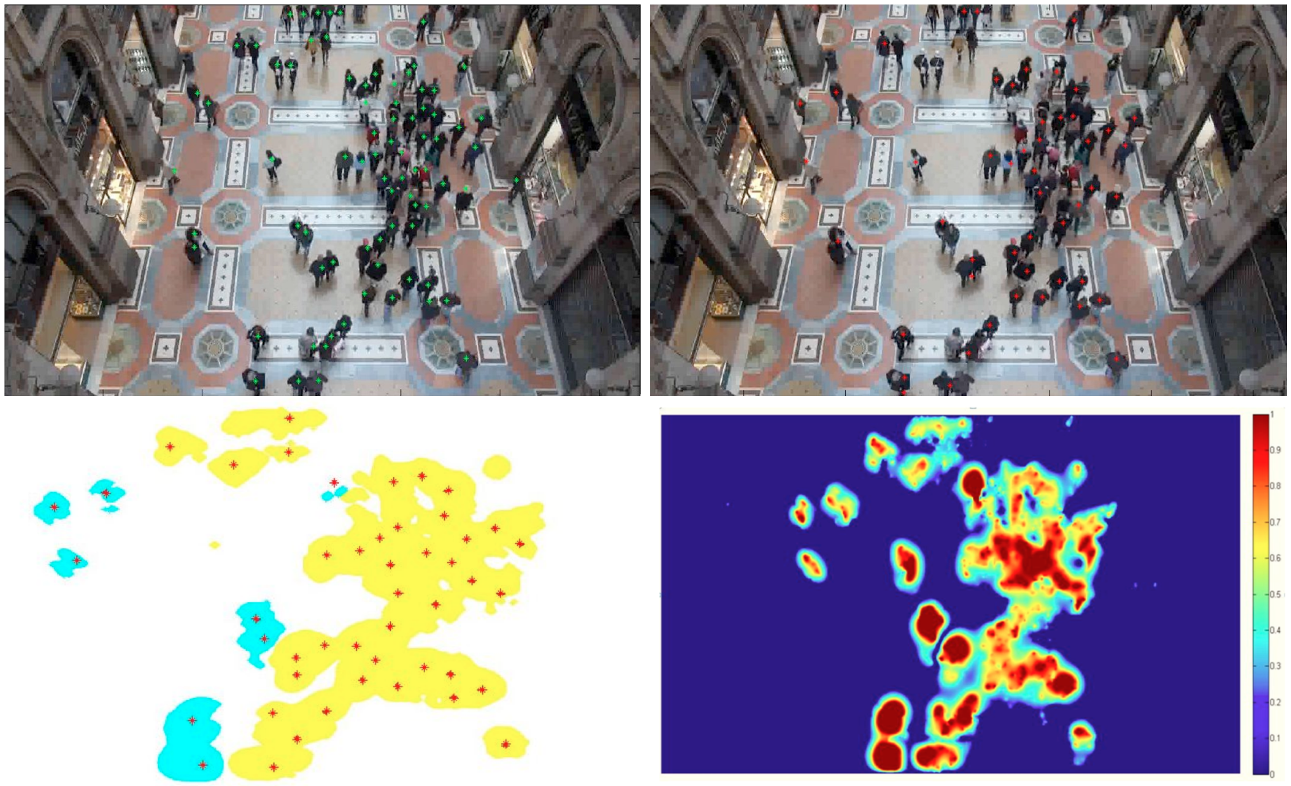


Fig. 5. The ground truth counting performed with GTA tool (a), people detected automatically by our algorithm (b); count of people in dominant flows (c) and the heat map describing speeds of pedestrians (d).

pedestrians move in distinct directions, originating multiple sources and sinks; similarly we could analyse flows in train stations or large floors of malls. The analysed video considers a situation in which, however, the flow of pedestrians is mostly on the North-South axis (referring to the video orientation). By analyzing sources and sinks of multiple motion patterns we can achieve information about the mostly visited or most attractive areas in the scene that can help us in understanding the behaviour of different pedestrian groups. In a transportation scenario, we could go as far as producing a so called origin–destination matrix which is an essential input for the creation of simulation scenarios. So, a generalisation of the presented work on dominant flows is object of current and future works.

VI. Discussion

The above described results of automated video analysis represent a first step in the direction of a more comprehensively integrated overall study of pedestrians and crowd dynamics. As of this moment, they still require some improvement to be directly applied, but in this section we will clarify some of the most immediate ways to exploit these results to support modeling and simulation.

A first employment of the above results is related to the actual configuration of the simulation scenario: qualitative analyses characterising the flow direction segmentation clarify that in the analysed portion of the environment most pedestrian movements are along the North-South axis (i.e. some pedestrians actually stop by a windows in one of the borders of the scenario or actually enter a shop, but their number is very low compared to the overall pedestrian flow). So, when designing the simulation environment we can exclude the presence of points of interest / attraction along the Eastern and Western borders of the environment: this was not obvious, since the analysed scenario comprises shops along the borders and since pedestrian behaviour in other areas of the gallery are quite different.

In particular, based on these data, we configured the environment as a large corridor with size 12.8 m $\times$ 13.6 m. At each end, one *start area* is placed for the agents generation, respecting the frequency of arrival observed in the videos; corridor ends also comprise a destination area corresponding to the start area positioned on the other end.

A second way to exploit data resulting from automated video analysis is represented by pedestrian counting and density estimation: the indication of the average number of pedestrians present in the simulated portion of the environment is actually important in configuring the start areas. In particular, in order to reproduce the counted number of pedestrians, also characterised according to their direction, we configured two different frequency profiles for the start areas which lead to achieve, respectively, 30 and 50 pedestrians in the environment on average. Should the automated analysis be able to discriminate different types of groups (i.e. individuals, couples, triples, etc.) this characterisation could

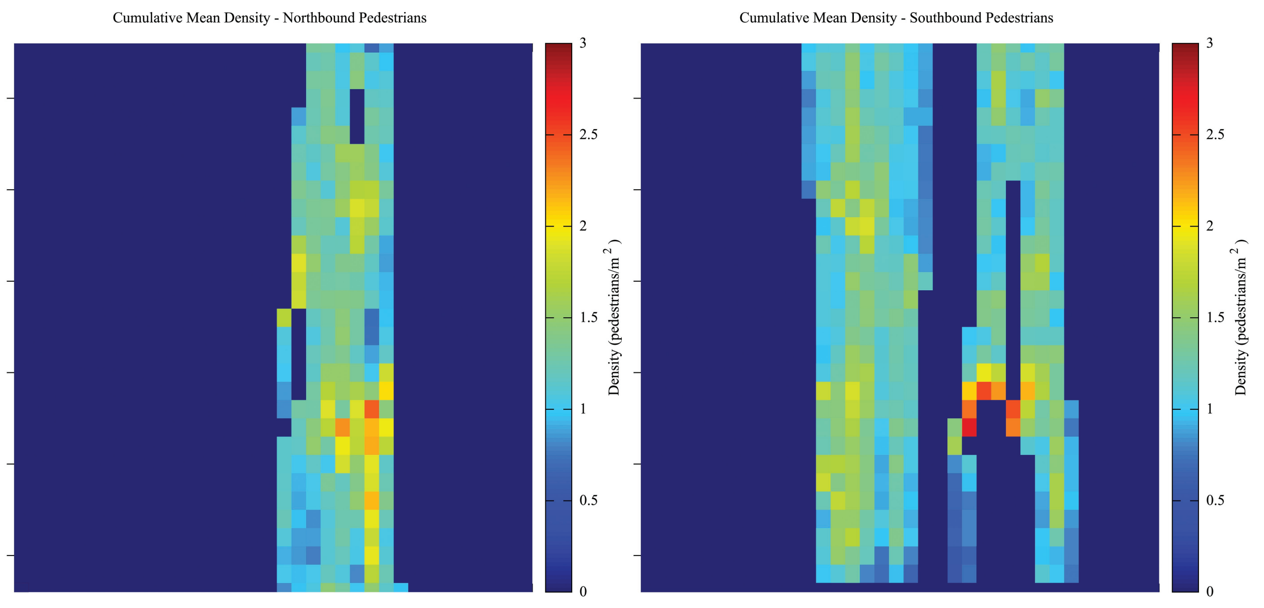


Fig. 6. Cumulative mean density maps calculated in a short period of the simulation (near 60 steps), which contains values of flow respectively towards North (a) and South (b).

further improve the starting areas configuration. The count of pedestrians in areas of the environment, and in particular portions of the overall analysed scene, could help generating Cumulative Mean Density maps [42], a heat-map diagram that can be used to validate simulation results. Examples of this type of result on the side of simulation (in the Galley scenario) are shown in Fig. 6, with an "instantaneous" (first 60 steps of the simulation, corresponding to 25 seconds of simulated time) calculation of the average *perceived*[4] local densities in the simulated environment, divided in the two directions of flow (i.e. North and South-bound). Even if it is only an example, this results is already able to provide useful information about the space utilization: while the flow towards North has remained relatively compact by forming one large lane, the one on the opposite direction has been divided in more lanes where little jamming are arisen, easily identifiable by the level of densities.

Finally, a third way to employ data resulting from automated video analysis is still related to the validation of simulation results and it still employs people counting data as a primary source. In particular, assuming that most of the counted pedestrians is actually in motion to enter or exit the monitored area (that could be a portion of the overall scene, like the northern border), we could estimated the instantaneous flow of pedestrians and average this value for certain time frames. The achieved measure can be compared to simulation results and this is particularly interesting in scenarios in which the density conditions have significant variations, since it could be possible to validate several points in the flow-density profile characterising the so called fundamental diagram [5] that can be achieved by means of simulation results.

[4]Values of local densities in each cell, contained in the *density* grid, are stored for the average calculation only when a pedestrian is located inside it.

Additional ways of employing other results of computer vision techniques for helping a simulation project, not necessarily discussed here or already employed for this analyses, could be done. For instance, once a source–sink analysis has been carried out, one could track a number of pedestrians completing a certain path and average out the travel time to have a reference value for evaluating simulations. However, the applicability and accuracy of tracking and many other techniques heavily depend on contextual factors like lightning conditions, changes in velocities and directions, but also crowding: a high density of pedestrians is very frequently always causing occlusions that can mislead the tracking algorithm. This work represents one first step in a more general research work aimed at contributing a fruitful interaction between pedestrian simulation and computer vision research producing (i) vertical results, namely techniques and case studies in specific contexts, and (ii) guidelines for the adoption of the most appropriate technique for a given context and situation.

VII. Conclusions

This paper has introduced the first results of a research effort putting together techniques of automated analysis of pedestrian and crowd dynamics and approaches towards the synthesis of these kind of phenomena. While in the present form the results of automated analysis mostly provide qualitative indications to the modeler, in the future they will represent also a quantitative empirical data for the initialization, calibration and validation of simulation models. The main contribution of the present work is represented by a systematic collaboration of automated analysis and simulation approaches in a specific and challenging real-world scenario, already producing useful indications that, however, will soon be improved for an even smoother and more quantitative integration. The main future directions are, on one hand, aimed at a more thorough quantification of the results of analyses and, on the other, at the identification and understanding the behaviour of pedestrian groups in the scene (e.g. source-sink analysis, converging to a point or dispersing, circling around points of reference).

References

[1] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, no. 5, pp. 4282–4286, May 1995.

[2] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3 - 4, pp. 507 – 525, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437101001418

[3] C. M. Henein and T. White, "Agent-based modelling of forces in crowds." in *Multi-Agent and Multi-Agent-Based Simulation, Joint Workshop MABS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, P. Davidsson, B. Logan, and K. Takadama, Eds., vol. 3415. Springer–Verlag, 2005, pp. 173–184.

[4] T. Bosse, M. Hoogendoorn, M. C. A. Klein, J. Treur, C. N. van der Wal, and A. van Wissen, "Modelling collective decision making in groups and crowds: Integrating social contagion and interacting emotions, beliefs and intentions," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 52–84, 2013.

[5] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, and A. Seyfried, "Evacuation dynamics: Empirical results, modeling and applications," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. Springer, 2009, pp. 3142–3176.

[6] R. Challenger, C. W. Clegg, and M. A. Robinson, "Understanding crowd behaviours: Supporting evidence," http://www.cabinetoffice.gov.uk/news/understanding-crowd-behaviours, University of Leeds, Tech. Rep., 2009.

[7] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PLoS ONE*, vol. 5, no. 4, p. e10047, 04 2010. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0010047

[8] S. Sarmady, F. Haron, and A. Z. H. Talib, "Modeling groups of pedestrians in least effort crowd movements using cellular automata," in *Asia International Conference on Modelling and Simulation*, D. Al-Dabass, R. Triweko, S. Susanto, and A. Abraham, Eds. IEEE Computer Society, 2009, pp. 520–525.

[9] R. A. Rodrigues, A. de Lima Bicho, M. Paravisi, C. R. Jung, L. P. Magalhães, and S. R. Musse, "An interactive model for steering behaviors of groups of characters," *Applied Artificial Intelligence*, vol. 24, no. 6, pp. 594–616, 2010.

[10] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. A. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, M. E. Taylor, X. Wang, A. Zilka, and M. Tambe, "Escapes - evacuation simulation with children, authorities, parents, emotions, and social comparison," in *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg, and Stone, Eds., 2011, pp. 457–464.

[11] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu, "Crowd analysis: a survey," *Mach. Vis. Appl.*, vol. 19, no. 5-6, pp. 345–357, 2008.

[12] M. Butenuth, F. Burkert, F. Schmidt, S. Hinz, D. Hartmann, A. Kneidl, A. Borrmann, and B. Sirmaçek, "Integrating pedestrian simulation, tracking and event detection for crowd analysis," in *ICCV Workshops*. IEEE, 2011, pp. 150–157.

[13] S. Ali and M. Shah, "A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *CVPR*. IEEE Computer Society, 2007.

[14] O. Ozturk, T. Yamasaki, and K. Aizawa, "Detecting dominant motion flows in unstructured/structured crowd scenes," in *ICPR*. IEEE, 2010, pp. 3533–3536.

[15] S. Srivastava, K. K. Ng, and E. J. Delp, "Crowd flow estimation using multiple visual features for scenes with changing crowd densities," in *AVSS*. IEEE Computer Society, 2011, pp. 60–65.

[16] S. Wu, Z. Yu, and H.-S. Wong, "Crowd flow segmentation using a novel region growing scheme," in *PCM*, ser. Lecture Notes in Computer Science, P. Muneesawang, F. Wu, I. Kumazawa, A. Roeksabutr, M. Liao, and X. Tang, Eds., vol. 5879. Springer, 2009, pp. 898–907.

[17] H. Ullah and N. Conci, "Crowd motion segmentation and anomaly detection via multi-label optimization," in *ICPR workshop on Pattern Recognition and Crowd Analysis*, 2012.

[18] C. Shiyao, L. Nianqiang, and L. Zhen, "Multi-directional crowded objects segmentation based on optical flow histogram," in *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 1. IEEE, 2011, pp. 552–555.

[19] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," in *CVPR (2)*. IEEE Computer Society, 2003, pp. 459–466.

[20] S. Yoshinaga, A. Shimada, and R.-i. Taniguchi, "Real-time people counting using blob descriptor," *Procedia-Social and Behavioral Sciences*, vol. 2, no. 1, pp. 143–152, 2010.

[21] L. Xiaohua, S. Lansun, and L. Huanqin, "Estimation of crowd density based on wavelet and support vector machine," *Transactions of the Institute of Measurement and Control*, vol. 28, no. 3, pp. 299–308, 2006.

[22] W. Ma, L. Huang, and C. Liu, "Advanced local binary pattern descriptors for crowd estimation," in *PACIIA (2)*. IEEE Computer Society, 2008, pp. 958–962.

[23] D. Yoshida, K. Terada, S. Oe, and J.-I. Yamaguchi, "A method of counting the passing people by using the stereo images," in *ICIP (2)*, 1999, pp. 338–342.

[24] K. Hashimoto, M. Morinaka, N. Yoshiike, C. Kawaguchi, and S. Matsueda, "People count system using multi-sensing application," in *Solid State Sensors and Actuators, 1997. TRANSDUCERS'97 Chicago., 1997 International Conference on*, vol. 2. IEEE, 1997, pp. 1291–1294.

[25] A. Davies, J. H. Yin, and S. Velastin, "Crowd monitoring using image processing," *Electronics Communication Engineering Journal*, vol. 7, no. 1, pp. 37–47, 1995.

[26] D. Roqueiro and V. A. Petrushin, "Counting people using video cameras," *IJPEDS*, vol. 22, no. 3, pp. 193–209, 2007.

[27] S. Velastin, J. Yin, A. Davies, M. Vicencio-Silva, R. Allsop, and A. Penn, "Analysis of crowd movements and densities in built-up environments using image processing," in *Image Processing for Transport Applications, IEE Colloquium on*. IET, 1993, pp. 8–1.

[28] ——, "Automated measurement of crowd density and motion using image processing," in *Road Traffic Monitoring and Control, 1994., Seventh International Conference on*. IET, 1994, pp. 127–132.

[29] A. Marana, S. Velastin, L. Costa, and R. Lotufo, "Estimation of crowd density using image processing," in *Image Processing for Security Applications (Digest No.: 1997/074), IEE Colloquium on*. IET, 1997, pp. 11–1.

[30] R. Ma, L. Li, W. Huang, and Q. Tian, "On pixel count based crowd density estimation for visual surveillance," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1. IEEE, 2004, pp. 170–173.

[31] S.-F. Lin, J.-Y. Chen, and H.-X. Chao, "Estimation of number of people in crowded scenes using perspective transformation," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 31, no. 6, pp. 645–654, 2001.

[32] B. K. P. Horn and B. G. Schunck, ""”determining optical flow”: A retrospective," *Artif. Intell.*, vol. 59, no. 1-2, pp. 81–87, 1993.

[33] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[34] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[35] U. Weidmann, "Transporttechnik der fussgänger - transporttechnische eigenschaftendes fussgängerverkehrs (literaturstudie)," Institut füer Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau IVT an der ETH Zürich, Literature Research 90, 1993.

[36] T. Kretz, C. Bönisch, and P. Vortisch, "Comparison of various methods for the calculation of the distance potential field," in *Pedestrian and Evacuation Dynamics 2008*, W. W. F. Klingsch, C. Rogsch, A. Schadschneider, and M. Schreckenberg, Eds. Springer Berlin Heidelberg, 2010, pp. 335–346. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04504-2_29

[37] M. Costa, "Interpersonal distances in group walking," *Journal of Nonverbal Behavior*, vol. 34, pp. 15–26, 2010, 10.1007/s10919-009-0077-y. [Online]. Available: http://dx.doi.org/10.1007/s10919-009-0077-y

[38] H. Klüpfel, "A cellular automaton model for crowd movement and egress simulation," Ph.D. dissertation, University Duisburg-Essen, 2003.

[39] A. Kirchner, A. Namazi, K. Nishinari, and A. Schadschneider, "Role of Conflicts in the Floor Field Cellular Automaton Model for Pedestrian Dynamics," in *2nd International Conference on Pedestrian and Evacuation Dynamics*, 2003.

[40] A. Kirchner, K. Nishinari, and A. Schadschneider, "Friction effects and clogging in a cellular automaton model for pedestrian dynamics," *Phys. Rev. E*, vol. 67, p. 056122, May 2003. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.67.056122

[41] S. Bandini, A. Gorrini, and G. Vizzari, "Towards an integrated approach to crowd analysis and crowd synthesis: a case study and first results," *CoRR*, vol. abs/1303.5029, 2013.

[42] C. Castle, N. Waterson, E. Pellissier, and S. Bail, "A comparison of grid-based and continuous space pedestrian modelling software: Analysis of two uk train stations," in *Pedestrian and Evacuation Dynamics*, R. D. Peacock, E. D. Kuligowski, and J. D. Averill, Eds. Springer US, 2011, pp. 433–446. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-9725-8_39

A Conceptual and Computational Model for Knowledge-based Agents in ANDROID

Fabio Sartori, Lorenza Manenti and Luca Grazioli
Department of Informatics, Systems and Communication,
University of Milano-Bicocca,
Viale Sarca 336/14, 20126, Milano, Italy
Email: {sartori, manenti}@disco.unimib.it,
l.grazioli3@campus.unimib.it

***Abstract*—Today ANDROID is the most popular mobile operating system in the world: the development of ANDROID, together with the performance improvement offered by modern PDAs, like smart-phones and tablets, has allowed many users to know new kinds of applications that were not accessible to them in the recent past.**

In this paper we present a framework for programming agents in the ANDROID world, based on the Knowledge Artifact notion to develop knowledge-based systems.

This framework has been modeled as a client–server architecture, with the aim to show how the implementation of agents modeled on the basis of Knowledge Artifacts can help everyone to design, implement and use decision support systems for a specific domain, with many potential benefits in their day-by-day activities.

The framework application will be presented in a prototype to support operators of Italian Fire Corps and Civil Protection Department in critical situations, like geographically distributed fires and earthquakes management.

I. Introduction

Nowadays ANDROID is the most popular mobile operating system in the world, and reaches out to touch peaks of diffusion, in some countries, more than 90% of the total smartphone market[1]. The development of ANDROID together with the performance improvement offered by modern PDAs, like smartphones and tablets, has allowed many users to keep in touch with new kinds of applications that were not accessible to them in the recent past.

Among them, applications related to the Agent Oriented Programming (AOP) paradigm [1] are particularly influenced by the wide diffusion of personal devices, thanks to their intrinsic mobile nature. Different open-source frameworks devoted to the development of agent-based programs, like JADE[2] [2], JASON[3] [3] and CArtAgO[4] [4] have been recently imported into ANDROID by means of the implementation of specific add-ons or ad-hoc frameworks (e.g. [5], [6]).

In this paper we present a framework for programming agents in the ANDROID world, making it transparent to the programmer, basing it on the Knowledge Artifact [7] notion. While the *artifact* concept is often described as *the result of a disciplined human activity, following rules based on training and experience*, Knowledge Artifacts [11] are artifact specializations devoted to represent expert knowledge in object–manufacturing or service–delivery fields. The final aim of our Knowledge Artifact is generating executable rule–based systems written in JESS[5], minimizing the knowledge engineering effort. To this scope, correlated tools for the representation of functional and structural knowledge (i.e. ontologies [8]), procedural knowledge (i.e. influence nets [9]) and experiential knowledge (i.e. task structures [10]) have been integrated into a unique conceptual and computational framework, providing the user with an opportune set of primitives for designing and implementing decision support systems without a deep knowledge on the specific language syntax.

This framework has been modeled as a client–server architecture, where the server is a knowledge-based agent having two tasks, the creation of the rule-based system according to the user (i.e. the expert) indications and the execution of it according to the data sent it by the client, that is a sort of reactive agent sending inputs and receiving outputs from the knowledge-based agent: in this way, it was possible to overcome the impossibility, at the current state of JESS implementation[6] to import JESS library under the ANDROID OS.

The main aim of this paper is to show how the implementation of agents modeled on the basis of Knowledge Artifacts can help everyone to design, implement and use decision support systems for a specific domain, with many potential benefits in their day-by-day activities.

The most interesting feature of the framework is its capability to act as a CAKE (Computer-Aided Knowledge Engineering) environment: the implementation of KBSs has been always conceived as a very specific task, which can be only conducted by knowledge engineers with the support of domain experts. Knowledge engineering methodologies, such as CommonKads [12] and MIKE [13], have been proposed in the past as standard and generalized solutions to overcome

[1]Data extracted from http://www.androidworld.it/2013/04/29/android-in-italia-al-625-e-nel-mondo-al-642-secondo-le-ultime-stime-153115/

[2]Available at http://jade.tilab.com/

[3]Available at http://jason.sourceforge.net/wp/

[4]Available at http://cartago.sourceforge.net/

[5]Acronym of Java Expert System Shell, available at http://herzberg.ca.sandia.gov

[6]See the discussion at http://jess.2305737.n4.nabble.com/JESS-Re-Jess-jar-on-Android-td3957868.html

the knowledge acquisition and representation bottlenecks, but addressed to users highly skilled in the design of complex software systems.

The rest of the paper is organized as follows: Section II will introduce the agent-based framework and the conceptual model of Knowledge Artifact it is based on, also discussing the computational model of the agents developed according to it. Section III will present a case study to show how it can be profitably used in a specific domain. In the end, in Section IV a discussion on how this model can be implemented considering the AOP paradigm will be presented along with a discussion on the future developments and works.

II. The Agent-Based Framework and the KA Notion

In this Section we first present the agent–based framework describing the main components of the model; then, we discuss the conceptual model and the relative implementation of the KA notion according to the agent–based model has been developed.

A. Agent-based Framework

In relationship with the client-server architecture, our framework is composed of two main elements:

- simple reactive agents $a_1, .., a_i, .., a_n$ located on mobile devices and that perceived and collected information on the “state of the world”. These information can be detected by means of device sensors (e.g. GPS, barometer, pressure altimeter and so on) and directly provided by the user of the system by means of an appropriate graphic user interface (GUI);
- a knowledge-based agent *KA-Agent* that is responsible for the management of its internal knowledge and the elaboration of information received from agents a_i. In particular the goal of a KA-Agent is twofold: (i) it has to interact with the domain expert by means of a graphic user interface (GUI), obtaining information about the domain in terms of concepts, relationships and rules, and creating the corresponding ontology, influence network and rule-based system; (ii) it has to interact with agents a_i and it has to elaborate the information they provide with the expert system previously created.

Fig. 1 graphically represents the components and the interactions in our framework: expert users interact with the KA-Agent GUI in order to create the expert system, while non-expert users interact with agent a_i GUI in order to provide information that will be used by KA-Agent in elaborating knowledge.

Since the KA-Agent represents the main relevant part of our system, in the follow we will focus on the KA notion upon which it is based on, considering the conceptual model and the relative implementation.

B. The Conceptual Model of Knowledge Artifact

In our approach, the Knowledge Artifact is described as a 3–tuple $\langle O, IN, TS \rangle$, where O is an ontology of the investigated domain, IN is an Influence Net to represent the causal dependencies among the ontology elements and TS are task structures to represent how one or more outputs can be produced by the system according to a rule–base system strategy.

In the KA model, the underlying ontology is a taxonomy: the root is the description of the problem to be solved, the inner nodes are system inputs or partial outputs and the leaves of the hierarchy are effective outputs of the system.

The Influence Net model is a structured process that allows to analyse complex problems of cause-effect type in order to determine an optimal strategy for the execution of certain actions, to obtain an optimal result. The Influence Net is a graphical model that describes the events and their causal relationships. Using information based on facts and experience of the expert, it is possible to analyze the uncertainties created by the environment in which we operate. This analysis helps the developer to identify the events and relationships that can improve or worsen the desired result. In this way you can determine the best strategy.

The Influence Net can be defined as a 4–tuple $\langle I, P, O, A \rangle$, where:

- I is the set of *input nodes*, i.e. the information needed to the KBS to work properly;
- P is the set or *partial output nodes*, i.e. the collection of new pieces of knowledge and information elaborated by the system to reach the desired output;
- O is the set of *output nodes*, i.e. the effective answers of the system to the described problem; outputs are values that can be returned to the user;
- A is the set of *arcs* among the nodes: an arc between two nodes specifies that a causal relationship exists between them; an arc can go from an input to a partial node or an output, as well as from partial node to another one or an output. Moreover, an arc can go from an output to another output. Every other kind of arcs is not permitted.

Finally, Task Structures allow to describe in a rule–based system way how the causal process defined by a given IN can be modeled. Each task is devoted to define computationally a portion of an Influence Net: in particular, *sub–tasks* are procedures to specify how a partial output is obtained, while *tasks* are used to explain how an output can be derived from one or more influencing partial outputs and inputs. A task cannot be completed until all the sub–tasks influencing it have been finished. In this way, the TS modeling allows to clearly identify all the levels of the system. The task and sub–task bodies are a sequence of rules, i.e. $LHS(LeftHandSide)- > RHS(RightHandSide)$ constructs.

Each LHS contains the conditions that must be verified so that the rule can be applied: it is a logic clause, which turns out to be a sufficient condition for the execution of the action indicated in the RHS. Each RHS contains the description of the actions to conduct as a result of the rule execution. The last step of our model is then the translation of all the task and sub–task bodies into production rules of a specific language (JESS in our case).

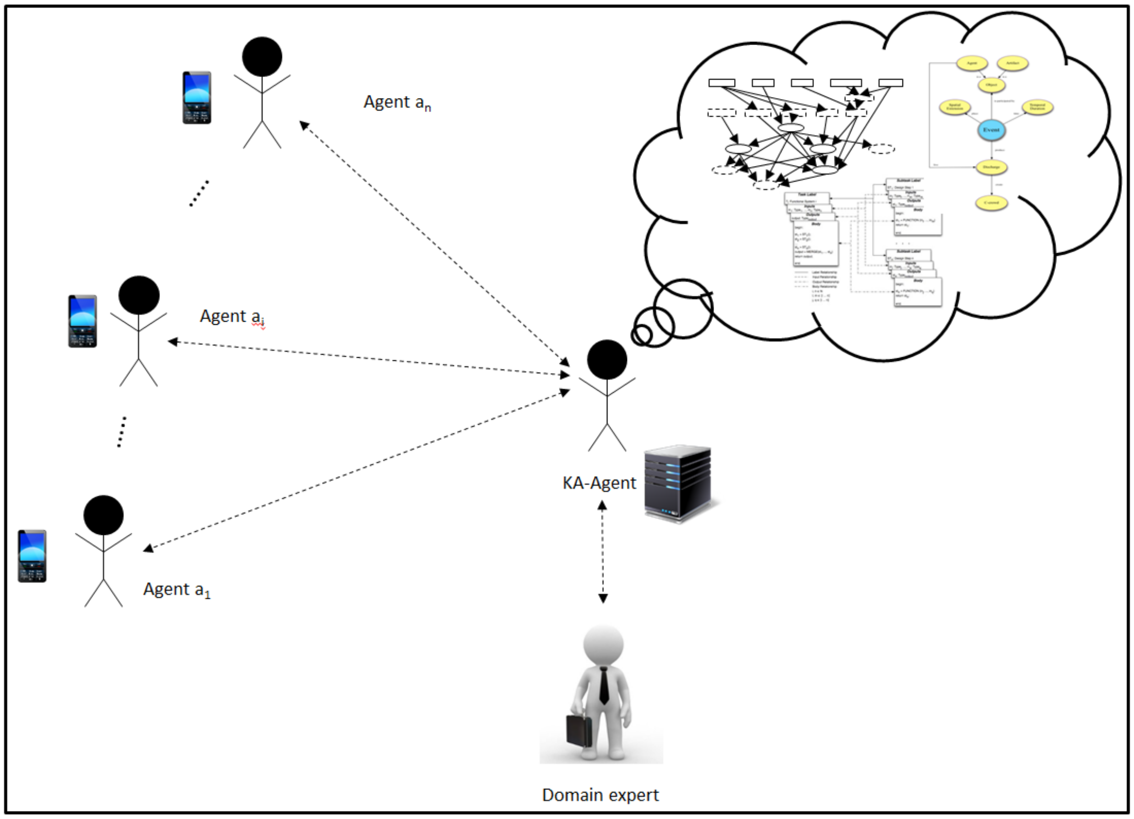


Fig. 1: Graphical representation of the agent-based model: agents a_i, KA-Agent, and interactions among them and with domain expert

C. Knowledge Artifact Implementation: the KA-Agent

The implementation of the different elements composing the knowledge engineering framework has exploited the XML language. A proper schema has been developed for each of them, as well as dedicated parsers to allow the user to interact with them. Following the conceptual model briefly introduced in the previous section, the first schema is the ontological one, defining opportune tags to specify *inputs*, where the name of the input can be put together with a description and a value for it. Moreover, it is possible to define an $\langle affects \rangle$ relationship for each input, in order to explain how it is involved in the next steps of the elaboration (i.e. which output or partial output does it contribute to state?).

A *partial output* (i.e. an inner node between an input and a leaf of the taxonomy) is limited by a specific pair of tags. The fields are the same as the input case, with the difference that a partial output can be influenced by another entity too: this is the sense of the $\langle influencedBy \rangle$ relationship. Finally, the $\langle output \rangle$ tag allows to describe completely an effective output of the system, i.e. a leaf of the taxonomy developed to represent the problem domain.

To produce an Influence Net, the taxonomy is bottom–up parsed, in order to identify the right flow from inputs to outputs by navigating the *influencedBy* relationships designed by the user. In this way, different portions of the under development system can be described. Outputs, partial outputs and inputs are bounded by arcs which specify the *source* and the *target* nodes.

Finally, an XML schema for the *task* (*subtask*) element of the framework have been developed. The parser produces an XML file for each output considered in the Influence Net. The tags *input* and *subtask* allows to define which inputs and partial outputs are needed to the output represented by the task to be produced. The *body* tag is adopted to model the sequence of rules necessary to process inputs and results returned by influencing subtasks: a rule is composed of an $\langle if \rangle$ ... $\langle do \rangle$ construct, where the if statement permits to represent the LHS part of the rule, while the do statement concerns the RHS part of the rule.

The XML files introduced so far have been incorporated into the KA–Agent knowledge base: they allow the definition of a rule–based system to solve problems in specific domains. The developed GUI permits the user to interact with the KA–Agent to design the underlying taxonomy, influence net and tasks/subtasks. Moreover, it is possible to transform the task into a collection of files containing rules written in the JESS language: Figure 2 shows a sketch of the supporting tool for this scope.

III. A Case Study

Once the KA–Agent has been created and programmed as a rule–based system, it is able to interact with one or more

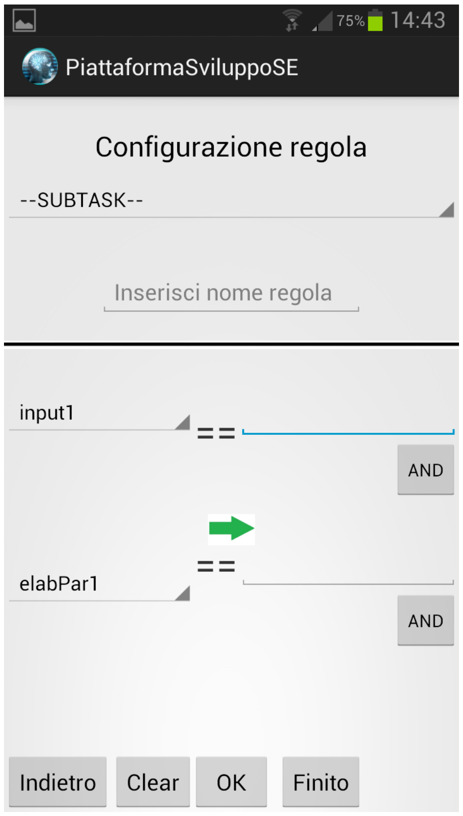


Fig. 2: The KA–Agent GUI for supporting domain experts in the creation of rule–based system from task structures

clients to receive inputs and send outputs. Clients are reactive agents in our model: they are spatially distributed and detect observations about the knowledge domain under investigation. They can send to the KA–Agent these observations in order to get suggestions about the action to take. These suggestions are elaborated by the KA–Agent according to its KA model. This scenario has been successfully tested in a case study inspired us by the *STOP*[7] handbook supplied to the Italian Fire Corps and Civil Protection Department of the Presidency of Council of Ministers for the construction of safety building measures for some building structures that have been damaged by an earthquake. After L'Aquila's earthquake in 2009, this document has been prepared in order to standardize the steps to follow in similar situations.

The structure of this document is suitable for the creation of a rule-based system, being modular and with specific and well defined cases. Using the created application, two knowledge models have been created (the first for the safety of the walls through rakers, the other for the safety of the openings through special scaffoldings).

A. The STOP–Agent: an ANDROID Client

Every operator involved in the emergency procedures to make safe buildings and infrastructures is provided with an ANDROID application on his/her smartphone: this application has been modeled as a reactive agent, namely the *STOP–Agent*, communicating with the KA–agent via the client–server architecture introduced above: Figure 3 shows the interface for its initialization. Each STOP–Agent sends to the KA–agent data about the conditions of the site it is analyzing: according to the STOP handbook, these data allow to make considerations about the real conditions of the building walls after the earthquake in order to understand which raker or scaffolding to adopt.

Exploiting the ANDROID primitives, it has been possible to create a stable mechanism for the communication with the server. In particular, the following tools were useful to implement the STOP–Agent:

- *activities*: a class that extends an *Activity* class is responsible for the communication with the user, to support him/her in setting the layout, assigning the listeners to the various widgets (ANDROID's graphical tools) and setting the context menu;
- *listener*: a class that implements the interface *OnClick-Listener* is a listener. An instance of this object is always associated with a widget;
- *asyncTask*: a class that extends *AsyncTask* is an asynchronous task that performs some operations concurrently with the execution of the user interface (for example the connection to a server must be carried out in a AsyncTask

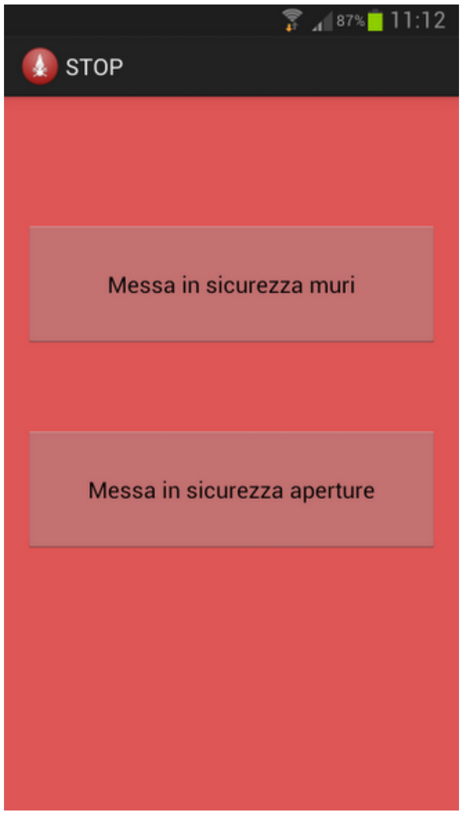


Fig. 3: The STOP–Agent GUI for the activation of the two application knowledge models

[7]Acronym of *Schede Tecniche di Opere Provvisionali*, see http://www.vigilfuoco.it/aspx/notizia.aspx?codnews=8293 for details.

instance, not in an Activity one);

The typical mechanism to interface the client and the server is the following one: the Activity object prepares the layout and sets the widgets' listeners and a container with the information useful for the server; then, it possibly starts the AsyncTask instance for sending the correct request to the server, passing to it the previously created container. Before starting the asynchronous task, in most cases, the listener activates a dialog window that locks the user interface in waiting for the communication with the server; the AsyncTask predisposes the necessary Sockets for the communication and then performs its request to the server, sending the information about the case study observation (see Figure 4) enclosed in the container. Before concluding, it closes (dismisses) the waiting dialog window.

B. The KA–Agent: a JAVA Server

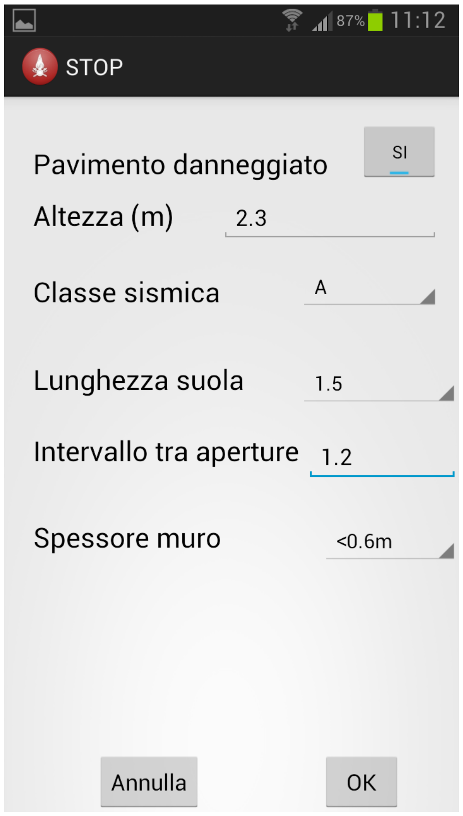
The KA–Agent creates an instance of the KA model for each active STOP–Agents in communication with it. Then, it executes the model according to the rule–based system previously generated and sends answers to the STOP–Agent that will be able to take the proper action, as shown in Figure 5.

The design of the server exploits an existent framework for the rule-based systems creation. This framework allows to produce a *.clp* file runnable under JESS and that contains the rules describing the behavior of the rule-based system. A *Controller* class has been added to this framework to build up the interface between all the server's classes and the preexistent knowledge engineering environment.

The server, once activated, can accept both requests for the creation of a new system and for the resolution of problems on the basis of existing rule-based systems. To do this, concurrent programming is used: the server manages the different requests concurrently, through an opportune thread, namely *MonitorThread*. A MonitorThread instance starts the thread in charge of listening the requests for the creation of a new rule-based system (i.e. *GestoreThread*). Moreover, MonitorThread allows to properly manage the ports on which other threads will interface, and provides methods necessary for their correct startup. Another thread, namely *ManagingThread* is instanced by GestoreThread for the use of previously created rule–based systems. This thread manages the .clp files archive, being sure that inputs and outputs are correctly received and sent by the right system and JESS' libraries are correctly invoked.

IV. Conclusion and Future Works

This paper has presented an ongoing research project aiming to design and implement tools for supporting the user in the development of knowledge-based systems. This framework is based on the Knowledge Artifact conceptual model and is general enough to be adopted in different contexts and



Pavimento danneggiato SI

Altezza (m) 2.3

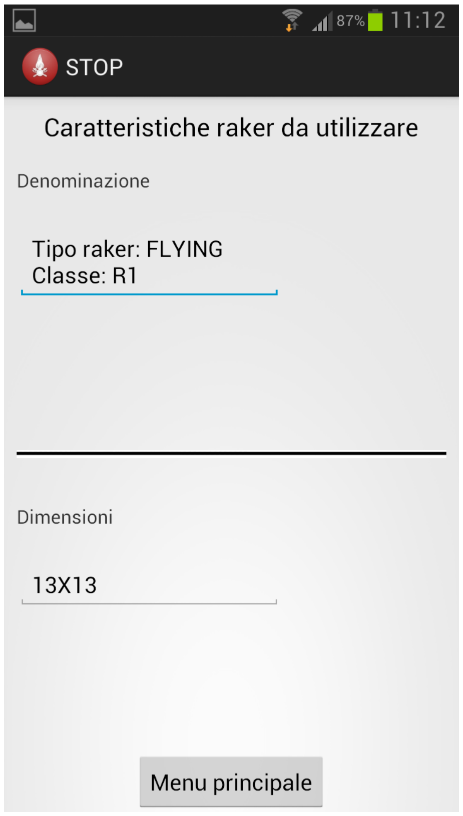
Classe sismica A

Lunghezza suola 1.5

Intervallo tra aperture 1.2

Spessore muro <0.6m

Fig. 4: The STOP–Agent GUI for sending information to the KA–Agent



Caratteristiche raker da utilizzare

Denominazione

Tipo raker: FLYING
Classe: R1

Dimensioni

13X13

Fig. 5: The KA–Agent sends characteristics on raker to use to the STOP–Agent, in order to make safe a wall

programming paradigms.

In particular, we have integrated it into the AOP model, according to a client–server architecture, to design and implement KBSs remotely exploiting the potentialities of ANDROID OS: in this way, the framework can be executed from every kind of PDAs, like smartphones and tablets, with the possibility to create an ad–hoc KBS when necessary. The framework was applied in a potential collaboration with the Italian Fire Corps and the Civil Protection Department of the Presidency of Council of Ministers, to provide each firemen with tools to understand how to operate in critical situations, like geographically distributed fires and earthquakes management.

The developed system has shown an excellent level of performance, especially about the client side of the project. The ANDROID application requirements are minimal, especially considering the Internet consumption data (an actually critical argument). On the other hand, the memory request by the server is definitely high. This last statement also opens the discussion on future developments which are certainly ample and varied. The server is, without doubt, the component that needs more attention; some modifications that could be made range from the addition of new JESS features to the improvement of memory management and a more efficient management of concurrency.

In the introduction we have already discussed the connection between AOP paradigm and the connection with ANDROID world: an idea to implement the model is to use JADE framework, for several reasons. First of all, JADE framework is a software framework fully implemented in Java language as our native project, and it implements FIPA [2] specifications to support communications and interactions among agents (and we need to model interactions between simple agents a_i and KA-Agent).

More in detail, agents a_i can be programmed using the JADE-ANDROID add-on[8], a JADE module that allows combining the expressiveness of JADE agents communication with the power of ANDROID platform. As already stated, another relevant feature of JADE is the integration with JESS, the rule engine that are the basis on which the expert system is built on according to the KA conceptual model. In fact, KA-Agent has the capability to create the expert system on the basis of domain expert information and it can directly execute it according to data sent it by client. In this way, it could be explored the possibility to import JESS under ANDROID, moving its execution from the current KA–Agent on the server side to the mobile environment.

Concerning the conceptual model of KA–Agent, it should be possible to create a multi-language environment, expanding it to other rule–based languages, such as Drools[9].

Furthermore, the definition of rules could be improved to provide the user with the possibility to define new kinds of constructs, like templates (and the relative slots), functions, shadow facts and so on.

References

[1] Y. Shoham, "Agent-oriented programming," *Artificial intelligence*, vol. 60, no. 1, pp. 51–92, 1993.

[2] F. Bellifemine, A. Poggi, and G. Rimassa, "Jade–a fipa-compliant agent framework," in *Proceedings of PAAM*, vol. 99, no. 97-108. London, 1999, p. 33.

[3] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Wiley. com, 2007, vol. 8.

[4] A. Ricci, M. Piunti, M. Viroli, and A. Omicini, "Environment programming in cartago," in *Multi-Agent Programming:*. Springer, 2009, pp. 259–288.

[5] M. Ughetti, T. Trucco, and D. Gotta, "Development of agent-based, peer-to-peer mobile applications on android with jade," in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*. IEEE, 2008, pp. 287–294.

[6] A. Santi, M. Guidi, and A. Ricci, "Jaca-android: An agent-based platform for building smart mobile applications," in *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, ser. Lecture Notes in Computer Science, M. Dastani, A. Fallah Seghrouchni, J. Hbner, and J. Leite, Eds. Springer Berlin Heidelberg, JACA2011, vol. 6822, pp. 95–114. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22723-3_6

[7] S. Bandini and F. Sartori, "From handicraft prototypes to limited serial productions: Exploiting knowledge artifacts to support the industrial design of high quality products," *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing)*, vol. 24, no. 1, p. 17, 2010.

[8] C. Brewster and K. O'Hara, "Knowledge representation with ontologies: the present and future," *Intelligent Systems, IEEE*, vol. 19, no. 1, pp. 72–81, 2004.

[9] J. A. Rosen and W. L. Smith, "Influence net modeling with causal strengths: an evolutionary approach," in *Proceedings of the Command and Control Research and Technology Symposium*, 1996, pp. 25–28.

[10] B. Chandrasekaran, T. R. Johnson, and J. W. Smith, "Task-structure analysis for knowledge modeling," *Communications of the ACM*, vol. 35, no. 9, pp. 124–137, 1992.

[11] G. Salazar-Torres, E. Colombo, F. S. C. da Silva, C. A. Noriega, and S. Bandini, "Design issues for knowledge artifacts," *Knowl.-Based Syst.*, vol. 21, no. 8, pp. 856–867, 2008.

[12] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, and W. Van de Velde, "Commonkads: A comprehensive methodology for kbs development," *IEEE expert*, vol. 9, no. 6, pp. 28–37, 1994.

[13] J. Angele, D. Fensel, D. Landes, and R. Studer, "Developing knowledge-based systems with mike," *Automated Software Engineering*, vol. 5, no. 4, pp. 389–418, 1998.

[8]Released with LGPLv2 Licence and available at http://jade.tilab.com/

[9]http://www.jboss.org/drools/

Mining the user profile from a smartphone: a multimodal agent framework

Giuseppe Loseto, Michele Ruta, Floriano Scioscia, Eugenio Di Sciascio, Marina Mongiello
DEI - Politecnico di Bari
via E. Orabona 4, I-70125, Bari, Italy
loseto@deemail.poliba.it, {m.ruta, f.scioscia, disciascio, mongiello}@poliba.it

***Abstract*—Nowadays smartphones play a significant role in gathering relevant data about their owners. Micro-devices embedded in Personal Digital Assistants (PDAs) perform a continuous sensing, the phone call lists, PIM (Personal Information Manager), text messages and so on allow to collect and mine data enough for a high-level description of daily activities of a user. This paper proposes an agent able to perform an automated profile annotation by adopting Semantic Web languages. As a proof of concept, the devised agent has been tested in an Ambient Intelligence (AmI) scenario, *i.e.*, a domotic environment where it interacts with its home counterpart to trigger services best matching the user needs. A toy example is presented as case study aiming to better clarify the proposal while an early experimental evaluation is reported to assess its effectiveness.**

Keywords*—*Ambient Intelligence; Agent-based Data Mining; Semantic Web of Things; Home and Building Automation.

I. Introduction

Mobile phones are both pervasive and personal –following the user and having clues about everyday situations– resulting extremely useful to infer a context. Embedded micro-devices (accelerometer, digital compass, gyroscope, GPS, microphone and camera) can be used to extract significant information about the user: GPS location traces, call and SMS lists, PIM (Personal Information Management) records including contacts and calendar, battery charging habits. By leveraging the smartphone processing capabilities, ever-expanding ways to investigate behavioral, spatial and temporal dimensions of the everyday life can be provided. The personal nature of mobile phones suggest they are well suited for pervasive computing, but data they are able to collect and process could be profitably used for a large set of context-aware applications, like the *Ambient Intelligence* (AmI) [1] ones.

This paper presents a smart profiling agent[1] which borrows languages and technologies from the Semantic Web experience to funnel inarticulate raw individual information toward a semantically rich glossary. A crawler agent runs on the user smartphone and performs a multimodal (*i.e.*, involving several heterogeneous data sources) and continuous sensing [2] collecting and processing information without human intervention. The multimodality requires specialized analyses for each kind of collected data. The agent mines the user habits automatically and annotates them in a logic-based formalism to build a daily profile to be further exploited in context-aware knowledge-based applications. The main motivation for adopting an agent-based approach is that the mobile profiler must modulate proactively the amount and complexity of data capture and processing, in order to use energy efficiently. Smart Home and Building Automation (HBA) [3] was selected as proof scenario: the profiling agent sends the inferred preferences to its HBA counterpart so that a logic-based matchmaking session could finalize the adaptation of the environment to user needs.

The remainder of the paper is organized as in what follows. Section II contextualizes the overall multi-agent HBA system motivating the proposed approach before presenting both architecture and algorithms of the profiler agent in Section III. The toy example in Section IV acts as a case study while an early experimental evaluation is reported in Section V. Finally, most relevant related work is discussed in Section VI and concluding remarks and future research are in Section VII.

II. Scenario: semantic-based home automation

The user agent proposed in this paper is intended as a part of a more complex HBA Multi Agent System (MAS) [4] leveraging the semantic-based evolution of the KNX domotic protocol in [5]. It introduced a semantic micro-layer on the top of the stack enabling novel services and functions while keeping a full backward-compatibility with current domestic devices and HBA appliances. The above enhancements allowed to fully describe device features by means of annotations expressed in logic-based languages such as RDF[2] and OWL[3]. The knowledge domain of building automation was conceptualized in a shared ontological vocabulary enabling a rich characterization of home resources and services. The MAS was implemented in Java on a testbed composed of off-the-shelf KNX domotic equipment[4].

The adopted multi-agent system comprised a home mediator agent as well as user and device agents. Each agent adopts the custom service-oriented model sketched in [4, Fig. 4]. Basically, the agent monitors its internal state and inputs; when a significant change occurs, it communicates with the other agents in order to discover suitable services that maximize its utility. The number of both resources/services and agents varied unpredictably (as new users or devices joined or disconnected the system at any time) without redefining the communication paradigm for that.

[1]Project home page: http/sisinflab.poliba.it/swottools/mobile-user-profiler/

[2]RDF (Resource Description Framework) Primer, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/rdf-primer/

[3]OWL 2 Web Ontology Language, W3C Recommendation, 11 December 2012, http://www.w3.org/TR/owl2-overview/

[4]See the related project home page http://sisinflab.poliba.it/swottools/smartbuildingautomation/ for more details.

– The *Mediator Agent* coordinates the explicit characterizations of available services, described w.r.t. a reference ontology modeling the conceptual knowledge for the building automation problem domain. Furthermore, it acts as a broker in order to discover the (set of) elementary services that cover (part of) the request coming from user or device agents.
– The *Device Agents* are thought to run on advanced devices, *i.e.*, home appliances with some computational capabilities and memory availability. Each one can expose one or more semantic descriptions, *i.e.*, functional profiles to be discovered by other agents, or alternatively each of them could issue semantic-based requests to the mediator agent when the device status changes and then require a home reconfiguration.
– *KNX Device Interface Agents* support semantic-based enhancements in case of legacy or elementary appliances, *e.g.*, switches, lamps, and so on. In such cases, there is only a static interaction between agent and device.
– Finally the *User Agents*, running on mobile clients, send requests toward the home environment, in order to satisfy user needs and preferences. W.r.t. the version in [4], an approach for the automated mining of a user profile in charge to that kind of agent is proposed as main contribution of this paper.

III. Framework and Approach

Figure 1 sketches the general architecture of the profiling agent. Raw data are extracted from smartphone embedded micro-devices, communication tools and PIM. The data mining life cycle consists of the following subsequent stages: (a) gathering; (b) feature extraction; (c) classification and interpretation; (d) semantic annotation. High-level information about user activities, whereabouts, mental and physical status is inferred and annotated w.r.t. an extension of the HBA ontology in [5]. The mined profile should be finally used to trigger the activation or deactivation of the most appropriate home services. A modular architecture allows to process the various data sources with specialized algorithms. In particular, as shown by icons in Figure 1, three modules fully characterize the agent at the moment: (i) Points of Interest Recognition; (ii) Transportation Mode Recognition; (iii) User Activity Recognition.

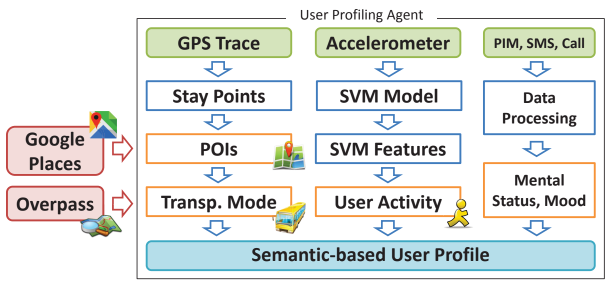
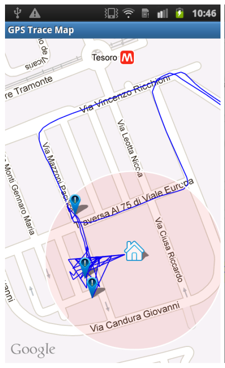
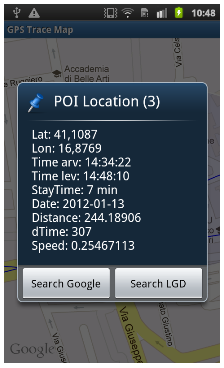


Fig. 1. Reference architecture of the user profiling agent

1. Points of Interest Recognition. A mining algorithm analyzes the smartphone GPS data in order to:
a. identify Stay Points (SPs) through a slightly refined version of the algorithm in [6];
b. for each SP, retrieve the nearest Point Of Interest (POI) via reverse geocoding queries to *Google Places*[5] Web service;
c. associate a "place category" to each POI, so as to further infer the kind of user activity;
d. enrich the daily user profile conjoining all detected activities, described w.r.t. a proper HBA ontology.
A SP represents a narrow geographic region where a user stands for a while. In particular, given two subsequent detected GPS locations P_1 and P_2, a SP satisfies both the following constraints: (i) maximum distance $d(P_1, P_2) < D_{max}$; (ii) minimum time difference $|T_1 - T_2| > T_{min}$, where the thresholds were set to $D_{max} = 200m, T_{min} = 350s$. An empirical evaluation was executed to assign the thresholds values granting the highest precision of the SP recognition algorithm.



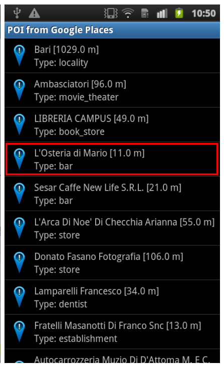
(a) Home POI



POI Location (3)

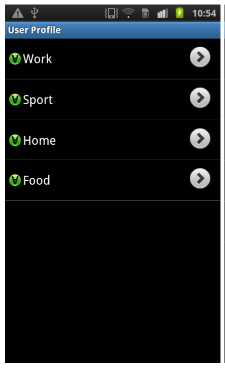
Lat: 41,1087
Lon: 16,8769
Time arv: 14:34:22
Time lev: 14:48:10
StayTime: 7 min
Date: 2012-01-13
Distance: 244.18906
dTime: 307
Speed: 0.25467113

(b) POI Info



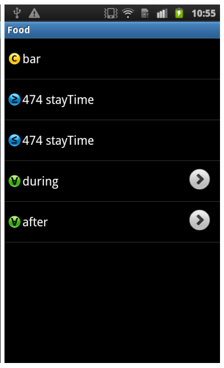
Bari [1029.0 m]
Type: locality
Ambasciatori [96.0 m]
Type: movie_theater
LIBRERIA CAMPUS [49.0 m]
Type: book_store
L'Osteria di Mario [11.0 m]
Type: bar
Sesar Caffe New Life S.R.L. [21.0 m]
Type: bar
L'Arca Di Noe' Di Checchia Arianna [55.0 m]
Type: store
Donato Fasano Fotografia [106.0 m]
Type: store
Lamparelli Francesco [34.0 m]
Type: dentist
Fratelli Masanotti Di Franco Snc [13.0 m]
Type: establishment
Autocarrozzeria Muzio Di D'Attoma M. E C.

(c) Extracted Places



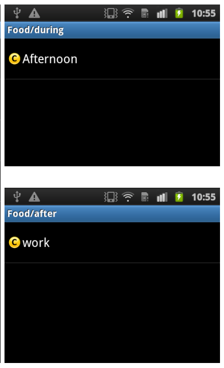
Work
Sport
Home
Food

(d) Profile mining



bar
474 stayTime
474 stayTime
during
after

(e) *Food* place detail



Afternoon

work

(f) Daily stay period and location visited before

Fig. 2. Screenshots of the GPS profiler

Figure 2 shows the GUI of the profiler prototype on the GPS-side. The daily GPS trace is drawn on Google Maps together with detected SPs, depicted as markers on the map in Figure 2(a). The *Home* and *Workplace* POIs are set by the user in a preliminary configuration step. As said, the SP classification leverages a Web-based reverse geocoding service: after comparing Google Places and *LinkedGeoData* (LGD) [7] (see Section V for further details) the first one service has been chosen at the moment, since it provides more available POIs even if LGD often seems to be more accurate. In the example reported in Figure 2(c), the agent selected a SP near to the *Politecnico di Bari* and all the nearby POIs were retrieved by means of the Google Places API. The main category of the nearest POI is used as label of the retrieved location. Starting from the Google Places classification[6], the

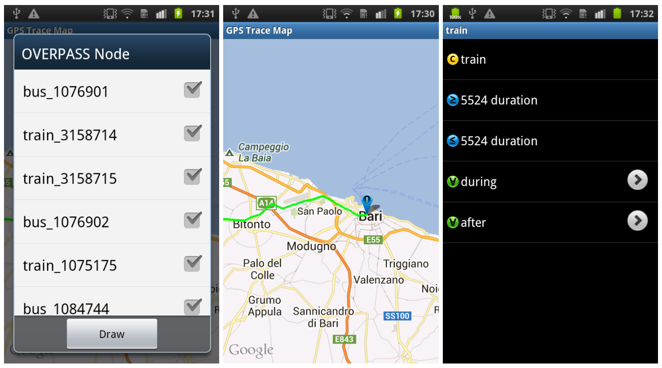
[5]http://developers.google.com/places/

[6]http://developers.google.com/places/documentation/supported_types/

reference ontology for domotics in [5] has been extended to include a places taxonomy. Finally, as reported by the Figure 2(d), a profile is generated through the conjunction of location information. As shown in Figure 2(e), each SP description contains an ontology class related to the specific location the user visited, the overall time spent there (in seconds), the daily period and the place visited before, if present (Figure 2(f)).

2. Transportation Mode Recognition. GPS data are exploited also to detect the transportation mode adopted by the user when moving during a day. Four transportation modes are supported: bus, train, car or walking. A pre-processing splits the whole daily GPS trace $\mathcal{P} = \{T_1, \ldots, T_n\}$ in trajectories $\mathcal{T}_i$. In turn, each trajectory $\mathcal{T}_i = \mathcal{Q}\{POI_i, POI_{(i+1)}\}$ consists of a set of GPS points $\mathcal{Q}$ included between two subsequent POIs. Starting from the trajectories set, the transportation mode detection is based on two reference parameters: (i) the *walking speed* threshold (WS_{th}), set to an average value of 2 m/s (*i.e.*, 7.2 km/h); (ii) the *minimum correspondence ratio* (CR_{min}) between user trajectories and bus/train routes, set to 0.8 (*i.e.*, at least a 80% correspondence is required). Also in this case, an experimental evaluation was performed to select the most suitable threshold values. The algorithm for detection progresses along the following stages:
a. For each trajectory T_i, the average user speed is evaluated. If it is lower than WS_{th} then walking mode is detected.
b. Otherwise, the algorithm queries OpenStreetMap[7] (OSM) via the Overpass API[8] to retrieve all available bus and train routes ($Rs = R_{bus} \cup R_{train}$) in a *bounding box* covering the geographical coordinates of the GPS points in T_i. Figure 3(a) shows an example for that.
c. A comparison between the GPS points of the user trajectory and the retrieved routes is performed. In case of a correspondence ratio greater than CR_{min} with a bus or train path, the trajectory T_i is associated to a bus or train mode, respectively (Figure 3(b)).
d. Finally, if the detected mean is neither walking nor train nor bus, then the car mode is selected.
Each transportation mode is associated to a semantic-based annotation fragment which includes a given class of the ontology, further extended to include also concepts and properties about user movements. Moreover, the description will include the overall time –in seconds– the user spent during the day for moving, the daily period and possible means of transport used before. Figure 3(c) shows the details about the user profile section related to a transfer by train.

3. User Activity Recognition. Beyond the above components, the profiling agent is completed by a module to detect some user activities. In particular, at the moment the following elementary actions can be discovered: sitting, standing, walking, walking upstairs and dowstairs. Starting from data acquired from the smartphone accelerometer and gyroscope, a supervised Machine Learning (ML) approach is adopted, exploiting the Support Vector Machines (SVM) classifier in [8]. W.r.t. the original approach, the classifier was simplified to improve its efficiency on PDAs and to reduce the training time. The early 568 features used on the dataset[9] associated to [8] as input


GPS Trace Map
OVERPASS Node
bus_1076901
train_3158714
train_3158715
bus_1076902
train_1075175
bus_1084744
Draw

GPS Trace Map

train
5524 duration
5524 duration
during
after

(a) Overpass routes (b) Train Mode (c) Train Mode details

Fig. 3. Screenshots of the Transportation Mode profiler

#	Feature description
1	tBodyAcc correlation(X,Y)
2	tGravityAcc mean(X)
3	tGravityAcc mean(Y)
4	tGravityAcc max(Z)
5	tGravityAcc min(X)
6	tGravityAcc energy(X)
7	tBodyGyro iqr(Z)
8	tBodyGyroJerk entropy(X)
9	tBodyGyroJerk entropy(Z)
10	tBodyAccJerkMag iqr(X,Y,Z)
11	tBodyGyroJerkMag energy(X,Y,Z)
12	fBodyGyro max(Y)
13	fBodyGyro max(Z)
14	fBodyGyro skewness(Z)
15	fBodyAccMag std(X,Y,Z)
16	fBodyAccMag energy(X,Y,Z)

t=time domain, f=frequency domain, Jerk=derived in time, Mag=Euclidean norm, iqr=Interquartile range

TABLE I. Features subset for the SVM classifier

for the classifier were reduced to 16 (see Table I) by applying the Recursive Feature Elimination (RFE) algorithm proposed in [9].

A training set composed by sensor raw data has been used to let the classifier learn directly on the mobile device. The smartphone used for the experimental evaluation is equipped with an accelerometer and a gyroscope measuring both the 3-axial linear acceleration and the angular velocity (*tAcc-XYZ* and *tGyro-XYZ*, respectively) at a fixed sampling rate of 25 ms, which is adequate to identify a human body motion. The collected data are subsequently processed through two first-order low-pass filters. The first one is used to reduce noise, while the second filter splits the acceleration signal into body and gravity components (*tBody* and *tGravity*). The classifier has been implemented using *Weka-for-Android*[10], an Android port of Weka [10]. The training set has been built fastening the smartphone in vertical position as reference; after the SVM training, the recognition process starts. Data are sampled in fixed-width sliding windows of 2.5 s (*i.e.*, 100 samples) with 50% overlap, and processed as described above. From each window, a vector with the 16 features in Table I is obtained by computing the extracted accelerometer and gyroscope data in the time and frequency domain. Finally, an energy saving strategy is implemented to avoid unnecessary data capture: after each activity recognition AR_i, a pause WP_i is waited

[7]http://www.openstreetmap.org/
[8]http://wiki.openstreetmap.org/wiki/Overpass_API
[9]http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[10]https://github.com/rjmarsan/Weka-for-Android

for. WP_i is defined as:

$$WP_i = \begin{cases} 0sec & \text{if } AR_i \neq AR_{i-1} \\ 2.5sec & \text{if } AR_i = AR_{i-1} \\ (WP_{i-1} * 2)sec & \text{if } AR_i = AR_{i-1} = AR_{i-2} \end{cases}$$

In this way, if the classifier consecutively detects two similar activities, then the data sampling is stopped for 2.5 seconds. This value is doubled in case of additional similar recognitions, up to a maximum value of $WP_i = 80s$. Otherwise, the waiting period is reset to zero when a different action is detected. The rationale is that users usually perform similar activities in a short period –consider for example the case of sitting and walking– so a continuous data gathering could be often avoided.

The vector containing the extracted features is then used as input of the trained SVM model. Finally, the user profile is enriched with the annotations related to the detected activities. For each of them it will be also considered the overall stay time and the daily period.

IV. Case study

In order to clarify the rationale behind the proposed approach and to let emerge the goal of the profiling agent, the following daily scenario is considered as example. *The user leaves home early in the morning to go to work. He remains at office until lunch, then reaches a bar for a fast meal. Afterward, he comes back to work, then goes to the gym in the evening and finally returns home late at night.* The profiling agent extracts the daily location sequence reported in Table II. Particularly, Home and Office POIs are mapped to the user profile directly as *Home* and *Work* activities; Bar is identified as a *Food* place; Gym is associated to the *Sport* place category. The agent also recognizes the adopted means of transport and the duration of each trajectory.

Route	Type	Duration (min)
Home → Office	car	30
Office → Bar	walk	4
Bar → Office	walk	5
Office → Gym	car	11
Gym → Home	car	21

TABLE II. Daily user locations and routes

Along the day, the agent also detects the activities of the user: he was seated for about 6 hours (*e.g.*, at work, within the car, during lunch), walked for 35 minutes (*e.g.*, to reach the bar or for short strolls) and was standing for 15 minutes. As a result of the mining and annotation processes, the following profile is extracted (expressed in Description Logic [11] notation w.r.t. the reference ontology)[11]:

User_Daily_Profile $\equiv \forall\ wasAtHome.HomeActivity \sqcap \forall\ wasAtWork.WorkActivity \sqcap \forall\ wasInFoodPlace.FoodActivity \sqcap \forall\ wasInSportPlace.SportActivity \sqcap \forall\ movedByCar.CarMode \sqcap \forall\ movedByWalk.WalkMode \sqcap \forall\ wasSitting.SittingActivity \sqcap \forall\ wasWalking.WalkingActivity \sqcap \forall\ wasStanding.StandingActivity$

HomeActivity $\equiv Home \sqcap \forall\ during.(Morning \sqcap Night) \sqcap \forall\ after.Gym \sqcap =_{1945}\ stayTime$

WorkActivity $\equiv Work \sqcap \forall\ during.(Morning \sqcap Afternoon) \sqcap \forall\ after.(Home \sqcap Bar) \sqcap =_{32470}\ stayTime$

[11]Due to space constraints, some sections have been voluntarily omitted.

FoodActivity $\equiv Bar \sqcap \forall\ during.Afternoon \sqcap \forall\ after.Work \sqcap =_{474}\ stayTime$

SportActivity $\equiv Gym \sqcap \forall\ during.Evening \sqcap \forall\ after.Work \sqcap =_{5362}\ stayTime$

WalkMode $\equiv Walk \sqcap =_{2115}\ moveTime \sqcap \forall\ during.Afternoon \sqcap \forall\ after.Car$

SittingActivity $\equiv Sitting \sqcap =_{21436}\ moveTime \sqcap \forall\ during.(Morning \sqcap Afternoon \sqcap Evening)$

The above generated profile will be adopted by the user agent to negotiate with the mediator agent at home the environmental situation best fitting needs and mood of the inhabitant via a semantic-based matchmaking. The elementary services and appliances covering the mined user profile as much as possible are automatically activated (or in case deactivated) to increase the overall MAS utility. As an example of this phase, let us consider the following available home services/resources:

CookingService $\equiv Service \sqcap \forall\ wasInSportPlace.(\ >=_{1800}\ stayTime) \sqcap \forall\ wasAtHome.(\ \forall\ after.(Sport \sqcap \neg Food)) \sqcap \forall\ suggestedForFeeling.Hungry$

SoftLightLevel $\equiv LightLevelRegulation \sqcap \forall\ wasAtWork.(\ >=_{10800}\ stayTime) \sqcap \forall\ wasAtHome.(\ \forall\ after.\ \neg Relax) \sqcap \forall\ suggestedForStamina.MentallyTired \sqcap \forall\ suggestedForDisease.Headache$

PlayMusic $\equiv Service \sqcap \forall\ wasAtHome.(\ \forall\ after.(\ \neg Work \sqcap Relax) \sqcap \forall\ during.\ \neg Night) \sqcap \forall\ suggestedForStamina.Rested \sqcap \forall\ suggestedForDisease.\ \neg Headache$

It should be noticed that service annotations are described in terms of both user features (such as a physical status, mood and health) and daily events which cause the activation. In this way, a service/resource selection can be performed through the matchmaking against the user profile. For example, a cooking service is activated not only if the user explicitly declares he is hungry, but also if the user agent detects he comes back home after a sport activity, performed for more than 30 minutes (expressed in seconds), without eating anything before. In a similar way, a soft lighting setting is selected to improve the comfort at home in case the user is mentally tired and he spent more than 3 hours at work not followed by a restful activity. The extracted user profile can also lead to a deactivation of previously enabled services. For example, the music service is normally activated to welcome the owner at home, but it is unsuitable if the user comes back during the night and in that case it must be turned off.

The above case study is purposely simplified in order to make the presentation of the proposed approach clear and short. In real scenarios, more articulated user profiles and service descriptions can be used.

V. Experiments

An overall evaluation of the proposed approach has been carried out following a reference user for a period of 14 months. Results reported here refer to the first 60 days of observation. In particular, only the days –24 in the evaluated dataset excerpt– with at least one Stay Point different from Home or Workplace have been selected for further investigation. The profiling agent has been tested on a smartphone equipped with an ARM Cortex A8 CPU at 1 GHz, 512 MB RAM, a 8 GB internal storage memory, and Android 2.3.3

as operating system. Done experiments basically aimed to measure: (i) the amount of data retrieved from services on the Web; (ii) the turnaround time (for which each test was repeated four times taking the average of the last three runs); (iii) the memory usage (for which the final result was the average of three runs). This experimental analysis only focuses on the user profiling aspects: [4] reports on evaluation of the remaining elements of the reference HBA MAS.

Figure 4 shows the total number of stay points detected with the mining algorithm compared with the overall GPS coordinates composing a daily trace. It can be noticed that the user agent collects 53 GPS points per day on average, detecting about 3 relevant SPs.

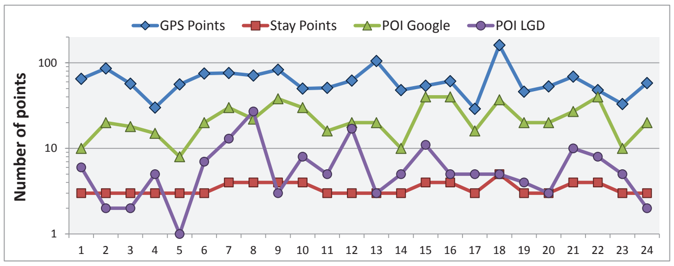


Fig. 4. GPS points, detected SPs and retrieved POIs

Starting from detected SPs, the results of Google Places and LGD services have been compared in terms of number of retrieved POIs in the neighborhood of each SP. As shown in Figure 4, Google Places usually returns 16 POIs w.r.t. 5 POIs on average retrieved by LGD, so an accurate identification of the locations the user visited is more likely. Nevertheless, as reported in Figure 5, in some cases the LGD replies are longer even though it returns fewer POIs. This is due to the LGD response format including, for each point, information annotated according to *Linked Data* principles [12]: Google Places uses 830 B per POI on average, whereas LGD uses 1.56 kB.

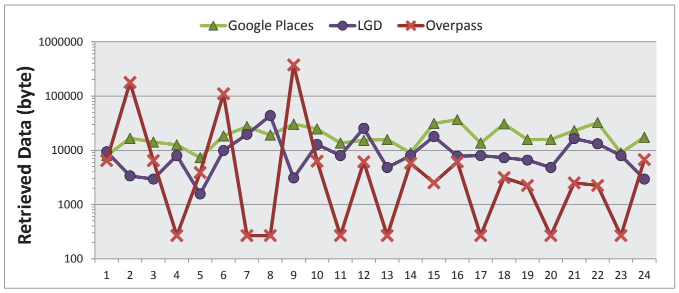


Fig. 5. Retrieved Data

The time required by the main processing steps for POIs recognition (GPS traces parsing; SPs detection; Google Places/LGD services querying; profile enrichment), transportation mode detection (Overpass service querying; traces comparison; profile enrichment) and activity recognition are reported in Figure 6. Google Places is slightly slower than LGD, but this is due to the greater amount of retrieved POIs. Considering Google Places as reference service, the agent spends about 1.2 s to retrieve the POIs from a detected SP.

Activity	**A**	**B**	**C**	**D**	**E**	**Recall %**
A Sitting	**340**	0	0	0	0	100
B Standing	0	**98**	0	1	0	98.9
C Walking	1	0	**70**	0	3	94.6
D Walking Upstairs	0	0	2	**125**	5	94.7
E Walking Downstairs	0	0	0	4	**130**	97.0
Precision %	99.7	100	97.2	96.2	94.2	**98.0**

TABLE III. CONFUSION MATRIX

In particular, the last step took about 1.15 s (49% of total time) to parse the ontology and create the semantic-based annotation. The remaining steps require only the 3% of the overall turnaround time, as these procedures use elementary data structures stored in the device main memory. For the transportation mode detection, only 1.7 s were spent to query the Overpass service, while traces comparison is one of the slower operations, needing 3.4 s. The activity recognition process has a very short turnaround time. After a preliminary task (required to train the SVM classifier) taking about 5.6 s and performed when the profiling agent starts, this module needs only 45 ms to extract the 16 reference features for each windows and 6 ms to detect the user activity. Finally, a daily profile was completely composed in about 1.2 seconds.

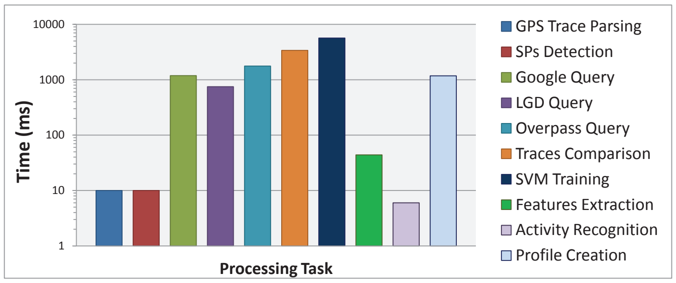


Fig. 6. Processing Time

A further evaluation of the activity recognition module required to measure precision and recall of the classifier. 100 datasets of activities containing a similar number of samples per class have been used. The confusion matrix shown in Table III reports on the weighted precision of the classifier and on single precision and recall values for each activity. It is referred to a single specific dataset with 779 sample vectors. However all confusion matrices for different tests showed similar outputs, varying slightly in the classification results. It is possible to notice that the classifier precision and recall are very high despite the usage of a small set of features.

RAM usage trend was also evaluated and results are shown in Figure 7, where memory peaks are reported. The profiler agent needs very low memory, only 4.2 MB on average, a satisfactory value for current mobile devices.

VI. RELATED WORK

The recent popularization of smartphones equipped with a wide range of embedded sensors and adequate processing capabilities has attracted increasing research efforts toward mobile sensing. Lane *et al.* [2] proposed a survey on existing algorithms, applications, and systems. In addition, many pervasive frameworks were defined to collect and capture the user's

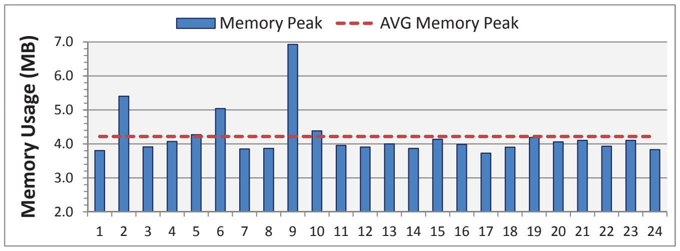


Fig. 7. Main memory usage trend

context via cellphones in latest years: remarkable works are *ContextPhone* [13], *UbiqLog* [14] and *LifeMap* [15]. The agent proposed here aims to improve upon these works by leveraging the multimodality aspect: the implemented prototype retrieve information from a data source richer than the above systems, even though further mining modules have been planned but not integrated yet. A comparison should be carried out also with respect to commercial location and context-aware mobile software: trekking and fitness applications like *Google MyTracks*[12] and *Endomondo Sportstracker*[13]; personalized assistants like *Google Now*[14] and *Xme*[15]. Nevertheless, these tools either require explicit user interaction or define context just by means of GPS location and time of day, hence they are quite far off the agent proposed here which uses more parameters and automatically recognizes a larger variety of contexts.

The activity recognition from accelerometer by means of machine learning is a frequent sensing application. Among other proposal, noteworthy are [16], [8] where smartphone accelerometer data are used to classify six common activities. With reference to context extraction via GPS data analysis, there are many approaches in literature. For example Zheng *et al.* [17] model multiple individuals GPS trajectories with a tree-based hierarchical graph to mine location history and travel sequences in a given geospatial region. In [6] mobile phones are used as sensors to collect location information. Places are first grouped using a time-based clustering technique to discover *stay points*; then the stay points are clustered in *stay regions* through a grid-based algorithm. In [18] a large-scale dataset is collected from 114 users over 18 months.

In the above cited works, however, the knowledge gap between acquired data and the understanding of human behavior is still huge. Stay points and movement patterns require to be interpreted to extract a user profile, implicitly providing knowledge about the user habits. Noteworthy attempts to enrich movement trajectories with semantics are in [19] and [20]. An ontology-based approach for a semantic modeling of trajectories is also proposed in [21]. Trajectories are seen as composed by three main elements: stops, moves and begin-ends. Each part is described through an annotation referred to a domain ontology and time information are also exploited to annotate activities to enable rule-based queries and to help users validate and discover moving objects.

Although previous solutions add a machine-understandable meaning to data collected by smartphones, a subsequent exploitation in an articulated AmI framework is still missing. Usually, collected data are only used to indicate detected user conditions or activities through messages or alerts displayed on the mobile phone. On the contrary, in the approach proposed here, the ontology-based characterization of user activities is used as an input for a context-aware HBA MAS [4], enabling a direct environment adaptation and a negotiation between user and home agents. This feature is not possible for any other current user profiler.

[12]http://www.google.com/mobile/mytracks/
[13]http://www.endomondo.com
[14]http://www.google.com/landing/now/
[15]http://xndme.com/

VII. Conclusion and Future Work

The paper presented a lightweight agent able to mine data collected by embedded micro-devices, logs and applications of a smartphone to build a semantic-based daily profile of its user. According to the AmI paradigm, such a description can be exploited to transparently adapt the environment to user preferences, implicitly inferred. In the matter in question, the agent interacts in a multi-agent framework for Home and Building Automation, grounded on knowledge representation theory and reasoning technologies. It has been designed and then implemented as an Android application and experiments in a concrete case study proved its feasibility and effectiveness.

Future work will include a more extensive experimental campaign involving several different users to be profiled and new performance indicators. Particularly, both battery drain and storage peaks will be taken into account to assess the feasibility of a continuous data collection and mining and to compare the provided framework with existing approaches. Also the exploitation of an agent-based framework w.r.t. to classical approaches will be posed under investigation to verify if it results in a more accurate profiling action. Finally, future research will be also devoted to the integration of the current multimodal information. A fusion of information coming from data sources which now are distinct and independent will be pursued in order to reach a more accurate and precise user characterization.

Acknowledgment

The authors acknowledge partial support of Italian PON project *Res Novae* and EU PO Apulia region FESR project *UbiCare*.

References

[1] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277 – 298, 2009.

[2] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, Sep. 2010.

[3] G. Loseto, F. Scioscia, M. Ruta, and E. Di Sciascio, "Semantic-based Smart Homes: a Multi-Agent Approach," in *13th Workshop on Objects and Agents (WOA 2012)*, ser. CEUR Workshop Proceedings, F. De Paoli and G. Vizzari, Eds., vol. 892, Sep 2012, pp. 49–55.

[4] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio, "Semantic-based resource discovery and orchestration in home and building automation: a multi-agent approach," *IEEE Transactions on Industrial Informatics*, 2013, to appear.

[5] M. Ruta, F. Scioscia, E. Di Sciascio, and G. Loseto, "Semantic-based Enhancement of ISO/IEC 14543-3 EIB/KNX Standard for Building Automation," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 731–739, 2011.

[6] R. Montoliu, J. Blom, and D. Gatica-Perez, “Discovering places of interest in everyday life from smartphone data,” *Multimedia Tools and Applications*, pp. 1–29, 2012.

[7] C. Stadler, J. Lehmann, K. Höffner, and S. Auer, “LinkedGeoData: A Core for a Web of Spatial Open Data,” *Semantic Web Journal*, vol. 3, no. 4, pp. 333–354, 2012.

[8] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz, “Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine.” in *Workshop of Ambient Assisted Living (IWAAL 2012)*, 2012.

[9] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene selection for cancer classification using support vector machines,” *Machine Learning*, vol. 46, pp. 389–422, 2002.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[11] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook*. Cambridge University Press, 2002.

[12] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data - The Story So Far,” *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[13] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, “Contextphone: A prototyping platform for context-aware mobile applications,” *IEEE Pervasive Computing*, vol. 4, no. 2, pp. 51–59, Apr. 2005.

[14] R. Rawassizadeh, M. Tomitsch, K. Wac, and A. Tjoa, “Ubiqlog: a generic mobile phone-based life-log framework,” *Personal and Ubiquitous Computing*, pp. 1–17, 2012.

[15] J. Chon and H. Cha, “LifeMap: A Smartphone-Based Context Provider for Location-Based Services,” *IEEE Pervasive Computing*, vol. 10, no. 2, pp. 58–67, Apr. 2011.

[16] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[17] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, “Mining Interesting Locations and Travel Sequences From GPS Trajectories,” in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW ’09. New York, NY, USA: ACM, 2009, pp. 791–800.

[18] T. M. T. Do and D. Gatica-Perez, “The Places of Our Lives: Visiting Patterns and Automatic Labeling from Longitudinal Smartphone Data,” *IEEE Transactions on Mobile Computing*, 2013, PrePrints.

[19] C. Renso, M. Baglioni, J. Macedo, R. Trasarti, and M. Wachowicz, “How you move reveals who you are: understanding human behavior by analyzing trajectory data,” *Knowledge and Information Systems*, pp. 1–32, 2012.

[20] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan, “Semantic Trajectories Modeling and Analysis,” *ACM Computing Surveys*, vol. 45, no. 4, 2013.

[21] R. Wannous, J. Malki, A. Bouju, and C. Vincent, “Time Integration in Semantic Trajectories Using an Ontological Modelling Approach,” in *New Trends in Databases and Information Systems*, ser. Advances in Intelligent Systems and Computing, M. Pechenizkiy and M. Wojciechowski, Eds. Springer Berlin Heidelberg, 2013, vol. 185, pp. 187–198.

Evaluating Negotiation Cost for QoS-aware Service Composition

Claudia Di Napoli
Istituto di Cibernetica "E. Caianiello"
C.N.R., Via Campi Flegrei 34, 80078
Pozzuoli, Napoli, Italy
c.dinapoli@cib.na.cnr.it

Dario Di Nocera*
Dipartimento di Matematica e Applicazioni,
Università degli Studi di Napoli
"Federico II", via Cintia MSA,
80126 Napoli, Italy
dario.dinocera@unina.it

Silvia Rossi
Dipartimento di Ingegneria Elettrica
e Tecnologie dell'Informazione,
Università degli Studi di Napoli
"Federico II", via Claudio 21,
80125 Napoli, Italy
silvia.rossi@unina.it

Abstract—**The value of commercial Service-Based Applications (SBAs) will depend not only on their functionality, but also on the value of their non-functional properties, known as QoS attributes, that are not tied to a specific functionality, but rather to its delivery features. QoS values may vary according to the provision strategies of providers as well as users' requirements expressed as global constraints on the SBA QoS. Automatic negotiation is a viable approach to drive QoS-aware selection of services for SBAs, but its adoption may result computationally expensive due to the communication overhead among the involved negotiators, so limiting its application to real service-based scenarios. In this paper, an empirical evaluation of the impact of negotiation communication costs occurred when composing services to deliver a QoS-aware SBA is carried out, in order to estimate the advantages and disadvantages of negotiation in a market of services, and to identify negotiation parameters settings for which communication costs can be compensated by an increased probability for the negotiation to succeed.**

I. Introduction

The increased popularity of the Service-Oriented-Computing (SOC) paradigm [1] is enhancing the development of Service Based Applications (SBAs) that are distributed applications obtained by combining existing services in a loosely coupled manner that collectively fulfill a requested task. Services are independent and autonomous entities provided by different service providers to be consumed by users requesting a given application. Users do not need to be aware of the actual composition as long as the functional and non-functional requirements are satisfied [2]. The consumption of these services is commonly governed by an agreement among providers and consumers, known as Service Level Agreement (SLA), which regulates terms and conditions of service provision [3]. Agreements on service provisioning may include not only the provider's commitment to execute a given task (coming from functional requirements), but they also include terms about performance levels (or quality levels) of services [4] (coming from non-functional requirements). Service non-functional requirements refer to service attributes known as Quality of Service (QoS) attributes that play an important role in service selection.

In a previous paper [5] we proposed a market-based negotiation mechanism among service providers and a service consumer to select the suitable services to compose QoS-aware SBAs. The approach allows for the selection of services according to the values of their quality attributes that, once aggregated, have to meet end-to-end user QoS constraints/preferences. The negotiation-based mechanism allows to take into account the variability of service QoS attribute values typical of the future market of services since service providers may change these values during the negotiation according to their own provision strategies, and market trends.

* Ph.D. scholarship funded by Media Motive S.r.l, POR Campania FSE 2007-2013.

The negotiation protocol is designed as a *one-to-many-to-many* iterative protocol since the service consumer negotiates at the same time with the different providers of each functionality required in the SBA, as well as with the providers of the different functionalities required in the SBA in a coordinated way. In fact, when dealing with end-to-end QoS requirements typical of SBAs, the QoS values of each functionality are not independent from one another, but it is necessary to find a set of interrelated QoS values. So, the negotiation process may require several iterations to successfully end, becoming computationally expensive in terms of the involved communication costs.

In this paper we propose an experimental evaluation of the proposed negotiation mechanism in terms of its computational costs due to the communication overhead coming from the possibility to negotiate with all the available providers during each iteration of the negotiation. The aim of the evaluation is to compare the cost of negotiation with respect to the potential benefit of having a success at the end of the process, and to evaluate the pros and cons in negotiating with all available providers until the negotiation ends.

The paper is organized as follows. In Section II some works related to service composition are reported; Section III describes the main features of the negotiation mechanism adopted in the present work; in Section IV the rationale of the experiments set to evaluate the cost of the negotiation protocol is reported, together with the decision making mechanisms adopted by the service compositor and the service providers during the negotiation. Then the experiments carried out, and the evaluation of the obtained results are described and discussed in Subsections IV-B, and IV-C. Finally, Section V reports some conclusions and planned future works.

II. Related Works and Background

Service composition allows to aggregate autonomous and independently developed services in order to build added value applications (SBAs). With the pervasive growth of available services, it is widely recognized that multiple services providing the same functionality may be available, but they may differ in their non-functional features, usually known as Quality of Service, such as cost, execution time, and so on. In this context, users will require SBAs specifying their own preferences/constraints on the global QoS values of the application, so it becomes crucial to select the appropriate component services, i.e., services whose QoS attribute values, once aggregated, meet users' preferences/constraints.

Several research efforts addressed this challenge proposing approaches that mainly apply static methods to find the set of services whose QoS values meet the global constraints set by the users. Some works propose algorithms to select service implementations relying on the optimization of a weighted sum of global QoS parameters as in [6] by using integer linear programming methods. In [7] local constraints are included in the linear programming model used to satisfy global QoS constraints. In [8] Mixed Integer Programming is used to find the optimal decomposition of global QoS constraints into local constraints so that the best services satisfying the local constraints can be found. Typically, these works rely on static approaches assuming that QoS values of each service are predefined by providers and do not change during the selection process.

In dynamic markets where service provision is regulated by demand and supply mechanisms, it is likely that different users may have different QoS requirements for the same (from a functional point of view) application, as well as QoS attribute values for the same service may change in time according to dynamic circumstances affecting service provision strategies. In this context, it becomes crucial to provide service-oriented infrastructures with mechanisms enabling the selection of services with suitable QoS attribute values so that QoS requirements can be satisfied when forming new added-value applications through service composition. Such mechanisms should allow to manage the dynamic nature of both QoS values, and QoS requirements. Negotiation has gained more and more attention in SOC applications as a viable approach to drive the selection of suitable component services. It allows to address the dynamic nature of both the provided and required QoS since the offered QoS value of single services may change as soon as new offers and counteroffers are exchanged.

Practical negotiation mechanisms for B2B applications must be computationally efficient [9]. This implies that the interaction rules have to guarantee the quick end of the process and that agents behaviors and negotiation strategies should be developed based on the assumption of bounded rather than perfect rationality [10]. One of the common requirements for a negotiation protocol is the monotonicity of the utilities of the offers as in [11]. This allows to guarantee the end of the process without a deadline: either an agreement is reached (sooner or later), or a conflict is reached in the case all agents stop to concede in utility.

Most approaches, that use negotiation mechanisms to select services according to their QoS values, usually apply negotiation for each required service independently from the others relying on bilateral one-to-one negotiation mechanisms [4], [12]. Attempts to propose a coordinated negotiation with all the providers of the different required services in a composition have been proposed as in [2], but they introduce a Negotiation Coordinator that instructs the negotiation of the single component services by decomposing end-to-end QoS into local QoS requirements, so making the negotiation process computationally heavier from the point of view both of the involved negotiators, and of the necessary decision making mechanisms.

III. One-to-Many-to-Many Negotiation Protocol

In this work we adopt the iterated negotiation mechanism proposed in [5], starting from the assumption that SLAs for QoS-aware SBAs have to be set by coordinating the single agreements of each component service.

In the proposed approach a *Service Compositor* (SC), acting on behalf of a service consumer, issues an SBA request represented by a Directed Acyclic Graph, referred to as an *Abstract Workflow* (AW), specifying the functionality of each service component (AW nodes referred to as *Abstract Service*s ASs), and their functional dependence constraints (AW arcs), together with the value(s) of the end-to-end QoS requirements the user wants the application to provide. It is assumed that for each AS a set of *Concrete Service*s (CSs) are available on the market, each one provided by a specific *Service Provider* (SP) with QoS attributes whose values are set by the corresponding SP dynamically. The protocol allows only the SPs to formulate new offers, and only the SC to evaluate them. The rationale of this choice is twofold: on one hand it makes it possible to simulate what happens in a real market of services where an SC does not have enough information on the SPs strategies to formulate counteroffers; on the other hand it takes into account that the offers for a single functionality cannot be evaluated independently from the ones received for the other functionalities. So, it is necessary to design a negotiation mechanism that allows both to negotiate with the SPs providing services for each required functionality in the AW, but at the same time to evaluate the aggregated QoS value of the received offers for all the required functionality in the AW during the negotiation. Indeed, the SC is not able to make single counter-proposals with respect to each received offer, because the change of a value of a particular QoS can impact the others QoS attributes of the same service, as well as the constraints to be fulfilled by the QoSs of the other services. In other words, negotiating over the attributes of the single AS cannot be done independently from each other.

Since SC does not provide counteroffers the negotiation could be model as simply an auction mechanism as in [13]. However, in order to model a real market of services, it cannot be assumed that all providers, providing different functionalities, follow the same rules when bidding (such as Vickrey, English, and so on), as it happens in auctions mechanisms. In fact, rules may vary according to the type of provided service (i.e., its functionality), and above all according to the trends of the market that may vary quickly, and not in the same way for all the QoS attributes. With auction mechanisms, each bidder may have its own strategy, but once the type of auction is decided, then all bidders know the rules and they have to stick to them until the auction ends. Moreover, a simple auction mechanism cannot be used in our setting because of the interdependence among the QoS attributes of the component services. In fact, it would not be possible to award an auction winner without evaluating the offers for a given AS with respect to the ones received for the other ASs in the AW. We argue that these solutions do not model what happens in real markets of services where predefined bidding mechanisms cannot be assumed and fixed for all the service types and for all the considered QoS attributes. Therefore, traditional methods with the protocols and strategies hard-coded in the agents would not work in real market of services that are open systems.

This is why an hybrid negotiation approach was used, where an auction-like protocol models the bidding of single component services, but without relying on a specific auction mechanism to allow SPs to adopt their own private strategies when bidding, and also to change them if required by the market trends.

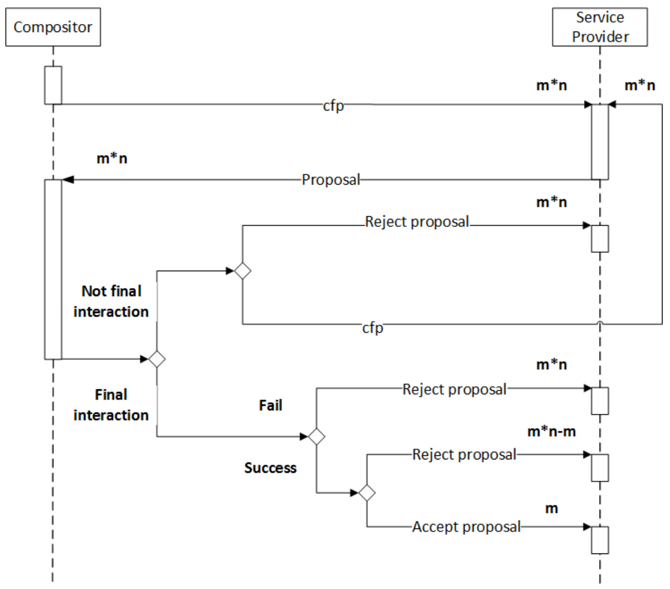


Figure 1. The negotiation protocol.

In [5], we presented a *one-to-many-to-many* protocol (OMM) for the dynamic selection of services, based on the FIPA Contract Net Iterated Protocol [14], [15] (ICNET), one of the most used protocols for negotiating SLAs [4]. As described in Figure 1, the negotiation occurs between the SC, that is the *initiator* of the negotiation, and the SPs available for each AS of the AW, and it may be iterated for a variable number of times until a *deadline* is reached or the negotiation is successful. The deadline is the number of allowed rounds. The SC prepares m call for proposals (`cfps`), one for each AS in the AW and, assuming that there are n SPs for each of the m AS, it sends $m * n$ `cfps` at each negotiation round. After waiting for the time set to receive offers, if there are not offers for each AS in the AW, the SC rejects the received proposals (`reject proposal`) since it is not possible to find a CS corresponding to each AS. Otherwise, it evaluates the received offers, and, if the QoS value obtained by aggregating the received offers does not meet the user request, it starts another negotiation round sending $m * n$ `reject proposals`, and, at the same time, new $m*n$ `cfps`. If the QoS values of the received offers, once aggregated meet the user request, it accepts the offers sent by the corresponding SPs (sending m `accept proposals` and $m*n-m$ `reject proposals`). If the deadline is reached without a success, the negotiation ends.

IV. Evaluating the Negotiation Cost

We evaluate experimentally the efficiency of the negotiation mechanism with respect to two performance measures: negotiation outcome (i.e., utility of the solution and/or the negotiation success rate), and communication complexity (i.e., the number of exchanged messages), by varying the parameters affecting the OMM protocol that are the number of allowed negotiation rounds, and the number of SPs involved in the interaction.

A. Compositor and Providers Utility Functions and Strategies

Here, we briefly describe the decision making algorithm for the SC evaluation of proposals, and the strategies adopted by SPs to generate an offer, as proposed in [5], considering the case of a single additive QoS parameter (the parameter considered here is the "price").

Following the approach formulated in [8], the SC first evaluates the utility of the offer provided by the jth SP for the ith AS, with respect to both the other offers for the same AS (local evaluation), and to the entire workflow (global evaluation):

$$U_{SC}(o_{i,j}) = \frac{max_k(price_{i,k}) - price_{i,j}}{\sum_{i=1}^{m}(max_k(price_{i,k}) - min_k(price_{i,k}))} \quad (1)$$

where i identifies one of the m ASs and the j identifies one of the n SPs. Once the most promising offer for each AS is selected according to Eq. 1, we modeled the global requirements satisfaction problem in terms of SC's global utility. This utility is related to the distance between the QoS preferences, expressed at the time the request is issued, and the aggregated QoS values obtained by combining the the best selected offers. It is normalized so that it is 1 in case the requirements are met, and in $[0, 1]$ otherwise. The SC's utility is expressed as follows:

$$U_{SC} = \begin{cases} 1 \quad if \quad \sum_{i=1}^{m} price_{i,s} < reqPrice \\ 1 - \frac{\sum_{i=1}^{m} price_{i,s} - reqPrice}{reqPrice} \quad otherwise \end{cases} \quad (2)$$

where, $price_{i,s}$ is the price offered for the ith AS by the selected sth SP, $\sum_{i=1}^{m} price_{i,s}$ is the aggregated value for the price, and $reqPrice$ is the user requested price.

On the contrary, SPs apply their own strategies to formulate their offers. These strategies are modeled as a set of functions that are both time and resource dependent [16], and they take into account both the *computational load* of the provider, and the *computational cost* of the provided service. The computational load of the provider accounts for the number of requests it agreed to fulfill, i.e., the amount of service implementations it will deliver, while the computational cost of the service represents a measure of the complexity of the provided service, i.e., the more complex the service is the higher its expected cost is. SPs strategies to concede in utility are modeled as Gaussian distributions [5]. The mean value of the distribution $maxU$ is the best offer the SP may propose in terms of its own utility and as such it has the highest probability to be selected. The standard deviation σ represents the attitude of the SP to concede during negotiation and it varies from SP to SP providing the same AS, so that the lower the SP's computational load is, the more it is available to concede in utility and the lower its reservation value is. The best offer $maxU$ is the same for all SPs of the same AS. This assumption models a scenario where services providing the same functionality have the same "market price" corresponding to the maximum utility for the SP providing that service. At each negotiation round, the SP generates, following its provision strategy, a new value of utility corresponding to a new offer to be sent to the SC by comparing the new offer with the previously generated one to decide whether to submit it or not. This is a variation of the standard negotiation mechanisms where the utility of a new generated offer is compared with the last one generated by the other negotiation partner.

B. *Experimental Settings*

The configurations considered for the experiments set five different deadlines at 1, 10, 20, 30 and 40 rounds, and for each deadline the number of SPs at 2, 4, 8, 16. Let us highlight that for a deadline of 1 round, the protocol is based on m concurrent ContractNet Protocols (CNET). We also considered a configuration where the negotiation occurs only with one SP for each AS, that is the best SP, in terms of $U_{SC}(o_x)$, according to the offers received at round 1. In this case the deadline varies from 10 to 40 rounds.

In the configurations there are 5 ASs, AS_1, AS_2, AS_3, AS_4, AS_5, with a computational costs decreasing from AS_1 to AS_5. The user requested price is 1500$ ($reqPrice$). For each AS a default price $bestPrice_i$ is set, and the corresponding SPs will send as initial offer a price randomly extracted in the neighborhood of $bestPrice_i$ $[bestPrice_i - 5\% * bestPrice_i, bestPrice_i]$. This is because, even though the market price of services corresponding to the same AS is the same, to model a real market of services the variability of the first offered price is introduced. In the experiments, the market prices for the ASs are: $bestPrice_1 = 540\$, bestPrice_2 = 468\$, bestPrice_3 = 351\$, bestPrice_4 = 270\$, bestPrice_5 = 216\$$. The σ value randomly varies for each SP in the range [0.0, 0.5], so including the possibility that SPs with the maximum computational load are not willing to concede.

C. *Evaluation of Results*

In all the experiments 100 tests were performed for each of the described configurations.

Table I. Success rate varying the number of rounds with only the "best" SP for each AS.

Rounds	10	20	30	40
1SP for AS	1%	6%	12%	12%

In Figure 2 the percentage of negotiation successes varying the number of rounds and the number of SPs (from 2 to 16) for each AS is plotted. In Table I the same percentage is plotted, varying the number of rounds, but in the case of negotiation with only the "best" SP for each AS. As expected, for a deadline of 1 round (simple CNET protocol) we have 100% of failures since the specific settings of the tests require a negotiation phase to find a solution. In Table I the results show that selecting the best SP at round 1 does not reduce the negotiation cost. In fact, only the 12% of success rate is obtained with 30 or 40 rounds of negotiation allowed, so the computational cost of the negotiation is not compensated by an high rate of success. A better trend is obtained by adding another provider for each AS (as shown in Figure 2). In fact, in this case, with 30 or 40 rounds of negotiation allowed, 50% of successes are obtained. Scaling up the number of SPs from 4 to 16 the success rate increases from 90% to 100% just after 10 rounds. These results support the choice to negotiate with all the available SPs, as proposed by the OMM protocol, since the cost of negotiation is partially compensated by an increase in the negotiation success rate.

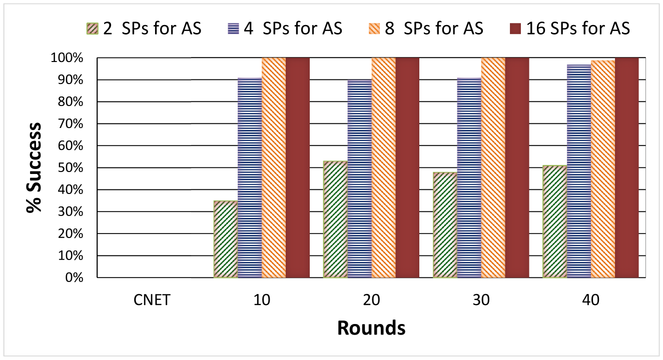


Figure 2. Success rate varying the number of rounds and the number of SPs for each AS.

Table II. Number of messages varying the number of SPs and the deadline.

		# SPs				
		1	2	4	8	16
# rounds	1	15	30	60	120	240
	10	150	300	600	1200	2400
	20	300	600	1200	2400	4800
	30	450	900	1800	3600	7200
	40	600	1200	2400	4800	9600

In order to evaluate the computational cost of the OMM protocol, also the number of exchanged messages are considered. At each negotiation round the SC sends $m * n$ cfps, receives at the most $m * n$ possible offers, and it sends back at most $m * n$ accept and/or reject messages. This means that for each round the cost of communication in terms of exchanged messages is $3 * n * m$. The numbers of messages are reported in Table II for all the experimental configurations.

A comparative evaluation of Table II and Figure 2 is shown in Figure 3 that provides information on the trade-off between communication costs and success rate. In particular from configurations with 2400 exchanged messages to configurations with 9600 no variation in success rate is obtained (that is stable at 100%). This means that from 2400 onward there is only a communication overhead without any gain in the success rate. A first conclusion of this evaluation is that negotiating with 16 SPs for each AS with respect to

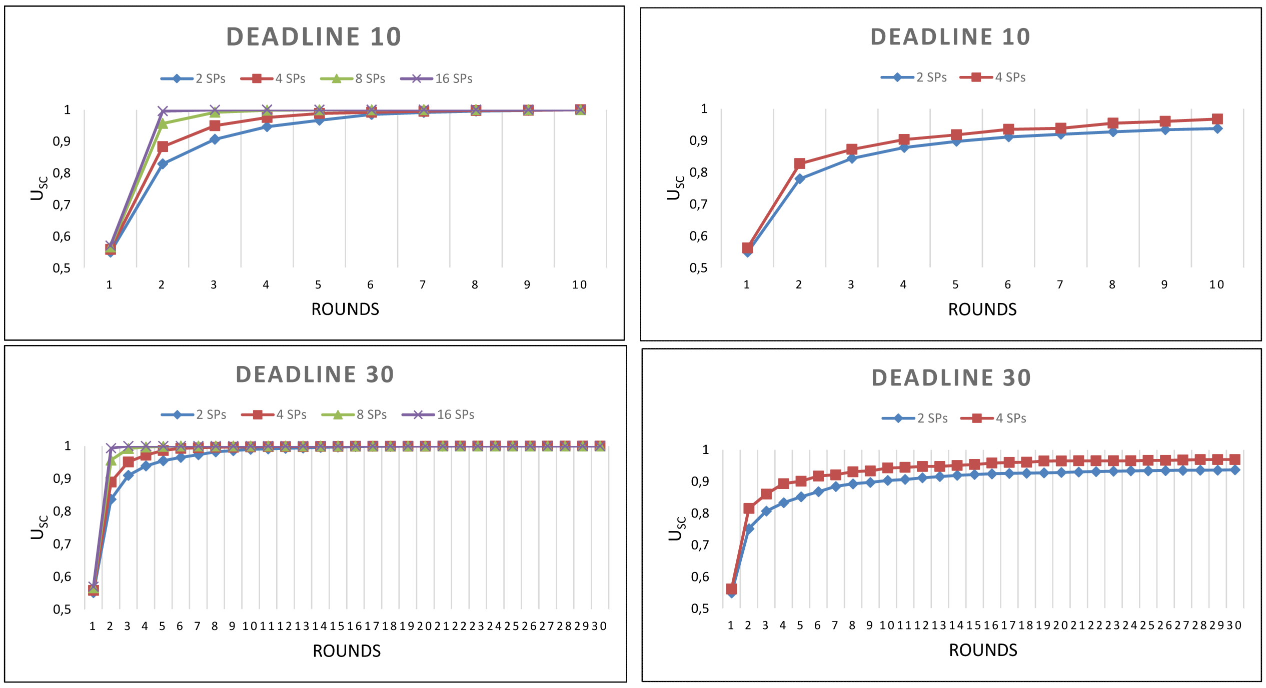


Figure 4. The SC's utility for different configurations in case of tests that led to a success (on the left) or to failure (on the right).

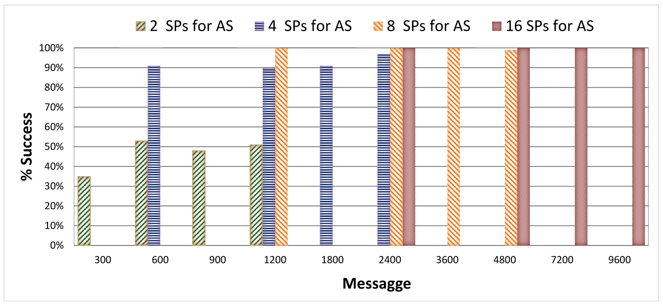


Figure 3. Success rate evaluated with respect to the number of messages.

negotiating with 8 SPs will not change the success rate, but it will only require more exchanged messages. So, in this case, selecting a subset of available providers reduces the cost of communication without affecting the success rate. Moreover, as already showed in Figure 2.b, such selection cannot be made only evaluating current offers because promising providers may change their concession strategy during the interaction. However, a bigger number of SPs, as shown in the following figures will provide a quicker achievement of the agreement, so reducing the negotiation length. Such evaluation of the number of exchanged messages with respect to the success rate shows that, in the case of a relevant number of available providers, long negotiation mechanisms are not necessary. So, in the following experimets only negotiation deadlines smaller than 40 rounds will be considered.

In Figure 4 the variation of the SC's global utility is reported at different negotiation rounds varying the number of available SPs and the deadline of the negotiation. In particular, on the left cases leading to success are considered varying the number of SPs for each AS from 2 to 16, and the deadline from 10 to 30 rounds. On the right cases leading to a failure are considered with the same deadlines as before, but varying the number of SPs for each AS from 2 to 4 since by increasing the number of SPs only successes are obtained from the round 10th onward. Let us note that the success is obtained as soon as the SC's utility becomes equal to 1, and it is reached in a less number of rounds by increasing the number of SPs for each AS. In case of failure, the SC's utility varies very little by increasing the number of the negotiation rounds, so making proceeding with negotiation expensive without any benefit.

Table III. SC'S UTILITY VARIATION IN CASE OF FAILURES.

		Round Range		
		10/20	20/30	30/40
#SPs	2	0,0223	0,0082	0,0042
	4	0,0221	0,0043	0,0083
Average		0,0222	0,0063	0,0062
STD		0,0002	0,0028	0,0029

In Table III the SC's utility variation in case of failure is reported in order to allow the SC to dynamically stop the negotiation according to its trend, i.e., according to whether and in which measure its utility is varying. The variation is calculated as a difference between the value of the SC's utility respectively at rounds 10 and 20 (for the first column), at rounds 20 and 30 (for the second column), at rounds 30 and 40 (for the third column). The variation is evaluated varying the number of SPs (2 and 4). The average SC's utility variation, and the corresponding standard deviation are also reported. As shown in Table III, in the configurations with the number of SPs equal to 2 and 4, by increasing the number of rounds the SC's utility variation is less than 1% after 20 rounds, so indicating that keeping on negotiating is not likely to lead to a success.

V. Conclusions and Future Works

Software agent negotiation is considered a promising approach for modeling the interactions between a service consumer and service providers when composing QoS-aware Service Based Applications. It is well recognized that the provision of such applications will be regulated by an agreement between the application consumer and the providers of the component services stating agreed terms and conditions related to both functional and non-functional application features. In particular, the negotiation mechanism is suitable to model the interactions occurring among consumers and providers when services are provided according to market-based mechanisms and when dynamic features, like QoS ones, have to be considered.

Nevertheless, automated negotiation did not succeed in real service-oriented market scenarios since it is computationally expensive in terms of the involved decision making mechanisms, and the communication overhead deriving from the complexity of the communication patterns.

In this work, an evaluation of the cost of the negotiation mechanism proposed in [5] is carried out with the aim to extract useful information to limit the length of the negotiation and also its communication overhead.

The experiments were carried out for different configurations obtained by varying the two parameters affecting the cost of communication that are the number of involved negotiators, i.e., the number of SPs, and the number of negotiation rounds determining the length of the negotiation. The results showed that the increase in communication costs due to the possibility of negotiating with all the SPs instead of just one for each service type, is partially compensated by the fact that by increasing the number of SPs the success rate of the negotiation increases. In fact, in the case the best SP is selected to continue the negotiation, a 100% failure is obtained also by increasing the length of the negotiation. This is because in a market of services, that is dynamic by nature, it is not possible to assume that a promising provider will keep on sending promising offers, because a less promising provider may change its strategy in the meantime according to market trends and/or market strategies that are tied also to the specific service or quality attribute. So, the choice of negotiating with all the SPs is supported by the obtained results. Furthermore, in most configurations, the overhead due to the communication cost is partially compensated by a decrease in the negotiation length, i.e its overall computational cost.

In some cases the negotiation progress in terms of the distance between the requested QoS and the QoS obtained at each negotiation round shows that it is not worth to proceed with the negotiation after a certain number of rounds since no gain is obtained in the success rate. This means that in these cases the negotiation can be stopped without any loss in terms of consumer's utility, so limiting the cost of negotiation in terms of its length. Of course, these results are related to the considered configurations, and also to the specific strategies adopted for the SPs.

We plan to extend the proposed negotiation mechanism by including the possibility for the SPs to change their strategies on fly, and to carry out more experiments by considering different set of strategies to evaluate the negotiation cost in different experimental settings.

Acknowledgements

The research leading to these results has received funding from the EU FP7-ICT-2012-8 under the MIDAS Project (Model and Inference Driven - Automated testing of Services architectures), Grant Agreement no. 318786, and the Italian Ministry of University and Research and EU under the PON OR.C.HE.S.T.R.A. project (ORganization of Cultural HEritage for Smart Tourism and Real-time Accessibility).

References

[1] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[2] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. K. Goh, and J. Zhang, "Autonomous service level agreement negotiation for service composition provision," *Future Generation Computer Systems*, vol. 23, no. 6, pp. 748 – 759, 2007.

[3] A. Keller and H. Ludwig, "The wsla framework: Specifying and monitoring service level agreements for web services," *J. Network System Management*, vol. 11, no. 1, pp. 57–81, 2003.

[4] S. Paurobally, V. Tamma, and M. Wooldrdige, "A framework for web service negotiation," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 4, Nov. 2007.

[5] C. Napoli, P. Pisa, and S. Rossi, "Towards a dynamic negotiation mechanism for qos-aware service markets," in *Trends in Practical Applications of Agents and Multiagent Systems*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2013, vol. 221, pp. 9–16.

[6] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, may 2004.

[7] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. on Software Eng.*, vol. 33, no. 6, pp. 369–384, 2007.

[8] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th Int. Conf. on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 881–890.

[9] R. Y. K. Lau, "Towards a web services and intelligent agents-based negotiation system for b2b ecommerce," *Electronic Commerce Research and Applications*, vol. 6, no. 3, pp. 260–273, 2007.

[10] A. Lomuscio, M. Wooldridge, and N. Jennings, "A classification scheme for negotiation in electronic commerce," *Group Decision and Negotiation*, vol. 12, no. 1, pp. 31–56, 2003.

[11] G. Zlotkin and J. S. Rosenschein, "Mechanism design for automated negotiation, and its application to task oriented domains," *Artif. Intell.*, vol. 86, no. 2, pp. 195–244, 1996.

[12] F. Siala and K. Ghedira, "A multi-agent selection of web service providers driven by composite qos," in *Proc. of 2011 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2011, pp. 55–60.

[13] P. R. Wurman, M. P. Wellman, and W. E. Walsh, "The michigan internet auctionbot: a configurable auction server for human and software agents," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. New York, NY, USA: ACM, 1998, pp. 301–308.

[14] R. G. Smith, "The contract net protocol: High–level communication and control in a distributed problem solver," *IEEE Trans. on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[15] "Fipa iterated contract net interaction protocol specification."

[16] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation Decision Functions for Autonomous Agents," *Robotics and Autonomous Systems*, vol. 24, pp. 3–4, 1998.

Towards a Cloud-assisted and Agent-oriented Architecture for the Internet of Things

Giancarlo Fortino and Wilma Russo
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)
Università della Calabria
Via P. Bucci, cubo 41C, 87036, Rende (CS), Italy
g.fortino@unical.it, w.russo@unical.it

Abstract*— In the Internet of Things (IoT), all things (e.g. sensors, actuators, smart devices, smart objects, RFID, embedded computers, robots) have their identities, physical attributes, and interfaces. They will be seamlessly integrated into the information network such that they will become active participants in business, information and social processes wherever and whenever needed and proper. The technical realization of this vision is a complex challenge as distributed heterogeneous IoT components at different levels of abstractions need to cooperate among themselves, with conventional networked IT infrastructures, and also with human users. To cope with this issue, we propose the synergic exploitation of two complementary mainstream paradigms for large-scale distributed computing: the agent-oriented and the cloud computing paradigms. While the former can support the development of decentralized, dynamic, cooperating and open IoT systems in terms of multi-agent systems, the latter can empower the IoT objects with more computing and memory resources and effectively support system-wide higher-level mechanisms and policies. In this paper, we introduce a cloud-assisted and agent-oriented vision for IoT based on layered reference architecture. Finally, we briefly overview our agent-oriented middleware for cooperating smart objects and a sensor-cloud infrastructure that represent the basic building blocks for technically achieving such vision.

Keywords - Internet of Things; Smart Objects; Agent-oriented Computing; Cloud Computing, Middleware

I. INTRODUCTION

The Internet of Things (IoT) term usually refers to a world-wide network of interconnected heterogeneous objects (sensors, actuators, smart devices, smart objects, RFID, embedded computers, etc) uniquely addressable, based on standard communication protocols [1]. In the IoT, such objects have therefore their identities, physical attributes, and interfaces. They are seamlessly integrated into the information network such that they become active participants in business, information and social processes wherever and whenever needed and proper.

In particular, in this paper we model the IoT as a loosely coupled, decentralized system of cooperating smart objects (CSOs). A CSO is an autonomous, physical digital object augmented with sensing and/or actuating, processing, storing, and networking capabilities. CSOs are able to sense, store, and interpret information created within themselves and in the environment where they are situated, act on their own by also performing directed actuation, cooperate with each other, and exchange information with other kinds of IT devices/systems and human users.

The actual implementation and integration of IoT smart objects, their management as well as the development of real applications atop them, are complex challenges that require the synergic use of suitable paradigms and technology for large-scale distributed computing. To deal with this challenge, we propose the synergic exploitation of two complementary mainstream paradigms for large-scale distributed computing: (i) the *agent-oriented* paradigm, which fully supports the development of decentralized, dynamic, cooperating and open systems, and (ii) the *cloud computing* paradigm, which efficiently enables the empowering of computing and storage resources of IT systems.

The *Agent-oriented Computing* paradigm defines distributed software systems in terms multi-agent systems (MAS). In particular, agents are networked software programs that can perform specific tasks for a user and possess a degree of intelligence that permits them to perform parts of their tasks autonomously and to interact with their environment in a useful manner. The agent features perfectly fit the CSO features [2, 3]:

- *Autonomy*: agents/CSOs should be able to perform the majority of their problem solving tasks without the direct involvement of humans or other agents/CSOs, and they should have a degree of control over their own actions and their own internal state.
- *Interaction*: agents/CSOs should be able to interact, when they deem appropriate, with other software agents/SOs and humans to complete their own problem solving and support others with their activities where appropriate.
- *Responsiveness*: agents/CSOs should perceive their environment, in which they are situated and which may be the physical world, a user, a collection of agents, the Internet, and so forth, and respond in a timely fashion to changes that may occur in it.
- *Proactiveness*: agents/CSOs should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behavior and take the initiative where and when appropriate.

The *Cloud Computing* paradigm provides flexible, robust and powerful storage and computing resources, which enables dynamic data integration and fusion from multiple data sources [4]. In addition a Cloud-based approach can offer flexibility and adaptability in the management and deployment of data analysis workflows. The dynamic deployment of software components as Cloud services removes the need for new client applications to be developed and deployed when the user requirements change. This also introduces an intrinsic competitive environment for the development of better services. Cloud computing layers (Infrastructure as a Service - IaaS, Platform as a Service - PaaS, Software as a Service - SaaS) and software components (e.g., databases, data mining, workflow tools) can be customized to support a distributed real-time system for the management and analysis of IoT objects and data streams generated by IoT objects.

This paper proposes a cloud-assisted and agent-oriented vision of IoT based on a layered reference architecture. Furthermore, we briefly overview our agent-oriented middleware for CSOs and BodyCloud, a sensor-cloud infrastructure, that represent the basic building blocks, which will be purposely integrated, for technically achieving such vision.

The rest of this paper is organized as follows. Section II provides a look at glance of the mainstream IoT visions. In Section III our reference architecture for cloud-assisted and agent-oriented vision of IoT is proposed. Section IV overviews our agent-oriented middleware for cooperating smart objects and a sensor-cloud infrastructure, and discusses their integration requirements to realize the proposed reference architecture. Finally conclusions are drawn.

II. SMART OBJECT-ORIENTED IOT

IoT semantically means a world-wide network of interconnected things (an ecosystem of things), which are uniquely addressable and based on standard communication protocols [1]. Things include sensors, actuators, sensor networks, embedded systems, RFID tags and readers, and other soft sensors in different forms. These things can be deployed in different physical environments to support diversified applications domains. They are communication-oriented objects and provide identification and information storage (e.g. RFID tags), information collection (e.g. sensor networks), information processing (e.g. embedded devices and sensor networks), and control and actuation (e.g. embedded systems including smart actuators). The interesting advantage is that everything is "reachable" and can be "exploited". The main disadvantages are that such enormous heterogeneity makes distributed communication and management very complex; moreover, "intelligence" is not embedded and should be provided at a higher level by means of smart services and/or applications.

Beyond such low-level and network-oriented vision of IoT, the smart object-oriented vision is at a higher level of abstraction and promotes an ecosystem of smart objects based on the Internet [5]. In particular, in such vision IoT is viewed as a loosely coupled, decentralized system of smart objects (SOs), which are autonomous physical/digital objects augmented with sensing/actuating, processing, and networking capabilities. Although, in such vision, not all "things" can be directly exploited as the object granularity is coarser, communication among smart objects is homogenized by the adoption of the Internet protocols and "intelligence" is mainly embedded inside the objects themselves.

Figure 1 shows a high-level layered architecture for the smart object-oriented IoT vision, where the main layers are: *Application*, *Middleware*, *Internet* and *Smart Object*. In particular:

- The *Application* layer encompasses applications based not only on SOs but also on other IT infrastructures.
- The *Middleware* layer provides as set of mechanisms for the naming, discovery, high-level interaction and state management of SOs.
- The *Internet* layer includes application, transport, and network protocols for supporting the communication with SOs and among SOs.
- The *Smart Object* layer offers programming frameworks and tools enabling the design and implementation of SOs.

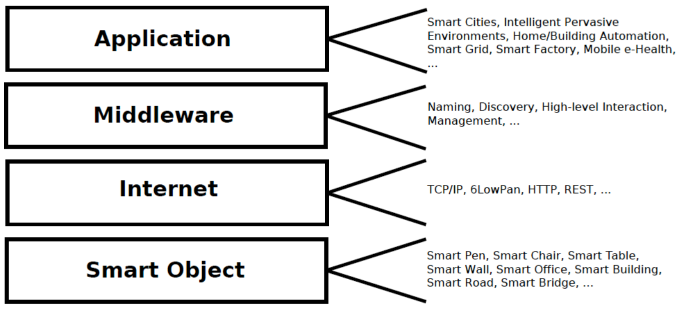

Application
Middleware
Internet
Smart Object
Smart Cities, Intelligent Pervasive Environments, Home/Building Automation, Smart Grid, Smart Factory, Mobile e-Health, ...
Naming, Discovery, High-level Interaction, Management, ...
TCP/IP, 6LowPan, HTTP, REST, ...
Smart Pen, Smart Chair, Smart Table, Smart Wall, Smart Office, Smart Building, Smart Road, Smart Bridge, ...

Figure 1. SO-oriented IoT architecture

Even though the standardization process of the SO communications based on IP is supported by the IP for Smart Objects (IPSO) alliance [6], the availability of middleware and frameworks that support development and management of SOs is still limited. Nowadays, apart from several available middlewares for smart environments (e.g. Smart-Its, 2WEAR, Ambient Agoras, Aura, Gaia, iRoom) [7, 8] that are not centered on the very concept of SO, a few SO-specific middlewares have been so far proposed: UbiComp [9], FeDNet [10], Smart Products [11], and ACOSO [12, 13].

UbiComp [9] defines a paradigm providing conceptual abstractions, the plug/synapse model and a middleware named GAS (Gadgetware Architectural Style)-OS, which is installed on each SO, to manage SO as components of distributed applications composed of ubiquitous computing services.

FeDNet [10] is based on a data-centric approach. Specifically, FeDNet uses XML-based documents to describe the requirements of an SO application, without considering the management of the SOs. Therefore, the services offered by SOs are described through structured documents. On the basis of such documents, the run-time FeDNet infrastructure provides a semantic association between the applications and the SOs. In FeDNet, SOs are objects of the daily life with computing and communication capabilities. However, as SOs are not proactive, task-oriented FeDNet applications are able to provide proactivity by SO orchestration.

Smart Products [11] aim at the development of SO equipped with proactive knowledge. Such knowledge is exploited to communicate and cooperate with human users, other smart objects and the external environment. SOs rely on MundoCore, a communication middleware, and a set of well-defined ontologies to enable effective cooperation among SOs.

ACOSO (Agent-based COoperating Smart Objects) [12, 13] is a middleware providing an agent-oriented programming model for CSOs and tools for their effective development. CSOs are based on an event-driven proactive architecture and on two different communication models (message passing and publish/subscribe). ACOSO is currently available atop JADE [14] and JADEX [15]. A more detailed description of ACOSO is reported in Sect. IV.A.

III. A Cloud-assisted and Agent-oriented Architecture for IoT

To deal with the complexity of the SO-based IoT, we propose a high-level architecture based on cloud and agent-based computing. The architecture named CA-IoT (Cloud-assisted and Agent-based IoT) is shown in Figure 2. The architecture components are:

- The *Smart User Agent*, which models human users in the context of specific smart systems. They therefore provide GUI-based functionalities through which users can request services and/or formalize specific service requests.
- The *Smart Interface Agent*, which defines an interfacing agent such as brokers, mediators, wrappers. Specifically, they are able to interact with and/or wrap components of the external IT systems.
- The *Smart Object Agent*, which models a CSO.
- The *CyberPhysical Environment*, which refers to the non-agent-oriented logical and physical context (made up of logical and physical components) in which agents are embedded. It can be modeled in terms of a reactive/proactive environment abstraction [16] that is able to interact with agents according to a specific coordination model.
- The *Cloud Computing Platform*, which supports all smart agents, empowering their specific resources, and allows for the definition of new (virtual) smart object agents as meta-aggregation of existing smart object agents.

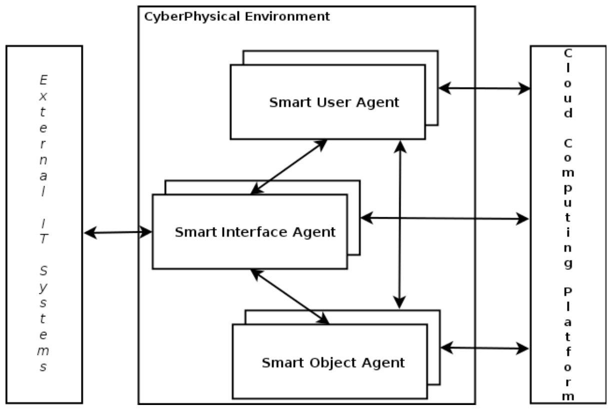


Figure 2. High-level architecture for Cloud-assisted and Agent-based IoT

IV. Integrating Smart Object Middleware and Cloud Platform for Large-scale IoT Systems Management

A. *An Agent-oriented Middleware for the Development and Management of CSOs*

The ACOSO middleware allows for the development and management of CSOs, which are modeled as agents that can cooperate with each other and with non-agent cyber-physical entities to fulfill specific goals. An ecosystem of CSOs therefore forms a multi-agent system (MAS). ACOSO currently relies on JADE that provides an effective agent management and communication support and on an external smart object discovery service that is purposely integrated into ACOSO [17]. Specifically, CSOs can be implemented as either JADE or JADEX agents, atop both Java-based and Android-based devices, and can cooperate by a direct coordination model based on ACL message passing and/or by a spatio-temporal decoupled coordination model relying on a topic-based publish/subscribe mechanism. Figure 3 shows the JADE-based ACOSO platform for CSO development and deployment. The platform is composed of three layers:

- The *high-level CSO architecture*, which is the reference architectural agent-oriented model for CSOs.
- The *JADE-based agent middleware*, which provides an implementation of the high-level CSO architecture through JADE and JADEX atop different computing devices (PCs and mobile devices).
- The *WSAN programming and management*, which is based on the Building Management Framework (BMF[1]) [18], for the management of the wireless sensor and actuator network (WSAN) of smart space-oriented CSOs, and on the Signal In-Node Processing Environment (SPINE[2]) [19, 20], for the management of the body area network (BAN) of body/personal oriented CSOs.

[1] http://bmf.deis.unical.it
[2] http://spine.deis.unical.it

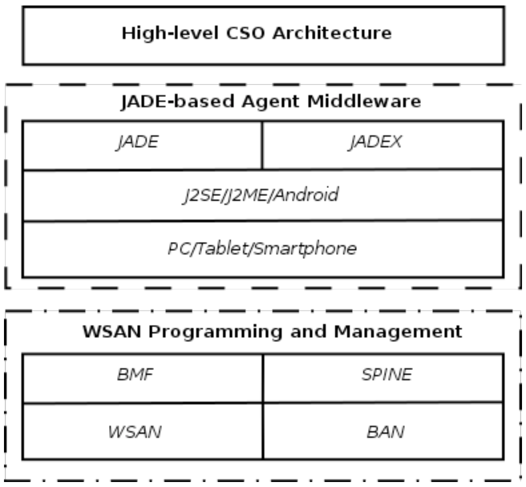

High-level CSO Architecture
JADE-based Agent Middleware
JADE
JADEX
J2SE/J2ME/Android
PC/Tablet/Smartphone
WSAN Programming and Management
BMF
SPINE
WSAN
BAN

Figure 3. Agent-based Platform for Smart Object Development

Figure 4 shows the JADE-based CSO architecture. CSOs are agents of the JADE platform (either simple JADE agents or encapsulated JADEX agents). They are therefore managed by the AMS (Agent Management System), communicate through the ACL-based message transport system, and use the Directory Facilitator (DF) to look up CSOs and other agents. The DF, which supports dynamic agent service discovery, was purposely modified/extended in ACOSO to allow searching CSOs on the basis of their specific metadata: type, offered services, location, static hw properties, dynamic system properties. Currently, the extended JADE DF receives requests from the JADE-based CSOs and fulfills them by using the remote interface of an external CSO discovery service [17]. The main CSO architecture components are:

- The *Task Management Subsystem*, which manages the reactive and proactive tasks of CSOs. In particular, tasks are event-driven and state-based software components encapsulating specific objectives to fulfill through computation, communication, sensing/actuation, and storage management operations. Tasks can be defined as JADE Behaviours or JADEX Plans so their execution is based on the mechanisms provided by the basic JADE behavioral execution model or plan-oriented Jadex execution model, respectively. Being tasks driven by events, external CSO communication, signals to/from the CSO devices, data to/from the knowledge base (KB) are internally formalized and managed as events.
- The *Communication Management Subsystem*, which provides a common interface for CSO communications. In particular, message-based communication is based on the ACL-based MessagingService whereas publish/subscribe coordination is the TopicManagementService. The subsystem is internally organized in handlers. The CommunicationManagerMessageHandler, which is implemented as Behaviour in JADE and as Plan in Jadex, captures the ACL messages targeting CSOs and translates them into internal events. Moreover, the TCPAdapter and UDPAdapter manage communication with external networked entities based on TCP and UDP, respectively.
- The *Device Management Subsystem*, which manages the sensing/actuation devices that belong to the CSO. It is organized in a DeviceManager handling several DeviceAdapters. Currently, two DeviceAdapters are available: the BMFAdapter, which allows to manage WSANs based on BMF, and the SPINEAdapter, which allows to manage BANs based on SPINE. BMF and SPINE are based on IoT standards protocols such as IEEE 802.15.4, ZigBee, and 6LowPan.
- The *KB Management Subsystem*, which supports CSOs through a knowledge base (KB). It consists of a KBManager, which manages and coordinates different KBAdapters, and a KBAdapter, which manages a KB containing the knowledge of the CSO. KB can be local and/or remote and archives information that can be shared among tasks.

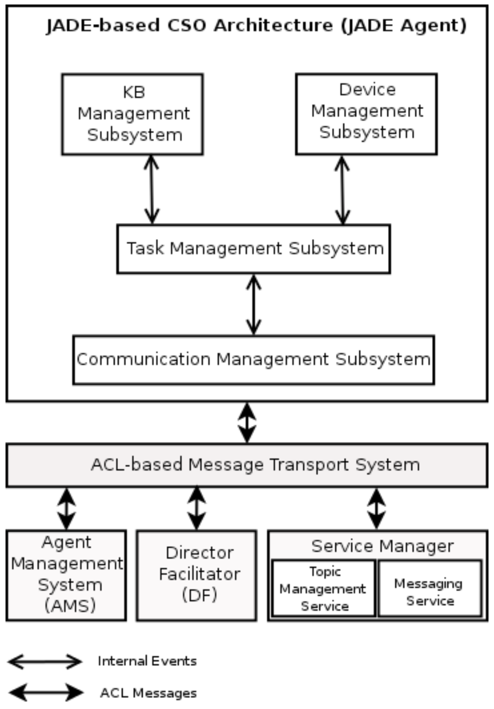

JADE-based CSO Architecture (JADE Agent)
KB Management Subsystem
Device Management Subsystem
Task Management Subsystem
Communication Management Subsystem
ACL-based Message Transport System
Agent Management System (AMS)
Director Facilitator (DF)
Service Manager
Topic Management Service
Messaging Service
Internal Events
ACL Messages

Figure 4. JADE-based CSO Architecture

Moreover, to support small-size CSOs running on a single sensor node, the Mobile Agent Platform for SunSPOT (MAPS[3]) framework [21] can be adopted to implement the CSO architecture. It is worth noting that MAPS agents can be wrapped by JADE agents through the JADE-MAPS gateway [22] so as to interact with JADE-based CSO agents.

B. *BodyCloud: an architecture for Cloud-assisted body area networks*

BodyCloud [23, 24, 25] is a SaaS-oriented architecture for the integration of BANs and a Cloud PaaS infrastructure. Its

[3] http://maps.deis.unical.it

architecture, shown in Figure 5, consists of four main subsystems (or sides):

- The *Body-side* manages the BAN and sends the collected sensor data to the Cloud-side through an Android-enabled mobile device. In particular, data acquisition is currently based on Android-SPINE, the Android version of SPINE. It allows Android-enabled smartphones and tablets to be used as coordinator of the BAN. In particular, the coordinator communicates with the wearable sensors by means of the SPINE application-level protocol atop Bluetooth and supports sensor discovery, sensor configuration, in-node processing, BSN activation/deactivation, data collection, and logging.
- The *Cloud-side* provides data collection and storage, processing/analysis and visualization. Each specific application can be defined through four programming abstractions: Group, Modality, Workflow/Node, and View. *Group* formalizes a specific application manipulating a well-defined BAN data source. *Modality* captures a specific interaction Body-Cloud sides and Viewer-Cloud sides. In particular, it models a specific service, such as data feeds to the Cloud-side, data analysis tasks, single-user or community applications. *Workflow* formalizes a data-flow process analyzing input data to generate output data. It consists of one or more nodes organized in a directed acyclic graph. *Node* is a specific algorithm that can be developed according to the Workflow Engine library (see Analyst-side in Fig. 5). A Node is uploaded to the Cloud-side where it can be used in different workflows. Finally, *View* defines the visualization layout of the output data for Viewers. Such abstractions are supported by the developed RESTLet-based SaaS Framework, which makes the interaction with the Cloud-side fully based on HTTP get, put, post, delete methods. The Cloud-side is supported by the Google App Engine PaaS[4] that enables data persistence and task execution.
- The *Analyst-side* supports the development of new application services. In particular, developers can create new BodyCloud services by defining the aforementioned abstractions. To program workflows, the Analyst-side is based on an appropriate development environment (XML Editor and Workflow Engine API).
- The *Viewer-side* visualizes the output produced by the data analysis through advanced graphical reports. By applying a View abstraction (see above) to the data, the graphical view is automatically generated. The current prototype is based on a Java library named jxReport that has been purposely implemented and integrated into the client application.

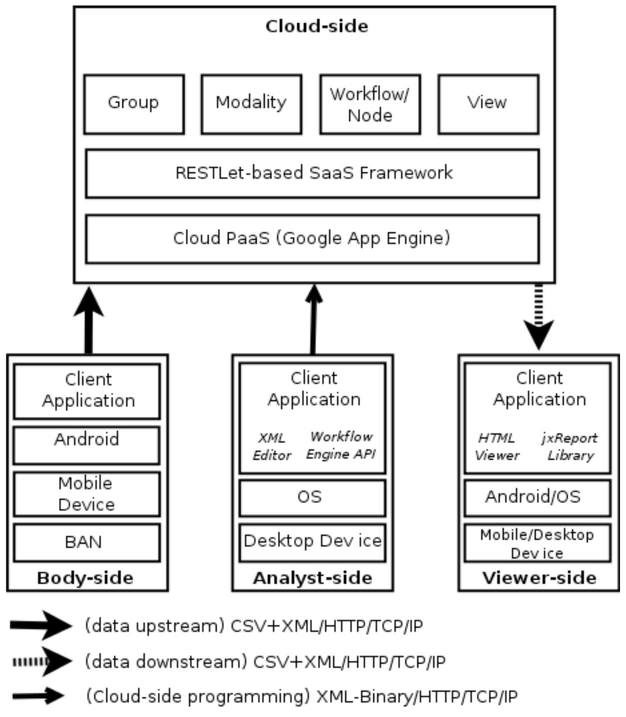


Figure 5. The architecture of BodyCloud

C. *Integration requirements*

To support the CA-IoT architecture introduced in Sect. III and thus develop large-scale IoT systems, ACOSO and BodyCloud are being jointly extended and integrated according to the following main requirements:

- *Smart agent enhancement*. While the basic smart agent layer is fully supported by ACOSO, the Cloud platform needs to provide a new functionality to define Cloud-based smart agents to dynamically create new virtual smart interface and object agents than run on the Cloud-side and seamlessly interact with the basic ACOSO agents.
- *Smart object data stream collection and management*. Data streams coming from highly decentralized smart objects needs to be efficiently uploaded onto the Cloud-side and here effectively managed.
- *Workflow-oriented analysis of smart object data*. Decision making applications should be dynamically developed through distributed workflows defined at the Cloud-side involving smart agents and cloud services.
- *Effective multi-level security architecture* for smart object data collection (from smart objects to the Cloud-side) and data analysis services (at Cloud-side).

V. Conclusions and Future Work

In this paper, we have proposed a novel high-level architecture for smart object-oriented IoT based on Cloud and Agents. Through the exploitation of agent-based computing, the proactiveness and cooperation capabilities of smart objects

[4] https://cloud.google.com/products/

could be effectively defined, whereas by means of cloud computing, smart objects could be empowered in terms of processing power and storage resources. Thus, cloud-assisted and agent-oriented smart objects could be used as basis for the development of large-scale IoT applications and systems. The proposed architecture can be actually implemented by adopting and integrating an effective smart object middleware and an efficient cloud platform. To this purpose, we briefly introduced ACOSO, an agent-oriented middleware for the programming and management of cooperative smart objects, and BodyCloud, an architecture for the integration of sensors on the Cloud. They are currently being integrated according to specific defined requirements to achieve a large-scale distributed platform for smart object-based IoT development. Finally, future work will be aimed at defining a novel methodology for the development of IoT applications. Such methodology will be obtained as extension of ELDAMeth [26, 27], a methodology for simulation-based prototyping of distributed agent systems.

ACKNOWLEDGMENT

Authors wish to thank Antonio Guerrieri and Michele Lacopo for their valuable efforts and contribution in the development of the smart object agent-oriented middleware and Giuseppe Di Fatta (University of Reading) for his valuable insights for the BodyCloud approach definition. This work has been partially supported by TETRis – TETRA Innovative Open Source Services, funded by the Italian Government (PON 01-00451). Giancarlo Fortino is also partially funded by DICET – INMOTO – Organization of Cultural Heritage for Smart Tourism and REal – Time Accessibility (OR.C.HE.S.T.R.A.) funded by the Italian Government (PON04a2_D).

REFERENCES

[1] D. Bandyopadhyay and J. Sen, “The internet of things - applications and challenges in technology and Standardization,” Springer International Journal of Wireless Personal Communications, 58(1), pp. 49-69, May 2011.

[2] M. Vinyals, J. A. Rodriguez-Aguilar, J. Cerquides, “A Survey on Sensor Networks from a Multiagent Perspective,” The Computer Journal, 54(3), pp. 455-470, 2010.

[3] A. Rogers, D. Corkill, and N.R. Jennings, N. R. “Agent technologies for sensor networks,” IEEE Intelligent Systems, 24, pp. 13-17, 2009.

[4] R. Hill, L. Hirsch, P. Lake, S. Moshiri, "Guide to Cloud Computing. Principles and Practice," Computer Comm. and Networks, Springer, 2013.

[5] G. Kortuem, F. Kawsar, V. Sundramoorthy, D. Fitton, "Smart Objects as Building Blocks for the Internet of Things," IEEE Internet Computing, 14(1), pp. 44-51, January/February, 2010.

[6] Adam Dunkels and JP Vasseur, “IP for Smart Objects”, Internet Protocol for Smart Objects (IPSO) Alliance White paper #1, Sept. 2008.

[7] G. Fortino, A. Guerrieri, W. Russo, "Middleware for Smart Objects: State-of-the-art and Research Challenges", in *Internet of Things based on Smart Objects: technology, middleware and applications*, Springer Series on the Internet of Things. 2014. *to appear*.

[8] L. Roalter, M. Kranz, and A. Moller, A Middleware for Intelligent Environments and the Internet of Things," in Proceedings of the 7th international conference on Ubiquitous intelligence and computing (UIC). Berlin, Heidelberg: Springer-Verlag, 2010, pp. 267--281.

[9] Christos Goumopoulos and Achilles Kameas, “Smart Objects as Components of UbiComp Applications,” International Journal of Multimedia and Ubiquitous Engineering, Vol. 4, No. 3, July, 2009.

[10] F. Kawsar, T. Nakajima, J. H. Park, and S. S. Yeo, "Design and implementation of a framework for building distributed smart object systems," Journal of Supercomputing, vol. 54, no. 1, pp. 4-28, Oct. 2010.

[11] M. Miche, D. Schreiber, D., and M. Hartmann, “Core Services for Smart Products,” In: AmI-Blocks'09, at AmI'09, 2009.

[12] G. Fortino, A. Guerrieri, and W. Russo. Agent-oriented smart objects development.In Proc. of IEEE 16th Int. Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 907--912, 2012.

[13] G. Fortino, A. Guerrieri, M. Lacopo, M. Lucia, and W. Russo, "An Agent-based Middleware for Cooperating Smart Objects", in Highlights on Practical Applications of Agents and Multi-Agent Systems, Communications in Comp. and Inform. Science (CCIS), Vol. 365, pp. 387-398, Springer, 2013.

[14] F. Bellifemine, A. Poggi, and G. Rimassa, “Developing multi agent systems with a FIPA-compliant agent framework,”. Software Practice And Experience 31, 103-128, 2001.

[15] A. Pokahr, L. Braubach, W. Lamersdorf, “Jadex: A BDI Reasoning Engine,” In Multi-Agent Programming: Languages, Platforms and Applications. Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol. 15, Springer, pp. 149-174, 2005.

[16] D. Weyns, A. Omicini, and J. Odell, Environment as a first-class abstraction in multiagent systems, International Journal on Autonomous Agents and Multi-Agent Systems 14 (1), pp. 5-30, 2007.

[17] G. Fortino, M. Lakovic, W. Russo, P. Trunfio, “A discovery service for smart objects over an agent-based middleware”, IDCS 2013, Lecture Notes in Computer Science (LNCS), Vol. 8823, pp. 1-14, 2013.

[18] G. Fortino, A. Guerrieri, G. O’Hare, A. Ruzzelli, “A Flexible Building Management Framework based on Wireless Sensor and Actuator Networks”, Journal of Network and Computer Applications, 35(6), pp. 1934-1952, 2012.

[19] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, “SPINE: A domain-specific framework for rapid prototyping of WBSN applications” Software Practice and Experience, 41(3), 2011, pp. 237-265.

[20] F. Bellifemine, F. Aiello, G. Fortino, S. Galzarano, R. Gravina, “An agent-based signal processing in-node environment for real-time human activity monitoring based on wireless body sensor networks”. Journal of Engineering Applications of Artificial Intelligence, Vol. 24, n. 7, pp. 1147-1161, 2011.

[21] F. Aiello, G. Fortino, R. Gravina and A. Guerrieri, A Java-based Agent Platform for Programming Wireless Sensor Networks, The Computer Journal, 54(3), pp. 439-454, 2011.

[22] M. Mesjasz, D. Cimadoro, S. Galzarano, M. Ganzha, Giancarlo Fortino, M. Paprzycki “Integrating JADE and MAPS for the development of Agent-oriented WSN-based distributed applications”, In Intelligent Distributed Computing VI, Springer, SCI series, Vol. 446, pp. 211-220, 2013.

[23] G. Fortino, D. Parisi, V. Pirrone, G. Di Fatta, “BodyCloud: A SaaS Approach for Community Body Sensor Networks,” in *Future Generation Computer Systems*. 2014. to appear.

[24] G. Fortino, G. Di Fatta, “Engineering Large-Scale Body Area Networks Applications”, accepted 8th International Conference on Body Area Networks (BodyNets), Sept. 30–Oct- 2, 2013 Boston (USA), ACM press.

[25] G. Fortino, M. Pathan, G. Di Fatta, "BodyCloud: Integration of Cloud Computing and Body Sensor Networks", IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012), Taipei, Taiwan, Dec 3-6, 2012.

[26] G. Fortino, A. Garro, S. Mascillaro, W. Russo, “Using Event-driven Lightweight DSC-based Agents for MAS Modeling,” in International Journal on Agent Oriented Software Engineering, 4(2), 2010.

[27] G. Fortino,W. Russo, “ELDAMeth: A Methodology For Simulation-based Prototyping of Distributed Agent Systems”, Information and Software Technology, 54(6), pp. 608-624, 2012.

Ontology and Goal Model in Designing BDI Multi-Agent Systems

Patrizia Ribino*, Massimo Cossentino*, Carmelo Lodato*, Salvatore Lopes*, Luca Sabatucci*, and Valeria Seidita†*
*Istituto di Calcolo e Reti ad Alte Prestazioni - Consiglio Nazionale delle Ricerche
Email: {cossentino, lodato, lopes, ribino, sabatucci}@pa.icar.cnr.it
†Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica
Email: valeria.seidita@unipa.it

***Abstract*—Nowadays several methodological approaches exist, each of them tightly tied up with the implementation platform supporting it. In this paper we propose an intermediate step toward the definition of a methodological approach for supporting the JACAMO framework. This paper resumes a previous work, focused on modeling BDI organizations, and we now address the requirements analysis phase. In particular, we propose the use of an ontological model and a goal model for representing requirements and the domain formalization respectively. The two portions of design process are connected by a heuristic process that allows to extract goals from the ontological model. The resulting models are also used for completing each other and for enhancing the problem description that is considered an input to our process. In the paper we use the well-known case study of the conference management system for illustrating the proposed portion of process.**

I. Introduction

It is widely accepted that methodological approaches offer significant advantages for system development. Accordingly, several complete methodological approaches have been proposed. Many of them are tightly tied up with the implementation platform supporting it. Each implementation platform gives support for the abstractions the methodology deals with and normally cannot be fit to methodologies different from the ones it has been created for.

In the past, we created and used AO methodologies such as PASSI [5] and ASPECS [6] that are strictly tied respectively to JADE [2] and Janus[11]. In the first case Jade allows to implement agents that principally are a state machine, whereas the second allows the implementation of holonic structures. A specific platform is suitable for a methodology in the sense that it supports its features, for instance we could not easily implement holonic structures/organizations with plain Jade.

In the latest period we have been experimenting the JaCaMo framework and its feature. JaCaMo is based on the BDI paradigm [15] and it provides an integration of tools and languages for programming the following dimensions: agents (Jason), environment (Cartago), and organisations ($\mathcal{M}$oise) [16]. Our work is focused on the normative and organizational aspects of developing BDI multi-agent systems under both the notational and methodological points of view. As regard the former, in [8] we illustrated a UML profile and a graphical notation for describing MASs based on the Jason metamodel. Whereas, as regard the methodological issues, our aim is to define a complete methodological approach that will include goal oriented analysis, the design of organizations and the design of agents based on the Jason platform; a first part of this work has been faced (see [7]) and a portion of methodology for developing MASs organized in hierarchical structures, implemented with $\mathcal{M}$oise+, has been completed.

The methodology we are creating is based on some cornerstones: *(i)* organizations, *(ii)* requirements expressed by goals and *(iii)* an ontological formalization of problem domain. This paper illustrates how we face goal modeling starting from a problem domain model represented by means of an ontology. Although we are aware that the use of ontologies is not shared unanimously by the software engineering community, we adhere to the trend that asserts ontologies may have a significant role in the model driven engineering and may offer several benefits to a design process (first of all to the requirement analysis)[1][18][3][17].

In such a context, the contribution of this paper is twofold. Firstly, we present two portions of design process able to cover the early requirements analysis phase resulting in problem specifications in terms of goals. Secondly, we propose the use of an ontology as an analysis (i.e: descriptive) model representing the reality of the problem domain in order to address the ambiguities and the incompleteness of the problem specifications. The two proposed activities result in an analysis model composed of an ontology and a goal model. The first one is used as a model for describing the problem domain by a set of meta classes (Concept, Action and Predicate). The second one is used to describe the objectives of the stakeholders involved in the problem and the dependencies among them. Hence, we have named these activities (portions of design process) *Problem Ontology Description* and *Goal Identification* respectively.

The ontological formalization we propose for performing Problem Ontology Description activity is thought to describe the intentional behaviors that typically characterize the class of problems addressed with agent oriented technologies. To this aim, inspired by the FIPA agent ontology [9], we introduce new elements in order to explicitly model intentional behaviors. These elements represent the problem domain entities able to act, according to their desires, in order to change the state of the world. Moreover, the instances of this kind of ontology present recurrent structures (*patterns*) that provide some useful information for identifying the goals that the solution of the problem has to satisfy.

The paper is organized as follows: section II focuses on the problem and sets our objectives, section III describes the portions of design process together with an example and, finally, in section IV some conclusions are drawn.

II. Motivation and Objectives

The concept of *goal* is widely used in requirements engineering since it allows, among other things, to reduce the gap between analysts' and stakeholders' knowledge. Many definitions exist for goals. In this paper, when talking about goal, we mean a condition or state of the world that the actor wants to achieve[19]. This definition highlights a very interesting connection with the agent oriented paradigm since it recalls the aspects of autonomy and proactiveness, typical of software agents.

In most methodologies (not only agent oriented ones) the analysis phase results in a model of the requirements and in a model of the problem; this latter is constructed in agreement with the used design paradigm and its abstractions. Usually, the problem model includes at least one structural view and one or more dynamic (behavioral) views, uses a domain specific language and has a direct counterpart with the implementation concepts. Some fundamental thoughts for the identification of proper MAS design abstractions have been presented in [12]. Here the *social level* deals with abstractions like organizations, collaborations, communications and other social aspects of the agents' life and the *knowledge level* sees agents as conceived in terms of the goals they have to achieve and the actions they may perform for pursuing them. An agent processes knowledge in order to attain its goals [14].

As a consequence, we think that useful models for AO developing should include (*i*) a model of requirements in terms of goals and (*ii*) a model of the problem in terms of abstractions to be mapped onto the AO paradigm. Such abstractions should come from both the social and knowledge levels: organization, communication, actions, predicates (beliefs) and so on.

Thanks to these considerations we may argue that a good MAS design methodology should include a *goal model*, an *ontology* and a *model of organizations*. The ontology describes the problem in terms of elements of the environment, their significative states, and what it can be done on them to achieve/mutate their states. Such an ontology should include, *concepts* of the domain, *predicates* (used to represent beliefs and significant states of the concepts), *actions* that can be done on concepts and *positions* that may willingly execute actions; such positions may represent human beings, agents or other not-AO systems.

The goal identification is the preliminary activity of the agent goal modeling phase in which analyst makes it explicit the strategic objectives of the system (intended as the sum of the software and its environment). User goals identification is typically conducted in the early phases of requirement analysis. In this phase the core task is to conquer a significant and transferable understanding of the portion of the world where to introduce the software. This knowledge influences subsequent design decisions. It is frequent that agents' behavior derives from the goals and needs of stakeholders. So far, research has mainly focused on the development of methods for modeling and reasoning on goals, for optimizing their achievement and for solving possible conflicts. We think there is room for novel methods for systematically identifying goals from the domain and from the users.

Starting from the hypothesis that the use of ontology for describing the problem domain is fully used by current research [1][18][3][17], which considers ontologies a powerful mean for requirements engineering, and since several AO methodologies use ontologies for different aims, we propose a way for identifying goals from an ontological model of the problem. More specifically, our main contributions are two portions of design process (to be put early in the analysis phase) for *constructing the problem domain ontology* in a way useful for *identifying goals.* The proposed way of creating the domain ontology lets us support intentionality, it is based on FIPA ontology and above all is characterized by some recurrent structures useful for identifying goals.

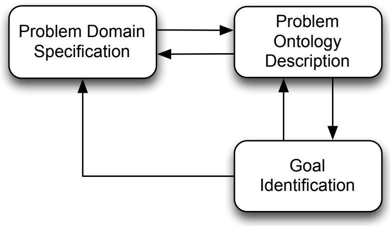


Fig. 1: The Cycle in the Early Requirements Analysis.

Generally, some advantages we found in the use of ontology are listed below:

- it is useful for formalizing knowledge and for disambiguation of terms;
- it helps in better understanding requirements and in eliminating redundancies and ambiguities;
- it simplifies comprehension among stakeholders and is useful for making clear the stakeholders' knowledge;
- it lets the designer create categories for the elements in the domain problem thus allowing to identify the artifacts that will compose the environment.

The portion of process we propose covers the initial activities of the requirement analysis for the methodological approach we are working on [7]. We named these activities respectively Problem Ontology Description and Goal Identification. The former is devoted to produce an in-depth knowledge and a formalization of the problem domain the software has to solve in terms of an ontology. The second one is devoted to produce a goal model describing the objectives of the stakeholders involved in the problem and the dependencies among them.

In the following sections we give some details about how to construct the problem domain ontology starting from a given problem statement, how to use it and how to carry out the goals identification from problem domain ontology.

III. The two proposed portions of design process

We propose an iterative approach to perform an early requirements analysis starting from textual problem specification (see Fig.1). Problem specifications are documents that could be affected by ambiguities since they are susceptible to different interpretations. Moreover, uncertainty in the interpretation of these documents could be caused by the implicit knowledge of domain experts. During knowledge elicitation, in fact, the problem is commonly described by different perspectives and by several stakeholders. This could generate overlapping terms and redundancies that could be misinterpreted. Moreover, the unconscious tendency of the stakeholder to neglect some implicit knowledge he owns may cause lacks in the problem description. The transition from a textual description to a formal and structured representation of the knowledge (i.e: the problem ontology activity) could make this gap visible. We include the problem ontology description activity in an iterative process in order to refine the domain knowledge and to disambiguate the initial documentation.

We may obtain a better understanding of the problem by deepening the domain knowledge using an ontology. Since we use the problem domain ontology for identifying goals, we expect an improvement in the goal identification. Thus obtaining a more complete list of goals to be considered during the system design. During the goal identification some inconsistencies among goals or deficiencies may also arise. These issues can be faced going back to the problem ontology trying to discover their cause (if any) and to improve the ontology. Otherwise, the origin of these issues could depend on problem specification, as it can be seen in the whole cyclic process of Fig.1.

Throughout this paper, we will use the conference management case study as defined in [20] in order to explain our approach. "*The conference management system is an open multi-agent system supporting the management of international conferences that require the coordination of several individuals and groups. During the submission phase, authors should be notified of paper receipt and given a paper submission number. After the deadline for submissions has passed, the program committee (PC) has to review the papers by either contacting referees and asking them to review a number of the papers, or reviewing them themselves. After the reviews are complete, a decision on accepting or rejecting each paper must be made. After the decisions are made, authors are notified of the decisions and are asked to produce a final version of their paper if it was accepted. Finally, all final copies are collected and printed in the conference proceedings*".

In the following of this section we introduce the proposed activities.

A. The Problem Ontology Description

The *Problem Ontology Description* (POD) activity allows to describe the problem domain elements and their relationships in a formal way. This activity grounds on an ontology used as an analysis (i.e. descriptive) model for representing the reality of problem domains typically addressed by agent oriented technologies. This ontology is described by the Problem Ontology metamodel shown in Fig.2. The Problem Ontology metamodel, we propose, has been inspired by the FIPA (Foundation for Intelligent Physical Agents) standard and ASPECS ontology. Thus, similarly, our meta ontology describes what are the elements of interest in a domain (*Concept*) with their properties (*Predicate*) and how they act in the domain (*Action*) (see the upper part of Fig.2) and it introduces some new elements in order to explicitly model intentional behaviors.

In the following we give a detailed definition of each element and relationship in the Problem Ontology metamodel. *(I)* ***Concept*** is a term usually used in a broad sense to identify "*anything about which something is said*"[4]. We use the term Concept just for representing categories of domain entities. Such categories may either be physical or logical (abstract); *(II)* ***Action*** is defined as follow: "*an action is the cause of an event by an acting concept*" (adapted from [13]). We classified actions as intentional and unintentional [10].

Intentionality implies a kind of consciousness to act, the capability to plan and enact strategies for the achievement of a purpose. Therefore, the entity that performs an Intentional Action should be able to carry out a more or less complex reasoning such as having the ability to acquire and apply knowledge. This means the entity should be endowed with some kind of intelligence. Conversely, an Unintentional Action is an automatic response governed by fixed rules or a particular set of circumstances, in other words a purely reactive action in the sense of automatics control theory. Usually, an action can have different kinds of relations with the other elements of the ontology. An action is related with (i) a concept that performs it, (ii) a concept (the target) on which the action is accomplished causing a change of this concept state, (iii) a concept (the input) required from the action to be performed even without changing the state of the input concept, (iv) a position that is notified or receives the result of the action, (v) a predicate that plays the role of precondition, i.e. a condition under which the action can be performed and (vi) a predicate that plays the role of postcondition, i.e. the condition determined by executing the action. *(III)* ***Position*** is a concept performing both Intentional Actions and Unintentional Actions. Therefore, its actions can be part of a plan to fulfill some purpose. Whilst ***Object*** represents all the concepts that can perform only unintentional actions. Positions and Objects can be both *input* and *target* of an action. *(IV)* a ***Predicate*** is the expression of a property, a state or more generally a clarification to specify a concept.

As concerns relationships, our ontology metamodel supports: *(I)* ***is-a*** that is the relationship between an ontological element and a refined version of it; *(II)* ***part-of*** that is the part-whole relationship, in which ontological elements representing the components of something are associated with the ontological element representing the entire assembly; *(III)* ***association*** that is used in order to express all types of relations different from the previous ones that can occur between two ontological elements. Usually a label is used to specify the peculiarity.

Several manual and semi-automatic methods for building an ontology exist in literature. In the following we propose some guideline for building a model of our Problem Ontology adapted from the ones proposed in literature. We suggest this one as a quick and easy guidelines to follow, but the

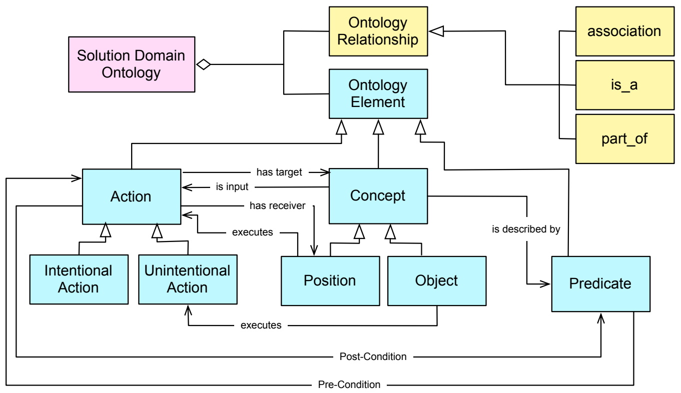


Fig. 2: The Problem Ontology Metamodel

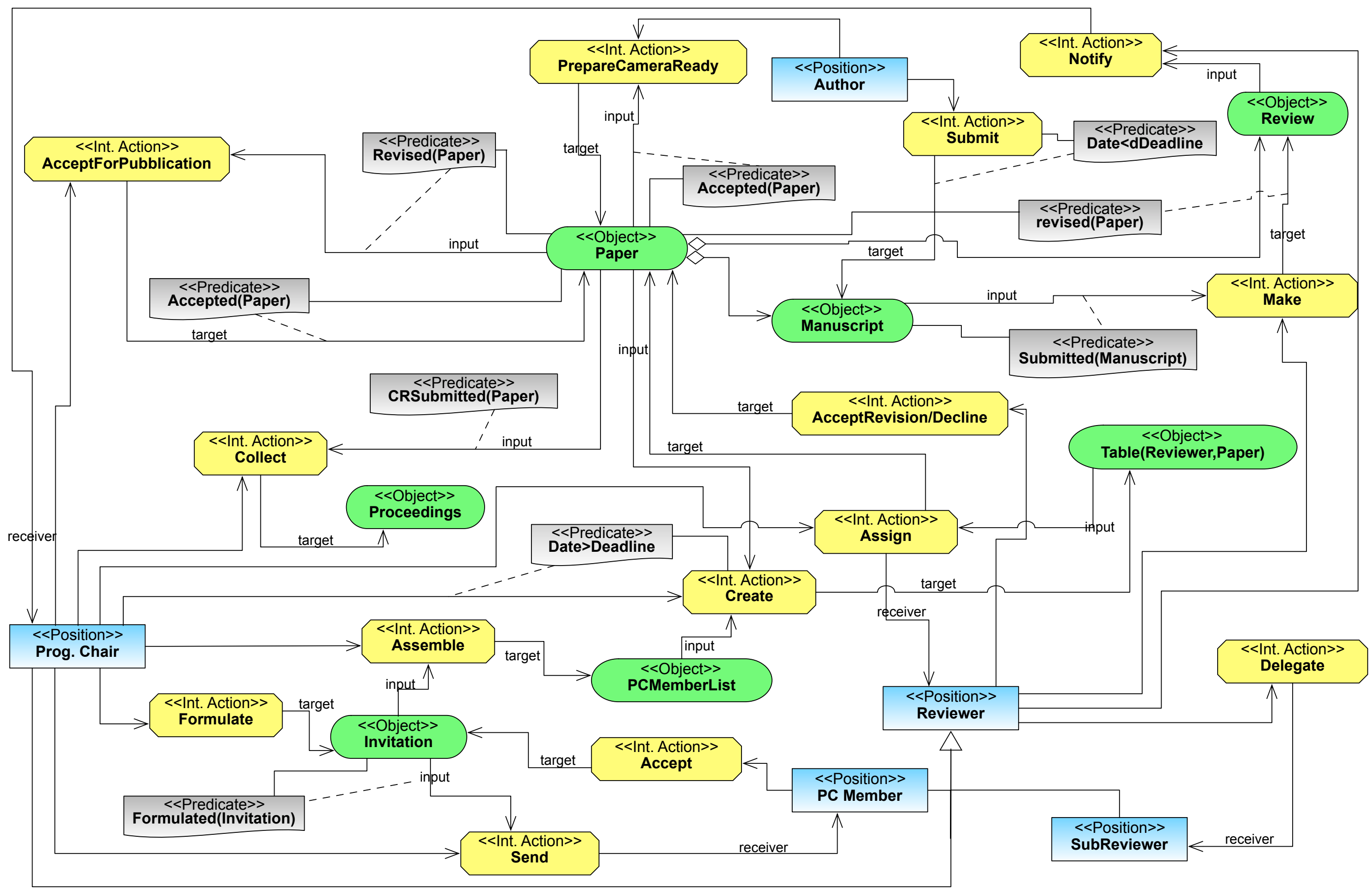


Fig. 3: The Problem Ontology Model for the Conference Management Case Study.

same results, or even better, may be achieved using different approaches.

Guidelines - With the aim of building a model of our Problem Ontology, we assume that a set of informal textual documents describe the problem the analyst is dealing with. Firstly, we extract the nouns from text and we create a list of items grouped in different clusters according to their grammatical function within the sentence. Thus, a noun used as: (i) "subject" followed by a verb is a candidate *Position* if the verb describes an *Action* on the domain; (ii) "adjective" is a candidate *Predicate* of the noun it describes; (iii) "direct object" is a candidate *Object*. Secondly, we identify verbs. In a sentence, a verb may indicate: (i) an action performed by the subject of the sentence thus becoming a candidate *Action*; (ii) a relation among nouns such as aggregation (PART-OF), inheritance (IS-A) or generic association. Finally, we identify the adjective to be candidate as a *Predicate*. Starting from these candidates we try to disambiguate the elements (if any) by asking more information to domain experts or stakeholders. Then we group in the same category the term with the same meaning. These categories will be the elements of the problem domain ontology.

Fig. 3 shows the ontology produced at the end of this activity and related to the before mentioned specification for the Conference Management case study. In this diagram the type of ontology elements is indicated by means of stereotypes and colors. Label on relationships indicates the type of ontological relationship.

B. *Goal Identification*

The problem domain ontology gives a deeper knowledge on the problem domain, especially when we perform some kind of reasoning on it. Thus moving in our context and reasoning about the elements and the relationships of our ontology, we determined many of the possible semantic combinations that could occur. We consider that the ontology is constructed by following the meta classes in Fig. 2, in so doing it presents repetitive semantic structures (hereafter *patterns*). Reasoning on these patterns, and on the meaning of the involved ontological elements, we deduced some useful information about the goals of the problem. A pattern contains many combination of elements useful for identifying at least one goal. In a pattern, at least one intentional action performed by one position is present. Intentionality implies a kind of consciousness to act, capacities to plan and enact strategies for achieving specific purposes. Consequently the intentional action may provide evidences to identify goals.

We have discovered many elementary and composed patterns. For space concerns, here we present only some of the elementary patterns we have found. We will show all possible patterns in a specific future work. The three patterns we present in this paper (see Fig.4) are elementary patterns we say atomic. These elementary structures may be surely found looking at each action and reasoning on its significance, on the position performing it and on the related input and target concepts.

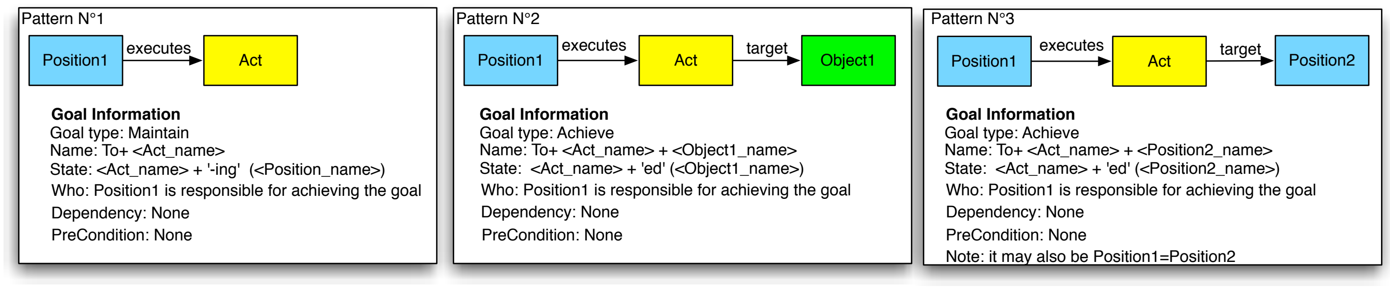

Pattern N°1
Position1
executes
Act
Goal Information
Goal type: Maintain
Name: To+ <Act_name>
State: <Act_name> + '-ing' (<Position_name>)
Who: Position1 is responsible for achieving the goal
Dependency: None
PreCondition: None
Pattern N°2
Position1
executes
Act
target
Object1
Goal Information
Goal type: Achieve
Name: To+ <Act_name> + <Object1_name>
State: <Act_name> + 'ed' (<Object1_name>)
Who: Position1 is responsible for achieving the goal
Dependency: None
PreCondition: None
Pattern N°3
Position1
executes
Act
target
Position2
Goal Information
Goal type: Achieve
Name: To+ <Act_name> + <Position2_name>
State: <Act_name> + 'ed' (<Position2_name>)
Who: Position1 is responsible for achieving the goal
Dependency: None
PreCondition: None
Note: it may also be Position1=Position2

Fig. 4: Patterns Used for Identifying Goals

Each pattern is completed by a set of information useful for completely describing the goal:

- the *Goal Type*. We identified two possible types of resulting goal: the *achieve goal* describes the achievement of a particular state of affair (e.g: win the game) while the *maintain goal* describes the maintenance of some state of affair (maintain temperature below 100 degrees);
- the *Name* of the resulting goal encapsulating the goal semantics. The name is gathered from the specific pattern (see Fig.4);
- the *State* that is the desired condition or state of the world achieved or maintained.
- *Who* is responsible for achieving the goal. It may differ from who has the real interest in achieving that state of the world;
- some kinds of *Dependency* that is a relationship between the current and another goal. It indicates that the achievement/maintenance of a goal depends on the achievement/maintenance of the other goal.

The pattern number 1 presents a Position (Position1) that executes an Action (Act). The pattern number 2 presents a Position (Position1) that executes an Action (Act) on a target Object (Object1). Finally, the pattern number 3 presents a Position (Position1) that executes an Action (Act) on another receiver or target Position (Position2). When pattern 2 occurs in a problem ontology model, we can deduce that the resulting goal type is *achieve* because the effect of the action is to change the current state of the object and to achieve a desired state by the position performing the intentional action. The name of the resulting goal is obtained as combination of the basic form of the verb representing the action along with the infinitive marker "to" and the name of the object. The state is obtained by using the past simple of verb representing the action along with the name of the object, in such a way we indicate that the state is reached. On the contrary, in the pattern 1, no object states are changed by the action so the associated goal type is *maintain* and the -ing form of the verb is used to highlight the progressive (or continuous) aspect of this state. The pattern 3 is similar to the 2, the action is made on a position instead of on an object and the information for describing the goal are the same.

These patterns are the basic configurations we may find but they may be composed with other ones and actions may have inputs or not. Dependencies can be deduced from the composition or extension of patterns and above all from the input to actions. For this reason, we do not have information on dependency for patterns of Fig.4.

The goal information we retrieve are very useful later in the design phase in order to obtain system goals, to identify their dependencies, to identify roles to be assigned to agents, to represent agent beliefs and so on.

Guidelines - In order to perform the goal identification activity, we assume that the problem ontology model has been sketched and, in each pattern, actions may have inputs. In the case there are not inputs the goal does not depend on another goal (see Fig. 4). Therefore, we accomplish the following three steps: Prepare Goals List, Describe Goals, Prepare Goals Diagram. These three steps may be done sequentially or iteratively, in the sense that the analyst may decide to initially identify the whole list of goals, then describe them etc. or (s)he may complete the description and insertion in the diagram of one goal every time.

In preparing the goals list, for each action that appears in the ontology, the analyst has to identify the smallest pattern to which it might belong. When the pattern is identified it may be described so as it is evident in the patterns (Fig. 4).
The resulting work products of these two activities are two text documents. Whereas listing goals is a trivial activity because the analyst has only to scan the ontology and to identify each couple action-position, and it is not important which is the starting element, the description of the identified goal prescribes some reasoning about the mutual position of elements in order to discover dependencies. Indeed, for each action presenting inputs, the analyst has to check if the input is a target of another action, if yes probably there is a dependency between the goals associated to that actions. The dependency is given by the semantic relationships among actions and concepts in a domain ontology. The following example in the conference management system illustrates this point.
During the goals identification some considerations may be done, they lead to reveal some inconsistencies, ambiguities, mistakes or missing elements in the domain ontology.

In order to illustrate how to identify goals, let us consider Fig. 3 and let us start from the *AcceptForPublication* action. This action is performed by the *Program Chair* and the target is the *Paper* that, after the action has been executed, is in the state *Accepted* (the final state may be underlined by the Predicate associated to the object and to the target relation whereas the precondition for action is linked to the input action). The pattern recognized is the second one where the action also presents an input. Thus, the goal may be named *To AcceptForPubblication Paper* and the final state is *Accepted-*

Goal Information		
NAME: [to Submit Manuscript]	STATE: [submitted(Manuscript)]	WHO: Author
GOAL TYPE: Achieve	PRE-CONDITION: [Date<deadline]	DEPENDENCIES: []
NAME: [to PrepareCameraReady Paper]	STATE: [PreparedCameraReady(Paper)]	WHO: Author
GOAL TYPE: Achieve	PRE-CONDITION: accepted(Paper)	DEPENDENCIES: [To AcceptForPublcation Paper]
NAME: [To Collect Proceedings]	STATE: [collected(Proceedings)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [CRSumbitted(Paper)]	DEPENDENCIES:[To PrepareCameraReady Paper]
NAME: [to Assign Reviewer]	STATE: [assigned(Reviewer)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [created (Table(Reviewer,paper))]	DEPENDENCIES: [to Create Table(Reviewer,paper)]
NAME: [to Send To PC Member]	STATE: [sentTo(PCMember)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [formulated(Invitation)]	DEPENDENCIES: [to Formulate Invitation]
NAME: [to Create Table(Reviewer,paper)]	STATE: [created (Table(Reviewer,paper))]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [collected(PCMemberList)] AND [Date>deadline]	DEPENDENCIES: [to Collect PCMemberList]
NAME: [to Collect PCMemberList]	STATE: [collected(PCMemberList)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [accepted(Invitation)]	DEPENDENCIES: [to Accept Invitation]
NAME: [to AcceptForPublication Paper]	STATE: [acceptedForPublication(Paper)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: Revised(Paper)	DEPENDENCIES: [to Make Review]
NAME: [to Formulate Invitation]	STATE: [formulated(Invitation)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES:[]
NAME: [to Accept Invitation]	STATE: [accepted(Invitation)]	WHO: PC Member
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES:[]
NAME: [to Delegate SubReviewer]	STATE: [delegated(SubReviewer)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES: []
NAME: [to Notify Prog.Chair]	STATE: [Notified(Prog.Chair)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: [made(Review)]	DEPENDENCIES: [to Make Review]
NAME: [to make Review]	STATE: [made(Review)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: [submitted(Manuscript)]	DEPENDENCIES: [to Submit Manuscript]
NAME: [to Accept Revision/Decline Paper]	STATE: [acceptedRevision(Paper)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION:[]	DEPENDENCIES: []

Fig. 5: The List of Goals for the Conference Management Case Study

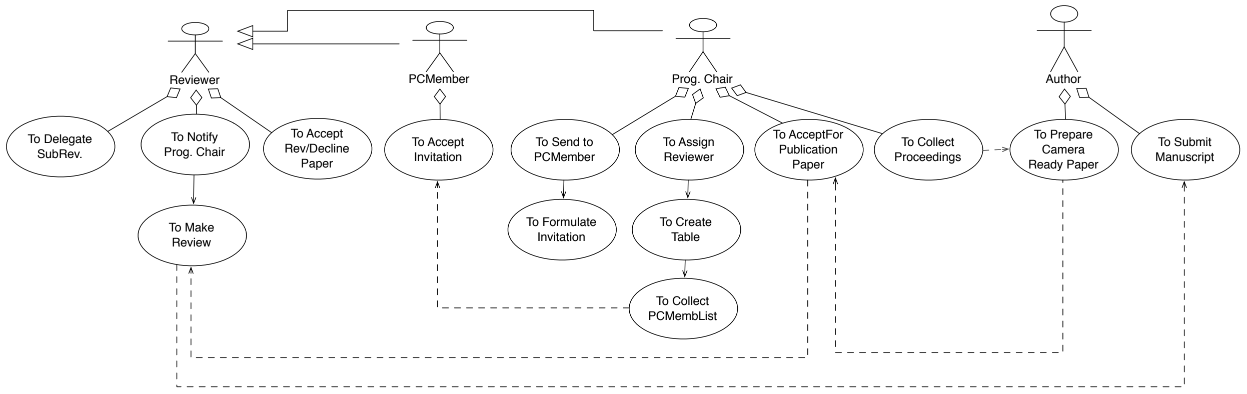


Fig. 6: The Goal Diagram Related to the Ontology in Fig. 3

ForPublication Paper. For pursuing this goal the related action has to have as input a *Paper* in the initial state *Revised*. The *Paper* is in this state after that *Reviewer* executes the *Make* action. This means that the goal *To AcceptForPubblication Paper* depends on the goal associated to the *Make* action; looking at this latter action the pattern number 2 may be identified, hence the related goal is *To Make Review* and the dependency is from *To AcceptForPubblication Paper* to *To Make Review*. Going on in this way the list of goals illustrated in Fig. 5 can be identified and the Goal Diagram in Fig.6 can be drawn. It is worth nothing that, during the first iteration of the Problem Domain Ontology description, the top part of Fig. 3 was in the form reported in Fig. 7, here we missed some predicates and we did not well represented the concept of Paper. We realized the omission while searching the *To AcceptForPubblication Paper* dependency. In fact, we have firstly realized the need for having a predicate RevisedPaper for the object Paper and then we realized that there was no actions devoted to change the state of Paper from submitted to revised. For this reasons we added the Manuscript and the Review as part-of Paper. In this way, we well represented the elements in the domain and then we were able to coherently find goal dependencies and the cycle presented in Fig. 1 was useful for identifying mistakes and omissions.

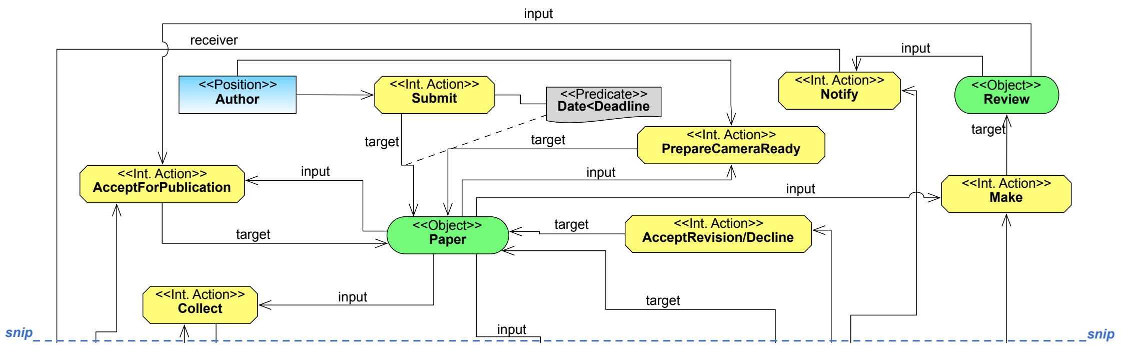


Fig. 7: A Portion of the Initial Problem Ontology Description.

IV. Discussions and Conclusions

The presented work is a part of a larger plan in order to cover the whole agent oriented requirement analysis for developing multi-agent systems for the JACAMO framework. This work addresses how to move from the problem formulation to the problem specification in order to obtain analysis models that can be fruitful employed for designing MASs. We experimented that the use of ontology in an iterative approach may allow to discover inconsistencies and lacks in the problem description and to improve the communication among stakeholders.

Moreover, the identification of goals from the ontological model of the problem domain allows the analyst to correlate goals with the corresponding portion of textual requirements. This double loop allows to act on the requirements after that the goal identification and problem ontology description activities have been performed. For instance, from goals identification it may arise a deficiency in the problem domain description that might thus be refined. Vice-versa a variation of the requirements can influence the goals that derive from them. This closed-chain process is ordered and not chaotic and it may result in a useful contribution to the quality of both the requirements elicitation and the analysis phase.

In addition, our approach allows to transform knowledge from an unstructured form to a structured one. This could result in an extra effort in the case of small size problems where it could be more easy to directly identify goals from the domain description, user interviews, etc. But the proposed approach can be an advantage in the case of large size problems. In fact, when the problem size grows up, the effort spent in managing unstructured information raises more quickly than the effort spent in applying our process.

Finally, the goal identification approach based on patterns we propose can be profitably used by an analyst with a limited expertise in goal identification and it could be automatized applying techniques of pattern recognition on problem ontology models.

References

[1] Uwe Aßmann, Steffen Zschaler, and Gerd Wagner. Ontologies, metamodels, and the model-driven paradigm. *Ontologies for Software Engineering and Software Technology*, pages 249–273, 2006.

[2] F. Bellifemine, A. Poggi, and G. Rimassa. JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, pages 97–108. Citeseer, 1999.

[3] Verónica Castañeda, Luciana Ballejos, Ma Laura Caliusco, and Ma Rosa Galli. The use of ontologies in requirements engineering. *Global Journal of Researches In Engineering*, 10(6), 2010.

[4] O. Corcho and A. Gómez-Pérez. A roadmap to ontology specification languages. *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, pages 80–96, 2000.

[5] M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies*, chapter IV, pages 79–106. Idea Group Publishing, Hershey, PA, USA, June 2005.

[6] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, 2010.

[7] M. Cossentino, C. Lodato, S. Lopes, P. Ribino, V. Seidita, and A. Chella. Towards a design process for modeling MAS organizations. In M. Cossentino, M. Kaisers, K. Tuyls, and G. Weiss, editors, *Multi-Agent Systems*, volume 7541 of *Lecture Notes in Computer Science*, pages 63–79. Springer Berlin Heidelberg, 2012.

[8] Massimo Cossentino, Antonio Chella, Carmelo Lodato, Salvatore Lopes, Patrizia Ribino, and Valeria Seidita. A notation for modeling jason-like bdi agents. *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 0:12–19, 2012.

[9] Foundation for Intelligent Physical Agents. *FIPA RDF Content Language Specification*, 2001.

[10] H.G. Frankfurt. The problem of action. *American Philosophical Quarterly*, pages 157–162, 1978.

[11] N. Gaud, S. Galland, V. Hilaire, and Koukam A. An organisational platform for holonic and multiagent systems. In *In Proceedings of Programming Multi-Agent Systems (PROMAS) workshop*, 2008.

[12] N.R. Jennings. On agent-based software engineering. In *Artificial Intelligence*, volume 117, pages 277–296, 2000.

[13] E.J. Lowe. *A survey of metaphysics*. Oxford University Press Oxford, 2002.

[14] A. Newell. The knowledge level. *Artificial Intelligence*, 1982.

[15] Anand S Rao, Michael P Georgeff, et al. Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[16] A. Ricci, M. Viroli, and A. Omicini. Carta go: A framework for prototyping artifact-based environments in mas. *Environments for Multi-Agent Systems III*, pages 67–86, 2007.

[17] M. Shibaoka, H. Kaiya, and M. Saeki. GOORE: Goal-oriented and ontology driven requirements elicitation method. In *Advances in Conceptual Modeling–Foundations and Applications*, pages 225–234. Springer, 2007.

[18] K. Siegemund, E. J Thomas, Y. Zhao, J. Pan, and U. Assmann. Towards ontology-driven requirements engineering. In *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*, 2011.

[19] Eric Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.

[20] Franco Zambonelli, Nicholas R Jennings, and Michael Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(03):303–328, 2001.

How to Improve Group Homogeneity in Online Social Networks

Pasquale De Meo, Emilio Ferrara, Domenico Rosaci, and Giuseppe M. L. Sarné

Abstract— **The formation and evolution of interest groups in Online Social Networks is driven by both the users' preferences and the choices of the groups' administrators. In this context, the notion of *homogeneity* of a social group is crucial: it accounts for determining the mutual similarity among the members of a group and it's often regarded as fundamental to determine the satisfaction of group members. In this paper we propose a group homogeneity measure that takes into account behavioral information of users, and an algorithm to optimize such a measure in a social network scenario by matching users and groups profiles. We provide an advantageous formulation of such framework by means of a fully-distributed multi-agent system. Experiments on simulated social network data clearly highlight the performance improvement brought by our approach.**

Index Terms—**Multi-agent systems, Online Social Networks, Group Recommendation, Group Homogeneity.**

I. Introduction

Online Social Networks (OSNs) such as Facebook, Google+ and Twitter have become very complex realities [6], [7], significantly grown in scale and content [5], [18], [26], with significant social effects [10], [11], [19], [30]. In this context, a relevant role is played by social *groups*, that are sub-networks of users sharing common interests [4], [21], [28], [29], [37].

Recent studies investigated the relationships between users and groups in OSNs [2], [23], [24]. For example, Hui *et al.* [23] considered four popular OSNs and empirically computed the probability that a user joins a group; the problem of choosing which group to join has been studied in [2] for a single user and in [24] for a group of users. So far, to the best of our knowledge, no study considers the evolution of a group as a problem of matching between users and groups profiles.

Although the concept of *social profile* is known in the context of virtual communities [25], that of *group profile* is rather novel. The definition of such concept is useful to face the problem of suggesting a user the groups she could affiliate to, so that to improve her satisfaction.

Commonly, a group might be considered (*i*) as a set of nodes (i.e., users) more densely connected among each other than to the others (i.e., the group formation is viewed as a graph clustering problem [13], [14], [20]); or, (*ii*) as a community of people sharing similar interests [12] (i.e., the group formation accounts for some definition of users similarity).

Satisfaction, on the other hand, is often related to the notion of group *homogeneity*: when the similarity/inter-connectivity among group participants is high, according to both structural and semantic dimensions, a OSN group is regards as homogeneous and this yields better satisfaction among its users [27].

However, if we assume that *homogeneity* should reflect users satisfaction, we argue that other behavioral characteristics of members and groups should be considered as important components [8]. For example, in virtual communities, users are often characterized by multiple interests, and groups enact common rules, define accepted behaviors, exhibit a manifold of communication styles and implement different facilities for sharing media content.

In this paper, we define a novel measure of group homogeneity that exploits users similarity and the other users' features cited above. By means of our new definition we are able to provide an algorithm to match the individual users' profiles with group profiles. The goal of this method is to find the matching between users and groups capable of improving the homogeneity of the social groups. More in detail:

- We introduce the notion of *group profile* in the context of OSNs considering a set of categories of interests, common rules, behaviors, communication styles and facilities for sharing media content. This definition of group profile is coherent with the definition of a *user profile* containing information comparable with those of a group profile.
- Each OSN group is associated with a *group agent* [15]–[17], capable of creating, managing and updating the group profile defined above. Similarly, a *user agent* is associated with each OSN user.
- We present a distributed agent platform to handle group formation [31], [32], [34]–[36]. The agents automatically and dynamically compute a matching between user and group profiles in a distributed fashion. We provide the user agent with a matching algorithm, named *Group Homogeneity Maximization* (GHM), and introduce a homogeneity measure between user and group profiles able to determine the group profiles best matching user ones.
- The GHM algorithm will be executed to improve the intra-group homogeneity as follows: (*i*) the user agent submits some requests for joining with the best groups; (*ii*) each group agent accepts only those requests whose originators have profiles matching with the group profile.
- The experimental evaluation of our matching algorithm, carried out on a set of simulated users and groups, clearly shows the advantages of our proposal.

P. De Meo is with the Dept. of Ancient and Modern Civilizations, University of Messina, 98166 Messina, Italy, e-mail: pdemeo@unime.it

E. Ferrara is with the Center for Complex Networks and Systems Research, School of Informatics and Computing, Indiana University, Bloomington (IN), USA, e-mail: ferrarae@indiana.edu

D. Rosaci is with the Dept. DIEES, University of Reggio Calabria, Loc. Feo di Vito, 89122 Reggio Calabria, Italy, e-mail: domenico.rosaci@unirc.it

G.M.L. Sarné is with the Dept. DICEAM, University of Reggio Calabria, Loc. Feo di Vito, 89122 Reggio Calabria, Italy, e-mail: sarne@unirc.it

II. The Reference Scenario

In our scenario, we consider an OSN, the set of its *users*, and the set of its *groups*, denoted by $\mathcal{S}, \mathcal{U}$ and $\mathcal{G}$, respectively. In $\mathcal{S}$, each group of users $g \in \mathcal{G}$ represents a subset of $\mathcal{U}$ (i.e, $g \subseteq \mathcal{U}\ \forall g \in \mathcal{G}$). A multi-agent system is associated with $\mathcal{S}$, such that: (*i*) each user u is supported by her personal agent a_u in the activities of participation to groups; and, (*ii*) each group g is supported by an administrator agent a_g managing all the received requests to join with the group.

A. The agents knowledge

To represent the knowledge that each agent a_u (resp., a_g) has about the orientations of its user u (resp., group g), a *profile* p_u (resp., p_g) is associated with it. This profile stores *preference* and *behavioral* information referred to the user u (resp., the users of g) in four section (called *interests*, *access preference*, *behaviors* and *friends*) storing data on topics of interest, mode to access groups, ways of performing activities and friends, respectively. The profile of a user u (resp., a group g) is represented by a 4-tuple $\langle I_u, A_u, B_u, F_u\rangle$ (resp., $\langle I_g, A_g, B_g, F_g\rangle$), where each component describes the properties of u (resp., g).

Let C be the set of all categories considered in the OSN, where each element $c \in C$ is an identifier representing a given category (e.g. *music*, *sport*, etc.). Each OSN user u (resp., group g) deals with some categories belonging to C where I_u (resp., I_g) denotes a mapping that, for each category $c \in C$, returns a real value $I_u(c)$ (resp., $I_g(c)$), ranging in $[0..1]$. This represents the level of interest of the user u (resp., the users of the group g) with respect to discussions and multimedia content dealing with c. The values of this mapping are computed based on the actual behavior of u (resp., of the users of g) — see Section II-B for the details.

The access mode property represents the policy regulating the access to a group (described by an identifier, e.g. *open*, *closed*, *secret*, etc.) preferred by u (set by the administrator of the group g) and denoted by A_u (resp., A_g).

The property B_u represents the types of behavior adopted (resp., required) by u in her OSN activities, for instance "publishing posts shorter than 500 characters". Let $b \in \mathcal{B}$ a behavior adoptable by user u (admitted in the group g) and described by a boolean variable set to $true$ if b is adopted (resp., tolerated) or $false$ otherwise and let $\mathcal{B}$ be the set of possible behaviors associated with the OSN (e.g., $B = \{b_1, b_2, \cdots, b_n\}$). Therefore, let B_u (resp., B_g) be a mapping that, for each $b \in \mathcal{B}$, returns a boolean value $B_u(b)$ (resp., $B_g(b)$), where $B_u(b_i) = true$ means that such behavior is adopted by u (resp., tolerated in g).

The property F_u (resp., F_g) represents the set of all users that are friends of u (resp., that at least have a friend among the members belonging to the group g).

B. The agents tasks

The agent a_u (resp., a_g) automatically updates the profile p_u (resp., p_g) of its user u (resp., group g) after that u (resp., a user affiliated to g) performs an action involving an information stored in its profile. In particular, every time u deals with a category c, the associated value $I_u(c)$ is updated as the weighted mean between its previous value and the new contribution to $I_u(c) = \alpha \cdot I_u(c) + (1-\alpha) \cdot \delta$. In detail, α and δ are real values arbitrarily set by u in [0..1], where δ is the increment to give to the u's interest in c due to her action, while α weights the two components of $I_u(c)$. Similarly, every time the $I_u(c)$ value of any user $u \in g$ changes, the $I_g(c)$ value of a group g is updated by the agent a_g as the mean of all the $I_u(c)$ values $\forall c \in g$. For each action performed by the user u (e.g. publishing a post, etc.) its agent a_u sets the appropriate boolean values of the variables in B_u. Analogously, the agent a_g updates the variables contained in B_g every time the administrator of g changes the associated rules. Besides, when u (resp., the administrator of g) modifies her preferences about the access mode, the associated agent updates A_u (resp., A_g). Also, when u (resp., a user of g) modifies her friends list, the associated agent updates F_u (resp., F_g). Note that a_g computes F_g as the union of the sets F_u of all the users of g.

Periodically, the agent a_u (resp., a_g) executes the user (resp., group) agent task described above, to contribute to the *group matching* activity of the OSN.

To perform the above tasks, the agents can reciprocally interact, send and receive messages thanks to a *Directory Facilitator* agent (DF), associated with the OSN, that provides a indexing service. The DF stores the names of each user and group belonging to the OSN and those of their agents. Note that the DF is the only centralized component in the proposed scenario, while the the GHM matching algorithm is completely distributed on the whole agent network.

C. Definition of homogeneity

In order to represent the potential attitude of the user u to stay in the same group with the user v (resp., to stay in the group g), we define the *homogeneity* between two users u and v (resp., a user u and a group g) as a measure representing how much u and v (resp., u and g) are similar (or, different) with respect to the properties I, A, B and F.

The homogeneity $h_{u,v}$ between the users' profiles of u and v is defined as a weighted mean of the contributions c_I, c_A, c_B and c_F, associated with the properties I, A, B and F, measuring how much the values of each property in p_u and p_v are similar. To this purpose:

- c_I is the average of the differences (in the absolute value) of the interests values of u and v for all the categories present in the social network, that is $c_I = \sum_{c\in C} |I_u(c) - I_v(c)|/|C|$.
- c_A is set to 0 or 1 if A_u is equal or not equal to A_v.
- c_B is the average of all the differences between the boolean variables stored in B_u and B_v, where this difference is set to 0 or 1 if the two corresponding variables are equal or different.
- c_F is computed as the percentage of common friends of u and v, with respect to the total number of friends of u or v as $c_F = |F_u \bigcap F_v|/|F_u \bigcup F_v|$. Note that, to make them comparable, the contributions are normalized in $[0..1]$.

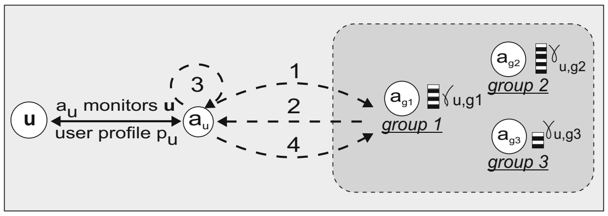


Fig. 1. User agent task schema.

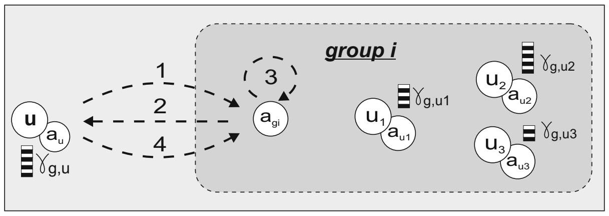


Fig. 2. The group agent task schema.

The homogeneity $h_{u,v}$ is then computed as

$$h_{u,v} = \frac{w_I \cdot c_I + w_A \cdot c_A + w_B \cdot c_B + w_F \cdot c_F}{w_I + w_A + w_B + w_F} \quad (1)$$

Similarly, homogeneity $h_{u,g}$ between a user u and a group g is simply computed as $h_{u,v}$ substituting user v with group g.

III. The GHM Algorithm

The GHM algorithm is a global activity distributed and periodically executed by each user agent a_u (resp., group agent a_g), where we call *epoch* every time the task is executed and T the (constant) period between two consecutive epochs.

A. The user agent task

Let X be the set of the n groups u is affiliated to, where $n \leq n_{MAX}$ and n_{MAX} is the maximum number of groups a user can join with. We suppose that a_u stores into a cache the profile p_g of each group $g \in X$, contacted in the past, with the date $date_g$ of its acquisition. Let m be the number of group agents that at each epoch is contacted by a_u. In such a context, a_u behaves as follows (see Figure 1):

- From the DF repository, a_u randomly selects a set Y of m groups so that $X \bigcap Y = \{0\}$ and let $Z = X \bigcup Y$ the set consisting of all the groups present in X or in Y.
- For each group $g \in Y \bigcap X$ such that $date_g > \psi$ (i.e., a fixed threshold), u sends a message to the agent a_g to ask the profile p_g associated with g (*cf.* Action 1, Fig. 2).
- For each received p_g (*cf.* Action 2, Fig. 1), u computes the homogeneity measure $h_{u,g}$ between her profile and that of the group g (*cf.* Action 3, Fig. 1).
- The groups belonging to Z and having the highest homogeneity values such that $h_{u,g} > \tau$, where τ is a real value ranging in $[0..1]$, are inserted by a_u in the set of good candidates, named $GOOD$, to join with (up to a maximum of n_{MAX} groups). For each group $g \in GOOD$ if $g \notin X$, a_u sends a join request and the profile p_u of u to a_g (*cf.* Action 4, Fig. 1). Otherwise, if $g \in X$ but $g \notin GOOD$, then a_u deletes u from g.

B. The group agent task

Let K be the set of the k users affiliated to the group g, where $k \leq k_{MAX}$, being k_{MAX} the maximum number of members allowed by the administrator of g. Suppose that a_g stores into its cache the profiles of the users $u \in K$ obtained in the past along with the date $date_u$ of their acquisition. When a_g receives a join request by a user agent u (along with u's profile p_u), it behaves as follows (see Fig. 2):

- For each user $u \in K$ such that $date_u > \eta$ (i.e., a fixed threshold), it sends a message to the agent a_u to require the profile p_u associated with u (*cf* Action 1, of Fig. 2).
- When a_g receives the required users' profiles (*cf.* Action 2, Fig. 2), it computes the homogeneity measure $h_{g,u}$ between the profile of each user $u \in K \bigcup \{r\}$ and the profile of the group g (*cf.* Action 3, Fig. 2).
- The user u having the highest homogeneity values such that $h_{g,u} > \pi$, where π is a real value ranging in $[0..1]$, is inserted by a_g in the set of good candidates, named $GOOD$, to join with (up to a maximum of k_{MAX} users). If $u \in GOOD$, a_g accepts its request to join with g (*cf.* Action 4, Fig. 2). Moreover, if $u \in K$ but $u \notin GOOD$, a_g deletes u from g.

IV. Evaluation

We evaluate the effectiveness of the GHM algorithm in increasing the homogeneity of the groups of an OSN by using a simulator, called GHM-Sim, capable of modeling all the required users and groups activities. The experiments involve a simulated OSN having 30.000 users and 100 groups, ad hoc generated by GHM-Sim, each one provided with a profile, having the structure described in Section II. More in detail, the profile p_u of a user u is generated as follows:

- The values of $I_u(c)$ are randomly chosen from a uniform distribution in the interval $[0..1]$;
- A_u is assigned the value *open* (resp., *closed* and *secret*) with a probability of 0.7 (resp., 0.2, 0.1) to implement the variability of OSNs group access restrictions;
- B_u contains the values, randomly generated, of six boolean variables representing in average the user's attitude to: (*i*) publish more than 1 post per day; (*ii*) publish posts longer than 200 characters; (*iii*) comment at least two posts of other users per day; (*iv*) respond to comments associated with her posts; (*v*) leave at least 2 "Like" rates per day; (*vi*) respond to the messages.
- The set of friends F_u are randomly generated choosing in the set of the users.

Users are initially randomly assigned to at least 2 and at most 15 of the available groups. The properties I_g, A_g, B_g and F_g of the profile p_g of each group g are randomly generated. The values of the parameters introduced in Section III are shown in Table I. We also limit to: (*i*) 250 the users who can join a given group; (*ii*) 15 the groups that a user can be joined with; (*iii*) 5 the maximum number of requests that a user can send in each epoch to new groups.

TABLE I
THE VALUES OF THE PARAMETERS USED IN THE GHM-SIM SIMULATOR.

τ	π	$KMAX$	$NMAX$	$NREQ$
0.4	0.4	250	15	5

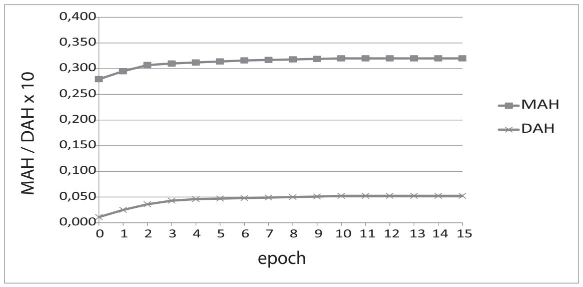


Fig. 3. Variation of MAH and DAH (x10) vs epochs obtained with the GHM-comp and GHM-diff algorithms, for a SN with 30.000 users and 100 groups.

To measure the internal *homogeneity* of a group g we use the *average homogeneity* AH_g, derived by [33], computed as $\sum_{x,y \in g, x \neq y} h_{x,y}/|g|$, while to measure the global homogeneity of the OSN groups we compute the *mean average homogeneity* MAH and the *standard deviation average homogeneity* DAH of all the AH_g, defined as

$$MAH = \frac{\sum_{g \in G} AH_g}{|G|} \qquad (2)$$

$$DAH = \sqrt{\frac{\sum_{g \in G}(AH_g - MAH)^2}{|G|}} \qquad (3)$$

In the simulations, the initial values for the above measures were $MAH = 0.266$ and $DAH = 0.0011$, denoting a very low homogeneity, due to the random generation. Applying the GHM algorithm we have simulated 15 epochs of execution per user. We can observe that the GHM algorithm quickly converges after few iterations (see Figure 3). The experimental results show that the GHM algorithm increases the homogeneity in OSN groups of about 14 percent on average, with respect to a random assignment of users to groups, achieving a stable configuration (e.g., $MAH = 0.320$ and $DAH = 0.0052$) after about 10 epochs. It is reasonable to suppose that the GHM algorithm, when applied to real OSNs, should lead to concrete benefits in terms of homogeneity.

V. RELATED WORK

In this section we describe some recent research results achieved in the fields covered by this paper, illustrating the main novelties brought in by our approach.

In the latest years, an increasing number of authors focused on the problem of recommending items to the member of a group [1], [22]. This implies the need to construct a *group profile*, often by simply aggregating the individual orientations of its members. This task is usually called *group modelling*.

More formally, let $\mathcal{U}$, $\mathcal{I}$ and $G \subseteq \mathcal{U}$ be the user population, a collection of items and a group of users, respectively. Suppose that a *rating function* $r : \mathcal{U} \times \mathcal{I} \rightarrow R$ is available, where R (*rating space*) is a discrete set. The function r receives a user $u_i \in \mathcal{U}$ and an item $i_k \in \mathcal{I}$ as input and returns an element $r_{ik} \in R$ as output. Building the profile of G is equivalent to compute a function $f_G : \mathcal{I} \rightarrow R$ receiving an item i_k as input and returns how much the members of G are satisfied by i_k.

To compute $f_G(\,)$ two popular strategies are: (*i*) *Average* [1], where $f_G(i_k)$ is equal to the average of the ratings the member of G have given to i_k. If none of the users in G has rated in i_k, then $f_G(i_k)$ is set equal to $\perp$ (this symbol specifying a not rated item)); (*ii*) *Least Misery* [3], where the rating that group G would assign to i_k is defined as $f_G(i_k) = \min r(u_i, i_k)$ if $\exists u_i \in \mathcal{U} : r(u_i, i_k) \neq \perp$ and $\perp$ otherwise.

In the *Average* strategy the score of an item i_k depends on how many users in G liked it and, if $f_G(i_k)$ is large, i_k could be recommended also to whom in G dislikes it. Otherwise, with *Least Misery* the opinion of who liked the less i_k has the biggest weight in computing $f_G(i_k)$ to minimize the chance that i_k is recommended to someone in G who dislikes it.

For example, if all of the group members *but one* like i_k and the *Least Misery* strategy is applied, i_k will automatically get a low score although almost all users in G are interested in it. Differently, in the *Average* strategy few low ratings on i_k are largely compensated by the ratings of other users.

Besides, most approaches assume that user's preferences are independent of users joining (or not) with a group: if a user alone likes (or dislikes) an item, she will continue liking (or disliking) it if she decides to join a group.

In the literature there are few papers dealing with the matching of a user and a group profile. Most of this work has been designed to recommend to an OSN user groups to join with (such a problem is also called *affiliation recommendation* in [39]). This differs from the group recommendation problem where the objects to recommend are items whereas the affiliation recommendation problem deals with groups.

Spertus *et al.* [38] presented a proposal that describes an empirical comparison of six distinct measures for computing the similarity of a user and a community to exploit for communities recommendation. Chen *et al.* [9] provide an algorithm called CCF (Combinational Collaborative Filtering) which is able to suggest users new friendship relationships as well as the communities they could join with. CCF considers a community from two different but related perspectives (e.g., users and interests) to alleviate the data sparsity arising when only information about users (resp., on words) is used.

Vasuki *et al.* [39] studied the co-evolution of the user's social network of relationships with the affiliation network modelling the affiliation of users to groups. The authors show how such information can be a good predictor to recommend to a user the groups she should join in the future.

Summarizing the benefits provided by our approach are: (*i*) to models user interests, behaviors, friendship relationships and the policies for accessing groups; (*ii*) to manage both group and user profiles by means of a multi-agent architecture where agents provide all the required affiliation activities; (*iii*) to provide a distributed greedy algorithm to match users and groups that computes, at each stage, how good a group is for a given user and selects, uniformly at random, some of these groups; (*iv*) to manage large networks with a large number of groups in a flexible and computationally feasible manner.

VI. Conclusions

The problem of dynamically increasing the intra-group homogeneity is emerging as a key issue in the OSN research field. The introduction of high-structured user profiles, the large dimensions of current OSNs and the increasing number of groups require to face efficiency and scalability issues. In this paper, we presented the *Group Homogeneity Maximization* algorithm that allows a set of software agents, associated with the OSN user profiles, to dynamically and autonomously manage the evolution of the groups, detecting for each user the best groups to join with based on the measures of homogeneity. The agents associated with the group administrators accept only those users having a profile compatible with that of the group. Our experiments on simulated social network data clearly show that the execution of the matching algorithm increases the internal homogeneity of the groups composing the social network, bringing about 15% of improvement with respect to the baseline.

In order to obtain more accurate results, in our ongoing research we are considering to combine the homogeneity measure with a new measure taking into account the trustworthiness of the users. Indeed, in virtual communities, interacting users reciprocally measure the trustworthiness of their counterparts to decide if these are reliable interlocutors or not. To this aim, we are planning a specific experimental session on real OSN data to evaluate our approaches.

References

[1] S. Amer-Yahia, S. Roy, A. Chawlat, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *Proc. of VLDB Endowment*, 2(1):754–765, 2009.

[2] E. Baatarjav, S. Phithakkitnukoon, and R. Dantu. Group recommendation system for Facebook. In *On the Move to Meaningful Internet Systems: OTM 2008 Work.*, pages 211–219. Springer, 2008.

[3] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proc. of ACM Conf. on Recommender Systems 2010*, pages 119–126. ACM Press, 2010.

[4] F. Buccafurri, D. Rosaci, G. M. L. Sarné, and L. Palopoli. Modeling cooperation in multi-agent communities. *Cognitive Systems Research*, 5(3):171–190, 2004.

[5] E. Ferrara, and G. Fiumara. Topological features of online social networks. *Comm. in Appl. and Ind. Math.*, 2(2):1–20. 2011.

[6] S. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Crawling Facebook for social network analysis purposes. In *Proc. of Int. Conf. on Web Intelligence, Mining and Semantics*. ACM, 2011.

[7] S. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Extraction and analysis of Facebook friendship relations. In Abraham, A., ed.: Computational Social Networks. Springer, pages 291-324, 2012

[8] D. Centola. The spread of behavior in an online social network experiment. *Science*, 329:1194–1197, 2010.

[9] W. Chen, D. Zhang, and E. Chang. Combinational collaborative filtering for personalized community recommendation. In *Proc. of ACM Conf. on Knowledge Discovery and Data mining*, pages 115–123. ACM, 2008.

[10] M. D. Conover, C. Davis, E. Ferrara, K. McKelvey, F. Menczer, and A. Flammini. The geospatial characteristics of a social movement communication network. *PloS one*, 8(3):e55957, 2013.

[11] M. D. Conover, E. Ferrara, F. Menczer, and A. Flammini. The digital evolution of Occupy Wall Street. *PloS one*, 8(5):e64679, 2013.

[12] S. Currarini, M. O. Jackson, and P. Pin. An economic model of friendship: Homophily, minorities, and segregation. *Econometrica*, 77(4):1003–1045, 2009.

[13] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Enhancing community detection using a network weighting strategy. *Information Sciences*, 222:648–668, 2013.

[14] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Mixing local and global information for community detection in large networks. *Journal of Computer and System Sciences*, 2013.

[15] P. De Meo, A. Nocera, G. Quattrone, D. Rosaci, and D. Ursino. Finding reliable users and social networks in a social internetworking system. In *Proc. of 2009 Int. Database Engineering & Applications Symp.*, pages 173–181. ACM, 2009.

[16] P. De Meo, A. Nocera, D. Rosaci, and D. Ursino. Recommendation of reliable users, social networks and high-quality resources in a social internetworking system. *AI Communications*, 24(1):31–50, 2011.

[17] P. De Meo, G. Quattrone, D. Rosaci, and D. Ursino. Dependable recommendations in social internetworking. In *Web Intelligence and Intelligent Agent Technologies, 2009*, pages 519–522. IET, 2009.

[18] E. Ferrara. A large-scale community structure analysis in Facebook. *EPJ Data Science*, 1(1):1–30, 2012.

[19] E. Ferrara, O. Varol, F. Menczer, and A. Flammini. Traveling trends: Social butterflies or frequent fliers? In *Proc. of 2013 ACM Conf. on Online Social Networks (COSN'13)*, 2013.

[20] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[21] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User Profiles for Personalized Information Access. In *The Adaptive Web*, volume 4321 of *LNCS*, pages 54–89. Springer, 2007.

[22] J. Gorla, N. Lathia, S. Robertson, and J. Wang. Probabilistic group recommendation via information matching. In *Proc. of Int. World Wide Web Conf. (WWW '13)*, pages 495–504. ACM Press, 2013.

[23] P. Hui and S. Buchegger. Groupthink and peer pressure: Social influence in online social network groups. In *Proc. of Int. Conf. on Advances on Social Network Analysis and Mining*, pages 53–59. IEEE, 2009.

[24] J. Kim, H. Kim, H. Oh, and Y. Ryu. A group recommendation system for online communities. *International Journal of Information Management*, 30(3):212–219, 2010.

[25] C. A. Lampe, N. Ellison, and C. Steinfield. A familiar face (book): profile elements as signals in an online social network. In *Proc. of SIGCHI Conference*, pages 435–444. ACM, 2007.

[26] J. Lehmann, B. Gonçalves, J. Ramasco, and C. Cattuto. Dynamical classes of collective attention in Twitter. In *Proc. of the 21st International Conference on World Wide Web*, pages 251–260, 2012.

[27] K. Lewis, M. Gonzalez, and J. Kaufman. Social selection and peer influence in an online social network. *Proc. of National Academy of Sciences*, 109(1):68–72, 2012.

[28] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. L. Sarné, "A Distributed Agent-Based Approach for Supporting Group Formation in P2P e-Learning," in *Advances in Artificial Intelligence*, ser. LNCS. Springer, 2013, vol. 8249, pp. 312–323.

[29] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. L. Sarné, "Hyson: A distributed agent-based protocol for group formation in online social networks," in *Multiagent System Technologies*, ser. LNCS. Springer, 2013, vol. 8076, pp. 320–333.

[30] P. T. Metaxas and E. Mustafaraj. Social media and the elections. *Science*, 338(6106):472–473, 2012.

[31] L. Palopoli, D. Rosaci, and G. M. L. Sarné. Introducing Specialization in e-Commerce Recommender Systems. *Concurrent Engineering: Research and Applications*, 21(3):187–196, 2013.

[32] L. Palopoli, D. Rosaci, and G. M. L. Sarné. A multi-tiered recommender system architecture for supporting e-commerce. In *Intelligent Distributed Computing VI*, pages 71–81. Springer, 2013.

[33] R. Pearson, T. Zylkin, J. Schwaber, and G. Gonye. Quantitative evaluation of clustering results using computational negative controls. In *Proc. of SIAM Int. Conf. on Data Mining*, pages 188–199, 2004.

[34] D. Rosaci and G. M. L. Sarné. Efficient personalization of e-learning activities using a multi-device decentralized recommender system. *Computational Intelligence*, 26(2):121–141, 2010.

[35] D. Rosaci and G. M. L. Sarné. A multi-agent recommender system for supporting device adaptivity in e-commerce. *Journal of Intelligent Information Systems*, 38(2):393–418, 2012.

[36] D. Rosaci and G. M. L. Sarné. Recommending multimedia Web services in a multi-device environment. *Inf. Sys.*, 38(2):198–212, 2013.

[37] D. Rosaci and G.M.L. Sarné. Matching users with groups in social networks. In F. Zavoral, J.J. Jung, and C. Badica, editors, *Intelligent Distributed Computing VII*, volume 511 of *Studies in Computational Intelligence*, pages 45–54. Springer, 2014.

[38] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proc. of ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 678–684. ACM, 2005.

[39] V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. Dhillon. Scalable affiliation recommendation using auxiliary networks. *ACM Transactions on Intelligent Systems and Technology*, 3(1):3, 2011.

Enhance Polarity Classification on Social Media through Sentiment-based Feature Expansion

Federico Alberto Pozzi
University of Milano-Bicocca
Viale Sarca, 336 - 20126
Milan, Italy
federico.pozzi@disco.unimib.it

Elisabetta Fersini
University of Milano-Bicocca
Viale Sarca, 336 - 20126
Milan, Italy
fersini@disco.unimib.it

Daniele Blanc
University of Milano-Bicocca
Viale Sarca, 336 - 20126
Milan, Italy
d.blanc@campus.unimib.it

Enza Messina
University of Milano-Bicocca
Viale Sarca, 336 - 20126
Milan, Italy
messina@disco.unimib.it

***Abstract*—Online social networking communities usually exhibit complex collective behaviors. Since emotions play a relevant role in human decision making, understanding how online networks drive human mood states become a task of considerable interest. One of the most relevant task in Sentiment Analysis is Polarity Classification, aimed at classifying the sentiment behind texts. We formulated different assumptions regarding which patterns within a message can be relevant sentiment indicators. Differently from well-formed texts, messages on social networks contain emoticons which could be strong sentiment indicators. For this, the first assumption states that the occurrences of emoticons representing a certain polarity could strongly agree with the overall message polarity. We then expanded the feature space including initialisms for emphatic and onomatopoeic expressions (e.g. bleh, wow, etc.) and "stretched words" (words with a letter repeated several times to emphasize a mood), extensively used in social media messages, because they could be useful information to help in determining the sentiment. Detailed analyses have been performed in order to support our assumptions. Four Machine Learning (supervised) classifiers are applied upon the expanded feature space model. Several experiments show that the considered features lead to increments of accuracy up to 5%.**

I. Introduction

According to the definition reported in [1], sentiment *"suggests a settled opinion reflective of one's feelings"*. The aim of Sentiment Analysis (SA) is therefore to define automatic tools able to extract subjective information, such as opinions and sentiments from texts in natural language, in order to create structured and actionable knowledge to be used by either a Decision Support System or a Decision Maker [2].

Sentiment Analysis is a growing area of Natural Language Processing with research ranging from document level classification [1] to learning the polarity of sentences [3] or features/objects [4]. The most widely studied problem is SA at document level [5], in which the naive assumption is that each document expresses an overall sentiment. When this is not ensured, a lower granularity level of SA could be more useful and informative.
Given the common characteristic of posts on social media to be short (e.g. the limit imposed by Twitter - a popular microblogging social networking web site - is 140 characters per post), classifying the sentiment of posts is most similar to sentence-level Sentiment Analysis.
However, the informal, specialized and length constrained language makes SA on social media a complex task. How well the features and techniques used on more well-formed data will transfer to the social media domain is an open question.

Characteristics that distinguish social media contents from well-formed contents (e.g. movie reviews [2], blogs or microblogs [6], and news [7]) is that review-type data often consists of relatively well-formed, coherent and at least paragraph-length pieces of text. Furthermore, resources such as polarity lexicons are usually available for these domains.
However, SA on social media leads towards new and more complex scenarios. In a post, a sentiment is conveyed in one or two sentence passages, which are rather informal and usually filled with abbreviations and typos. These messages are less consistent in terms of language, and usually cover a much wider array of topics. Since 2001, several studies based on polarity classification for well-formed scenarios have been proposed [4], [8], while polarity classification on user-generated content has rapidly grown only the last few years.
For instance, Barbosa and Feng [10] explored the linguistic characteristics of how tweets are written and the meta-information of words for polarity classification. In the study of Davidov et al. [11] four different feature types (punctuation, words, n-grams and patterns) are used for polarity classification and the contribution of each feature type evaluated for this task. Celikyilmaz et al. [12] proposed a new method for text normalization and investigated its effect when used for polarity classification. In particular, they used pronunciations of words to map alternative and shorter spellings into the intended words (reducing the sparseness caused by the noise in tweets).

SA is a multidisciplinary field that affects different branches of Computer Science, Social and Management Sciences. During the last years, several intersections between Sentiment Analysis and the Multi-Agent system technology are emerging [13]–[15]. For instance, Almashraee et al. [16] proposed a method that uses both multi-agent system technologies and machine learning techniques to provide a solution to the problems of polarity classification of on-line product features.They are able to extract data from several social media networks (one agent per network) and analyze sentiments using a learning mechanism for future predictions.
By using a stochastic multi-agent based approach, the system proposed by Gatti et al. [17] models and simulates user behavior on real-world social networks, taking into account what users (agents) post. Several challenges are faced: sampling the networks from the real-world social networks, performing text classication (Natural Language Processing) to predict topic

and sentiment from posts, modeling the user behavior to predict his/her actions (pattern recognition), and large-scale simulation.

In this work, we propose different approaches for text normalization and feature expansion to improve classification performance: after that messages are analyzed and normalized/preprocessed, different additional features (initialisms for emphatic and onomatopoeic expressions, emoticons, adjectives and "stretched words"[1]) are integrated within the bag-of-words model and used by common Machine Learning (supervised) classifiers. Further details are reported in Sect. II.

Although adaptation of this framework to other microblogs is straightforward, experiments are addressed on Twitter (tweets are short status updates of 140 characters or less) since it is an increasingly popular platform able to convey opinions and thoughts. Polarity classification approaches based on Twitter could provide unprecedented utility for different parties (e.g. marketing and financial purposes). For instance, an industry could gauge its recent marketing campaign by aggregating user opinions regarding their products. Moreover, it might be possible to identify the sentiment of financial news to forecast returns of markets [18] or sound out public opinion during political campaigns [19].

II. System architecture

We propose a system that is composed of three main modules: the first deals with preprocessing techniques, such as Text Normalization and Spelling Corrections, the second with Feature Expansion and the last with supervised classification techniques. Data are stored in a database to be subsequently easily reused and plotted. The system architecture is reported in Fig. 1.

A. Preprocessing module

Since tweets are similar to SMS messages, the writing style and the lexicon is widely varied. Moreover, tweets are often highly ungrammatical, and filled with spelling errors.

a) Text Normalization: In order to clean the dataset, we captured a set of patterns which are detected using dictionaries a priori defined and regular expressions. The applied filters are:

- **URLs**: All tokens matching the REGEXP `(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]` are transformed in its form without punctuation to avoid URL segmentation during tokenization (e.g. http://www.mind.disco.unimib.it becomes httpwwwminddiscounimibit);
- **Hashtags**: The symbol # is removed from all the tokens;
- **Mention Tags**: The tokens corresponding to a mention tag, identified through the REGEXP @(.+?), are removed;
- **Retweet Symbols**: All the tokens matching the expression `RT @(.+?):` are removed.

[1]Words that have a letter repeated several times to emphasize a mood, e.g. "I'm so happyyyyyyyyyy!"

Note that the adaptation and modifications of the REGEXPs adopted in these filters to other microblogs is straightforward.

b) Spell-Checker: In addition to filters, misspelled tokens have been corrected using the Google's Spell Checker API[2]. Since the Google's algorithm takes the neighbourhood (context) of a misspelled token into account in suggesting the correction, the whole previously filtered tweet is considered as a query rather than the single token.

B. Feature Expansion module

Once the text normalization step has been performed, some additional features have to be extracted:

- **Emoticons**: in order to detect positive, neutral and negative emoticons, three dictionaries have been defined. For instance, positive emoticons are ':-)', ':)', '=)', ':D', neutral emoticons are ':-|', ':|', '=|', ';|' and negative emoticons are ':-(', ':(', '=(', ';('.
  If a token appears in the dictionary of positive emoticons then it is replaced with `POSEXPRESSIONS`, if it appears in the dictionary of neutral emoticons it is replaced with `NEUEXPRESSIONS`, otherwise with `NEGEXPRESSIONS`;
- **Initialisms for emphatic expressions**: several emphatic expressions are used in English. For instance, expressions such as 'ROFL','LMAOL','LMAO','LMAONF' represent positive expressions. They are replaced with `POSEXPRESSIONS`, `NEUEXPRESSIONS` or `NEGEXPRESSIONS`;
- **Slang correction**: in order to aggregate terms with the same meaning but different slangs, a dictionary of a priori defined slang expressions with their meaning, such as 'btw' (by the way), 'thx' (thanks), 'any1' (anyone) and u (you) has been built;
- **Onomatopoeic expressions**: as the previous point, a mapping dictionary has been defined for onomatopoeic expressions, such as 'bleh' (`NEGEXPRESSIONS`) and 'wow' (`POSEXPRESSIONS`). Also laughs are considered as onomatopoeic expressions: if a token has a sub-pattern matching `((a|e|i|o|u)h|h(a|e|i|o|u))\\1+|(ahha|ehhe|ihhi|ohho|uhhu)+`, then the whole token is replaced with `POSEXPRESSIONS`;
- **Stretched words**: a specific procedure has been defined to detect weather a term is a stretched word or not:

```
if Term has a lengthening then
    root ← Extract term root
    correction_list ← GoogleSpellChecker (root)
    if correction_list = ∅ then
        return isStretched
    end if
else
    return isNOTStretched
end if
```

[2]https://code.google.com/p/google-api-spelling-java/

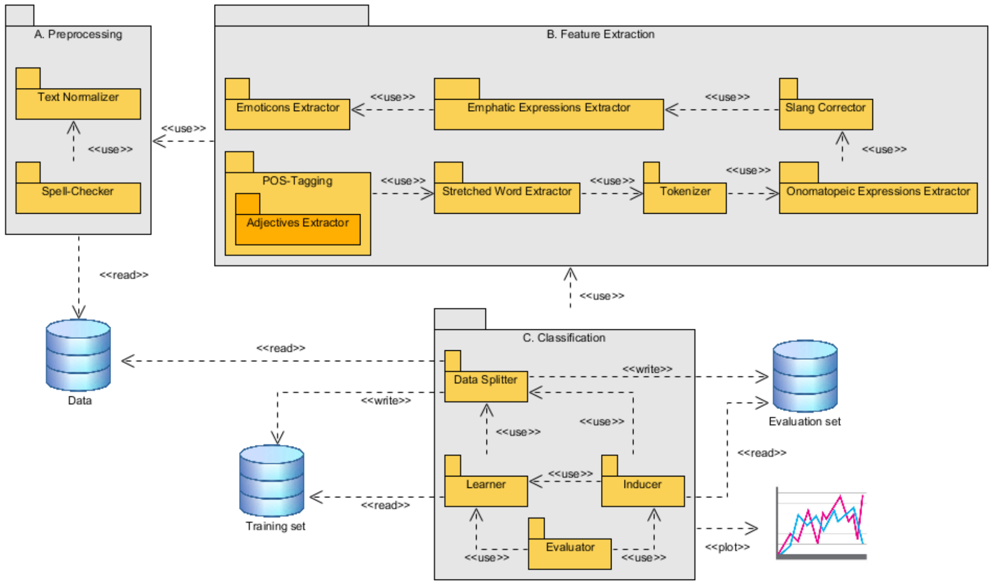


Fig. 1: System Architecture

A particular REGEXP has been defined to detect the presence of a lengthening (or *stretching*) in a term. Whether there is a match, the term root is extracted, otherwise the term is discarded and the next token is analyzed. The term root is analyzed by the Google's Spell Checker[3]. The spell checker's output is a list of possible corrections, ordered with respect to their probability. If the list is empty, the term is declared to be a stretch word.

- **Adjectives**: a Part-Of-Speech (POS) tagging process has been performed in order to tag each term with respect to its verbal form, to subsequently extract the adjectives (tagged as JJ, JJR, JJS) and determine their polarity depending on the fact that the term is in the dictionary of positive or negative terms. If the adjective is neither in the positive nor in the negative dictionary, its polarity is assumed to be neutral. The Stanford Log-linear Part-Of-Speech Tagger library[4] of Stanford University has been used for this task.

C. Classification module

Let $\vec{d} = (t_1, ..., t_n)$ a traditional feature vector composed only of terms, the new representing feature vector is defined as:

$$\begin{aligned}\vec{d}_{new} &= (t_1, ..., t_n, e_{pos}, e_{neu}, e_{neg},\\ &\quad se, adj_{pos}, adj_{neu}, adj_{neg}, strw, class)\end{aligned}$$

where $\{pos, neu, neg\} \in pol$ is the polarity, e_{pol} represents the emoticons, initialisms for emphatic and onomatopoeic expressions according to polarity pol, adj_{pol} represents the adjectives according to polarity pol, $strw$ represents the stretched words and $class$ is the ground truth polarity. According to the used term weighting method, boolean (0/1) or Term-Frequency (TF), adj_{pol} and $strw$ represents the presence or absence of the feature or how many times the feature occurs, respectively. Considering the boolean weighting schema, e_{pol} is zero if all the three atomic features involved (emoticons, initialisms for emphatic and onomatopoeic expressions) are zero and one if at least one of them exists, while considering the TF method it assumes the sum of how many times each atomic feature occurs. Experiments on the datasets have returned higher results using the 0/1 weighting schema for terms and TF for the additional external features.

The supervised classifiers used and compared in the system are: Naive Bayes (NB), K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Decision Trees (DT). Several classifier configurations have been tested (linear, polynomial and gaussian kernel for SVM and Naive Bayes Multinomial) and the most performing are used. For SVM, the linear kernel is used, while the Naive Bayes Multinomial overperforms the Naive Bayes classifier. K-NN has been tested for $k = 1, 3, 5, 10$ and the most performing value is $k = 3$. A 10-folds cross validation has been adopted as evaluation criteria. In order to obtain more statistically significant results, each experiment has been performed 10 times. The final performance are obtained by the arithmetic mean among the experiments. Classification experiments have been performed using Java. In particular, the WEKA[5] libraries have been adopted as tools for classification.

III. Datasets

In order to evaluate the proposed method, we performed our experiments on three datasets. The first is called **Gold Standard Person** [20], the second **Gold Standard Movie** [20] and the third is a concatenation of the two plus 3258 additional posts (that we called '**merged**').

Each gold standard dataset originally contains 1,500 manually labeled Twitter data for target-specific sentiment (i.e.,

[3]https://code.google.com/p/google-api-spelling-java/
[4]http://nlp.stanford.edu/software/tagger.shtml
[5]http://www.cs.waikato.ac.nz/ml/weka/

TABLE I: Precision, Recall and F-Measure per class on the Merged Dataset. The expression (v/ /*) indicates whether the values is statistically more (v), equally ( ) or less (*) significant (95% of confidence) than the baseline configuration (C).

Label	Classifier	Precision				Recall				F-Measure			
		C	C-F	NC-F	NC	**C**	C-F	NC-F	NC	**C**	C-F	NC-F	NC
POS	NB Multinomial	0,67	**0,7 v**	**0,7 v**	0,67	0,75	**0,79 v**	**0,79 v**	0,76	0,7	**0,74 v**	**0,74 v**	0,71
	SMO (PKl e=1)	0,71	**0,74 v**	0,73 v	0,71	0,69	**0,73 v**	**0,73 v**	0,69	0,7	**0,73 v**	**0,73 v**	0,7
	Ibk (k=3)	0,69	**0,74 v**	0,71	0,68	0,19	0,26 v	**0,27 v**	0,19	0,3	0,38 v	**0,39 v**	0,3
	J48	0,64	0,69 v	**0,7 v**	0,64	0,54	**0,64 v**	**0,64 v**	0,54	0,59	**0,66 v**	**0,66 v**	0,59
	(v/ /*)		(4/0/0)	(3/1/0)	(0/4/0)		(4/0/0)	(4/0/0)	(0/4/0)		(4/0/0)	(4/0/0)	(0/4/0)
NEU	NB Multinomial	0,84	0,86 v	**0,86 v**	0,84	0,88	0,88	0,88	0,88	0,86	**0,87 v**	**0,87 v**	0,86
	SMO (PK e=1)	0,8	0,81 v	**0,81 v**	0,8	**0,9**	**0,9**	**0,9**	0,89	0,85	0,85 v	0,85	0,84
	Ibk (k=3)	0,62	0,63 v	**0,64 v**	0,62	**0,98**	**0,98**	0,97 *	**0,98**	0,76	**0,77 v**	**0,77 v**	0,76
	J48	0,74	**0,76 v**	**0,76 v**	0,73	**0,88**	**0,88**	**0,88**	**0,87**	0,8	**0,82 v**	**0,82 v**	0,8
	(v/ /*)		(4/0/0)	(4/0/0)	(0/4/0)		(0/4/0)	(0/3/1)	(0/4/0)		(4/0/0)	(3/1/0)	(0/4/0)
NEG	NB Multinomial	0,76	**0,78**	0,77	0,75	0,42	**0,5 v**	**0,5 v**	0,42	0,54	**0,61 v**	**0,61 v**	0,53
	SMO (PK e=1)	0,66	**0,72 v**	0,69	0,65	0,34	**0,39 v**	0,38 v	0,35	0,45	**0,5 v**	0,49 v	0,45
	Ibk (k=3)	0	0,02	0,02	**0,06**	0	0	0	0	0	0	0	0
	J48	0,37	0,53 v	**0,55 v**	0,37	0,17	0,28 v	**0,29 v**	0,18	0,23	0,36 v	**0,38 v**	0,24
	(v/ /*)		(2/1/0)	(2/6/0)	(0/4/0)		(3/1/0)	(3/1/0)	(0/4/0)		(3/1/0)	(3/1/0)	(0/4/0)

the sentiment towards a specific target such as a movie or a person).
Each line of the data entry follows the format (id, topic, content, polarity), where 'id' is the id of the tweet, 'topic' is the name of the movie/person talked in the tweet, 'content' is tweet content and 'polarity' is the sentiment polarity about the topic expressed in the tweet, which can be 'pos' (positive), 'neg' (negative), 'neu' (neutral), or 'no sentiment' (not considered in this work). Polarity distributions for each of the studied dataset are reported in Figure 2.
We report in the following the descriptive analysis for adjectives, emoticons and stretched words. Similar statistics have been obtained (but omitted) for initialisms for emphatic and onomatopoeic expressions.

IV. Descriptive Analysis

The assumptions we formulated regard which patterns behind messages can be relevant sentiment indicators. Wilson et al. [21] shows that using emoticons for the learning phase of a classifier can lead to performance improvements. Moreover, Marchetti-Bowick and Chambers [22] present an approach that instead uses distant supervision (using emoticons as ground truth for labels) to train a classifier on a dataset of tweets, achieving higher performance in polarity classification. For this reason we argue that the occurrences of emoticons representing a certain polarity could strongly agree with the overall message polarity.
In addition to emoticons, we expect that also adjectives could be relevant sentiment indicators: in human interactions, the use of adjectives offers to the author the possibility to describe, in the best possible way, the own subjectivity within the discourse. Moghaddam and Popowich [23] demonstrated that the inclusion of adjectives as features in the bag-of-words model improves the classification accuracy.
Finally, we assume that also stretched words, extensively used in social media posts, could be useful information to help in determining the sentiment. To the best of our knowledge, no studies consider the combination of adjectives, initialisms for emphatic and onomatopoeic expressions, emoticons and stretched words as possible additional features.

In order to verify whether the proposed preprocessing techniques and generated features improve classification performance, four experiment configurations have been considered for each studied dataset (Table II).

TABLE II: Experiment configurations

Configuration	**Text Normalization**	**Feature Expansion**
Content (C)	✓	✗
PreprocessedContent (PC)	✓	✗
Content-FeatureExpansion (C-FE)	✗	✓
PC-FeatureExpansion (PC-FE)	✓	✓

A. Adjectives

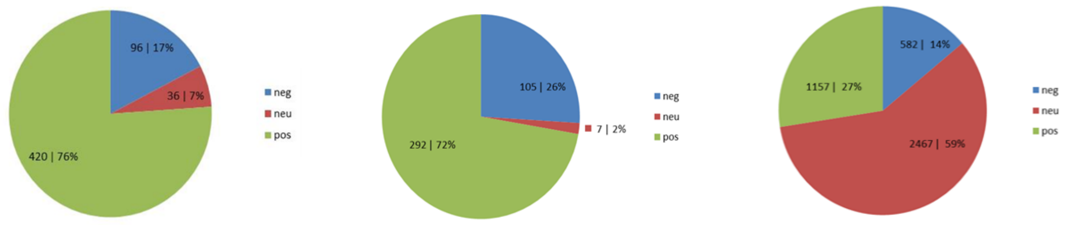
An analysis on the adjective distribution has been performed on the studied datasets (Table III).
First of all, as expected, positive and negative messages have a high percentage of adjectives. In order to verify that adjectives could be an important source of information for polarity classification, a further and detailed analysis has been conducted calculating conditional probabilities (conditioning the adjective presence to the overall message polarity) and with inverse conditional probabilities (viceversa). Conditional probabilities give us information about how much posts p classified with a contain polarity pol contain emoticons with the same polarity:

$$P(adj = pol \in p \mid p = pol) = \frac{\#(p = pol \wedge adj = pol \in p)}{\#(p = pol)}$$

where P stands for Probability.
Results (Table VI) generally show that the polarity of adjectives in messages agrees with the overall message polarity: this leads a positive message to have positive adjectives with a higher probability. Moreover, also inverse conditional probabilities have been calculated:

$$P(p = pol \mid adj = pol \in p) = \frac{\#(p = pol \wedge adj = pol \in p)}{\#(adj = pol \in p)}$$

as the ratio between the number of messages p with polarity pol that contain adjectives with polarity pol and the number of messages that contain adjectives with polarity pol.
The estimation of conditional probabilities and inverse conditional probabilities leads us to state the agreement between the adjective and message polarities.



(a) Gold Standard Movie Dataset (b) Gold Standard Person Dataset (c) Merged Dataset

Fig. 2: Polarity distribution

TABLE III: Adjective distribution

	Dataset	Tweets (%)	Polarity (%)	
With Adjectives	Movie	82% (455)	78% (353)	pos
			6% (29)	neu
			16% (73)	neg
	Person	67% (271)	76% (207)	pos
			2% (5)	neu
			22% (59)	neg
	Merged	54% (2262)	38% (850)	pos
			46% (1053)	neu
			16% (359)	neg
Without Adjectives	Movie	18% (97)	69% (67)	pos
			7% (7)	neu
			24% (23)	neg
	Person	33% (133)	64% (85)	pos
			2% (2)	neu
			35% (46)	neg
	Merged	46% (1944)	16% (307)	pos
			73% (1414)	neu
			11% (223)	neg

TABLE IV: Emoticon distribution

	Dataset	Tweets (%)	Polarity (%)	
With Emoticons	Movie	16% (88)	80% (70)	pos
			8% (7)	neu
			12% (11)	neg
	Person	7% (28)	14% (4)	pos
			0%	neu
			86% (24)	neg
	Merged	9% (381)	61% (231)	pos
			19% (74)	neu
			20% (76)	neg
Without Emoticons	Movie	86% (464)	75% (350)	pos
			6% (29)	neu
			18% (85)	neg
	Person	93% (376)	71% (268)	pos
			2% (7)	neu
			27% (24)	neg
	Merged	91% (3825)	24% (926)	pos
			63% (2393)	neu
			13% (506)	neg

B. Emoticons

Table IV shows the emoticons distribution on the three studied datasets. Positive and negative messages, as expected, have a high percentage of emoticons.
In order to verify that emoticons could be an important source of information for polarity classification (as well as adjectives), a further and detailed analysis has been conducted conditioning the emoticons presence to the message polarity and viceversa with inverse conditional probabilities. Conditional probabilities give us information about how much posts p classified with a contain polarity pol contain emoticons e with the same polarity:

$$P(e = pol \in p \mid p = pol) = \frac{\#(p = pol \wedge e = pol \in p)}{\#(p = pol)}$$

Results generally show that the polarity of emoticons in messages agrees with the message polarity: this leads a positive message to have positive emoticons with a higher probability. Moreover, as well as for adjectives, the inverse conditional probabilities have been calculated:

$$P(p = pol \mid e = pol \in p) = \frac{\#(p = pol \wedge e = pol \in p)}{\#(e = pol \in p)}$$

as the ratio between the number of messages p with polarity pol that contain emoticons e with polarity pol and the number of messages that contain emoticons with polarity pol.
Both probabilities further confirm the agreement between emoticons and message probabilities.

C. Stretched words

An further analysis has been performed on the stretched word distribution for the three studied datasets (Table V).
First of all, as expected, positive and negative messages have higher percentages of stretched words than neutral messages (even if messages which contain stretched words are very few).
In order to verify that stretched words could be an important source of information for polarity classification, a further and detailed analysis has been conducted conditioning the stretched words presence to the message polarity and viceversa with inverse conditional probabilities.
Conditional probabilities (and inverse conditional probabilities) are calculated as shown above for adjectives and emoticons. Supported from the analyzed data, we can conclude that stretched words have a high correspondence with positive and negative polarities.

TABLE V: Stretched words distribution

	Movie	Person	Merged
$P(stretch \in p)$	0,054	0,035	0,032
$P(stretch \notin p)$	0,946	0,965	0,968
$P(stretch \in p \mid p = pos)$	0,055	0,041	0,056
$P(stretch \in p \mid p = neu)$	0,028	0	0,013
$P(stretch \in p \mid p = neg)$	0,063	0,019	0,064

TABLE VI: Conditional probabilities for adjectives

(a) Movie

Positive Tweets		Neutral Tweets		Negative Tweets	
$P(adj \in p \mid p = pos)$	**0,840**	$P(adj \in p \mid p = neu)$	**0,806**	$P(adj \in p \mid p = neg)$	**0,760**
$P(adj \notin p \mid p = pos)$	0,160	$P(adj \notin p \mid p = neu)$	0,194	$P(adj \notin p \mid p = neg)$	0,240
$P(adj = pos \in p \mid p = pos)$	**0,643**	$P(adj = pos \in p \mid p = neu)$	0,278	$P(adj = pos \in p \mid p = neg)$	0,250
$P(adj = neu \in p \mid p = pos)$	0,386	$P(adj = neu \in p \mid p = neu)$	0,333	$P(adj = neu \in p \mid p = neg)$	0,375
$P(adj = neg \in p \mid p = pos)$	0,155	$P(adj = neg \in p \mid p = neu)$	**0,528**	$P(adj = neg \in p \mid p = neg)$	**0,448**

(b) Person

Positive Tweets		Neutral Tweets		Negative Tweets	
$P(adj \in p \mid p = pos)$	**0,709**	$P(adj \in p \mid p = neu)$	**0,714**	$P(adj \in p \mid p = neg)$	**0,562**
$P(adj \notin p \mid p = pos)$	0,291	$P(adj \notin p \mid p = neu)$	0,286	$P(adj \notin p \mid p = neg)$	0,438
$P(adj = pos \in p \mid p = pos)$	**0,524**	$P(adj = pos \in p \mid p = neu)$	0,143	$P(pos\, adj \in p \mid p = neg)$	0,124
$P(adj = neu \in p \mid p = pos)$	0,329	$P(adj = neu \in p \mid p = neu)$	**0,571**	$P(neu\, adj \in p \mid p = neg)$	**0,400**
$P(adj = neg \in p \mid p = pos)$	0,058	$P(adj = neg \in p \mid p = neu)$	0,143	$P(neg\, adj \in p \mid p = neg)$	0,238

(c) Merged

Positive Tweets		Neutral Tweets		Negative Tweets	
$P(adj \in p \mid p = pos)$	**0,735**	$P(adj \in p \mid p = neu)$	0,427	$P(adj \in p \mid p = neg)$	**0,617**
$P(adj \notin p \mid p = pos)$	0,265	$P(adj \notin p \mid p = neu)$	**0,573**	$P(adj \notin p \mid p = neg)$	0,383
$P(adj = pos \in p \mid p = pos)$	**0,528**	$P(adj = pos \in p \mid p = neu)$	0,120	$P(adj = pos \in p \mid p = neg)$	0,127
$P(adj = neu \in p \mid p = pos)$	0,373	$P(adj = neu \in p \mid p = neu)$	**0,327**	$P(adj = neu \in p \mid p = neg)$	**0,347**
$P(adj = neg \in p \mid p = pos)$	0,090	$P(adj = neg \in p \mid p = neu)$	0,042	$P(adj = neg \in p \mid p = neg)$	0,308

V. Results

In this section the performance achieved from the studied classifiers on the configurations of the Merged dataset are presented (Table I), since the Movie and Person datasets present few instances that could be not statistically significant (Figure 2). To this purpose, we measured Precision (P), Recall (R) and F$_1$-measure, defined as

$$P = \frac{TP}{TP+FP} \quad R = \frac{TP}{TP+FN} \quad F1 = \frac{2 \cdot P \cdot R}{P+R}$$

for the positive, neutral and negative labels. We also measured Accuracy, defined as

$$Acc = \frac{TP+TN}{TP+FP+FN+TN}$$

Figure 3 shows the final classification accuracy of each studied classifier on the Merged dataset[6]. The configurations C-FE and PC-FE lead to increments of approximately 2% of accuracy. In some cases, increments achieve 5%. Moreover, results of the two configurations are usually statistically significant (i.e. results are not randomly achieved). However, text normalization does not lead to significant improvements and results are not statistically significant. Regarding classifier performance, Multinominal Naive Bayes and SVM achieve the highest results.

VI. Conclusion

In this work, we proposed a system aimed at classify the polarity of messages on social media. We formulated different assumptions regarding what elements within a message can be relevant sentiment indicators. The first assumption states that the occurrences of emoticons representing a certain polarity could strongly agree with the overall message polarity. As well as expanding the feature space including emoticons, we assumed that also adjectives and stretched words, extensively used in social media messages, could be useful information to help in determining the sentiment. To the best of our knowledge, no studies consider the combination of adjectives, initialisms for emphatic and onomatopoeic expressions, emoticons and stretched words as possible additional features. Subsequently, detailed analyses have been performed in order to verify our assumptions. For each studied dataset, four different configurations have been considered to measure the improvements led from each component (not preprocessed content and no additional features, not preprocessed content but additional features, preprocessed content but not additional features and preprocessed content with additional features). The supervised classifiers used in the system are Naive Bayes (NB), K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Decision Trees (DT). Several experiments show that text normalization does not lead to significant improvements but expanding the feature space of the traditional bag-of-words model with the considered features lead to accuracy increments up to 5%. Regarding classifier performance, Multinominal Naive Bayes and SVM achieve the highest results.

[6]Results on the other datasets are similar.

Acknowledgment

This work has been partially supported by the I-ShErPA project.

References

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, pp. 1–135, 2008.

[2] F. A. Pozzi, E. Fersini, and E. Messina, "Bayesian model averaging and model selection for polarity classification," in *18th International Conference on Applications of Natural Language to Information Systems*, ser. LNCS. Springer Berlin Heidelberg, 2013, vol. 7934, pp. 189–200.

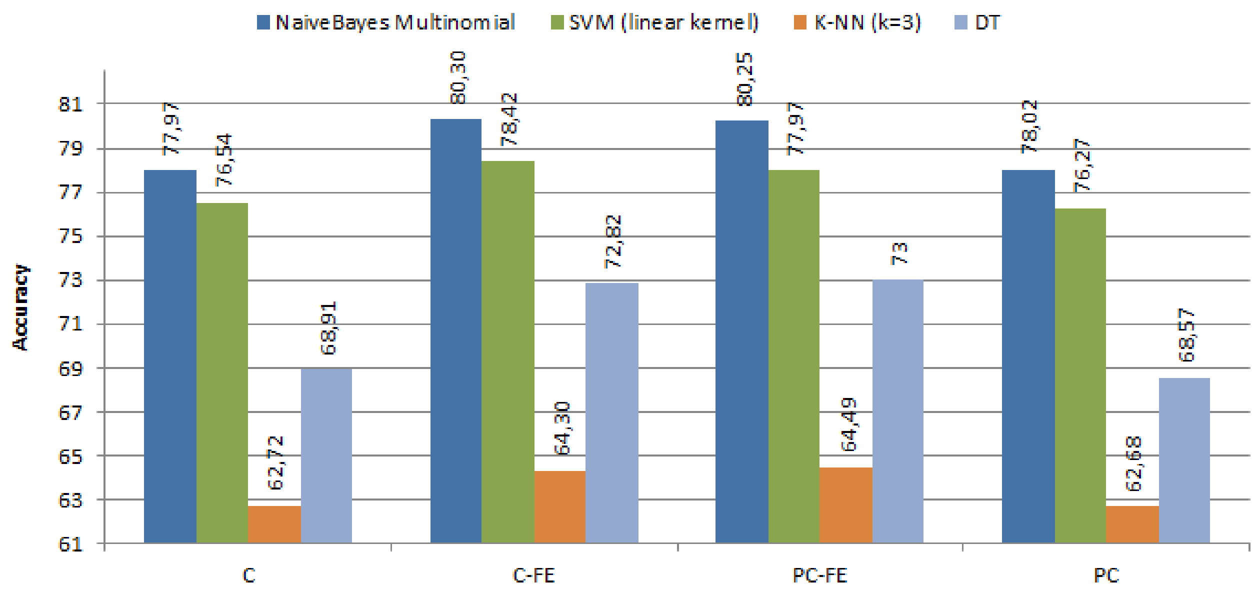


Fig. 3: Final results about the Merged dataset

[3] V. S. Jagtap and K. Pawar, "Analysis of different approaches to sentence-level sentiment classification," *International Journal of Scientific Engineering and Technology*, vol. 2, no. 3, pp. 164–170, 2013.

[4] H. Zhang, Z. Yu, M. Xu, and Y. Shi, "Feature-level sentiment analysis for chinese product reviews," in *3rd International Conference on Computer Research and Development (ICCRD)*, vol. 2, 2011, pp. 135–140.

[5] A. Yessenalina, Y. Yue, and C. Cardie, "Multi-level structured models for document-level sentiment classification," in *Proc. of the Conf. on Empirical Methods in NLP*, 2010.

[6] F. A. Pozzi, D. Maccagnola, E. Fersini, and E. Messina, "Enhance user-level sentiment analysis on microblogs with approval relations," in *AI*IA 2013*, ser. LNAI, M. B. et al., Ed. Springer International Publishing Switzerland, 2013, vol. 8249, pp. 133–144.

[7] Y. Rao, J. Lei, L. Wenyin, Q. Li, and M. Chen, "Building emotional dictionary for sentiment analysis of online news," *World Wide Web*, pp. 1–20, 2013.

[8] S. Mukherjee and P. Bhattacharyya, "Feature specific sentiment analysis for product reviews," in *13th International Conference on Intelligent Text Processing and Computational Linguistics*, ser. Lecture Notes in Computer Science, vol. 7181. Springer, 2012, pp. 475–487.

[9] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford, Technical Report, 2009.

[10] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proc. of ACL*, 2010.

[11] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10. Association for Computational Linguistics, 2010, pp. 241–249.

[12] A. Celikyilmaz, D. Hakkani-Tur, and J. Feng, "Probabilistic model-based sentiment analysis of twitter messages," in *Spoken Language Technology Workshop (SLT), IEEE*, 2010, pp. 79–84.

[13] D. Garcia and F. Schweitzer, "Emotions in product reviews-empirics and models," in *Privacy, security, risk and trust (passat), 2011 ieee third international conference on social computing (socialcom)*. IBaI Publishing, 2011, pp. 483–488.

[14] F. Schweitzer and D. Garcia, "An agent-based model of collective emotions in online communities," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 77, pp. 533–545, 2010.

[15] K. Fujimoto, "A computational account of potency differences in ewom messages involving subjective rank expressions," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, vol. 3, 2011, pp. 138–142.

[16] D. M. D. Mohammed Almashraee and R. Unland, "Sentiment classification of on-line products based on machine learning techniques and multi-agent systems technologies," in *Industrial Conference on Data Mining - Workshops*. IBaI Publishing, 2012, pp. 128–136.

[17] M. Gatti, A. P. Appel, C. Pinhanez, C. dos Santos, D. Gribel, P. Cavalin, and S. B. Neto, "Large-scale multi-agent-based modeling and simulation of microblogging-based online social network," in *The 14th International Workshop on Multi-Agent-based Simulation (MABS, AAMAS)*, 2013.

[18] G. Mitra and L. Mitra, *The Handbook of News Analytics in Finance*. John Wiley & Sons, Ltd., 2011.

[19] B. O'connor, R. Balasubramanyan, B. Routledge, and N. Smith, "From tweets to polls : Linking text sentiment to public opinion time series," in *International AAAI Conference on Weblogs and Social Media*, 2010.

[20] M. N. S. W. Lu Chen, Wenbo Wang and A. P. Sheth, "Extracting diverse sentiment expressions with target-dependent polarity from twitter," in *6th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2012.

[21] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Association for Computational Linguistics, 2005, pp. 347–354.

[22] M. Marchetti-Bowick and N. Chambers, "Learning for microblogs with distant supervision: political forecasting with twitter," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, ser. EACL '12. Association for Computational Linguistics, 2012, pp. 603–612.

[23] S. Moghaddam and F. Popowich, "Opinion polarity identification through adjectives," *CoRR*, 2010.

An Overview of the AMUSE Social Gaming Platform

Federico Bergenti
Dipartimento di Matematica e Informatica
Università degli Studi di Parma
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: federico.bergenti@unipr.it

Giovanni Caire and Danilo Gotta
Telecom Italia S.p.A.
Via Reiss Romoli 274, 10148 Torino, Italy
Email: {giovanni.caire, danilo.gotta}@telecomitalia.it

Abstract—**This paper presents an overview of the novel platform *AMUSE* (*Agent-based Multi-User Social Environment*), an agent-based social gaming platform that leverages the power of industrial-strength agent technologies. The core need that motivated the initial work on AMUSE was to provide game developers with a solid tool targeting common horizontal issues in social gaming, like user management and game state management, for games with synchronous and asynchronous interactions. AMUSE fulfills such a need by means of industrial-strength agent technology. Actually, AMUSE is not only a development framework that can be effectively used to implement prototypes and small-scale games with just a few concurrent players. Rather, it is thought as a *PaaS* (*Platform as a Service*) tool that enables service provides, like game portals and community portals, to relief game factories from the burden of implementing horizontal functionality that are common to a large set of games. This paper is a first presentation of the work on AMUSE and it starts framing AMUSE into the scope of social gaming. Then, the paper describes the architecture of the multi-agent system that represents the core of AMUSE and it relates the presented agent types with the functionality that AMUSE provides. Finally, the paper outlines some directions of future development.**

I. Introduction

This paper describes a recent work in the important industrial sector of online social games: an agent-based innovative platform that leverages the power of industrial-strength agent technologies to provide game developers with horizontal features that are common to most, if not all, online social games. Such a platform, namely *AMUSE* (*Agent-based Multi-User Social Environment*), gives developers a set of functionality that free them from the burden of implementing, and possibly reimplementing over and over again, common features like user management and game state management. The approach that AMUSE fosters lets developers concentrating their effort on game-specific features, it ensures solidity, and it ultimately reduces time-to-market and increases product quality.

AMUSE is designed to meet the requirements of large-scale service providers and it is intended for a *PaaS* (*Platform as a Service*) usage in large-scale scenarios. This, combined with the expected scalability of underlying agent technology, makes AMUSE an ideal tool for experimental prototypes intended can scale up to large-scale services. In fact, AMUSE is developed on top of *WADE* (*Workflows and Agents Development Environment*) [3], [24], the popular open-source platform for *agent-based BPM* (*Business Process Management*). One of the key characteristics of WADE is that it can be easily deployed on commodity computers and networks, and it can also be smoothly scaled up to huge services. We use WADE every day in our laboratories, and it is worth noting that the same software has been in daily use over the last 5 years [23] for large-scale network and service management in Telecom Italia for more than 8.95 million broadband connections for retail and business customers over a network of 114 million km of copper lines and 5.7 million km of optical fibers [22]. This is the reason why we say that the choice of implementing AMUSE on top of WADE ensure low-budget game development, giving the possibility to deploy the platform, and then smoothly scale up the service to a large number of users and to hosted deployment, if needed.

WADE is essentially the main evolution of *JADE* (*Java Agent and DEvelopment framework*) [4], [5], [7], [8], [16], the open-source framework that facilitates the development of interoperable multi-agent systems. JADE has been used in many research and industrial systems at an international scale since its initial development back in 1998 and today it is a reference for industrial-strength agent technology. WADE mainly adds to JADE the support for the execution of tasks defined according to the *workflow metaphor*, and it also provides a number of mechanisms that help managing the inherent complexity of a (distributed) multi-agent system both in terms of administration and fault tolerance.

While we measure a decline in investments in social gaming [1], social gaming, and mobile gaming in particular, is still on the rise from a game count perspective, with the industry seeing a 105% [18] increase in the number of mobile and social games on the market since 2000. Moreover, the industry experienced its biggest boom just last year, in 2012, when total games reached from 90 million to more than 211 million total [21]. This is sufficient to justify a research investment in this industrial sector as it is one of the driving forces of IT today.

This paper is not only intended to provide an overview of AMUSE, rather it is also meant to frame the work on AMUSE in the broad scope of social gaming in order to motivate, identify and justify the core design decisions. In the following section we frame AMUSE into the broad research on social gaming by giving essential definitions and terminology. Then, in Section III we outline the coarse-grained architecture of AMUSE and we detail the roles of single agents and their responsibilities. Finally, we conclude the paper with a brief summary of the work and with some insight on future developments of AMUSE.

II. What is Social Gaming?

The industry of video games, and *online video games* in particular, plays a significant role in our society that has been recently boosted by the pervasive diffusion of games for mobile appliances. Actually, the peculiar combination of novel game dynamics with the functionality of modern mobile appliances, like undisrupted connectivity, advanced graphics and sound capabilities, and on-board sensors, ensures a prolific and long lasting synergy between the industries of mobile appliances and video games.

A. Basic Terminology

Scientists and philosophers from diverse background have been discussing the notion of *play* and *game* for a long time, and they have already established agreed results and terminology.

One of the most cited attempts to characterize the play activity dates back to mid-50's to Huizinga's *Play Theory*. In his seminal book, best known as *Homo Ludens*, Huizinga characterizes the play activity as:

> . . . a free activity standing quite consciously outside "ordinary" life as being "not serious" but at the same time absorbing the player intensely and utterly. It is an activity connected with no material interest, and no profit can be gained by it. It proceeds within its own proper boundaries of time and space according to fixed rules and in an orderly manner. It promotes the formation of social groupings that tend to surround themselves with secrecy and to stress the difference from the common world by disguise or other means. [14]

Even if this characterization of the play activity has undergone very reasonable critiques over the years, it is worth noting that the act of playing is always associated with a social nature, and we can broadly say that the play activity is social *per se*.

Unfortunately a characterization of the play activity is not enough because we generally *differentiate games from play*. The Huizinga's characterization includes rules in the play activity, but such rules are always flexible and subject to change, with no real need for rules to be agreed or adopted beforehand. On the contrary, games are based on rules that are, often implicitly, adopted and that are not subject to frequent or unjustified change. Rules structure games, and make them repeatable.

One of the most cited definitions of game, which has been recently developed in the scope of video games by Juul, defines a game as a:

> . . . rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable. [17]

Games inherit much from the play activity and all games are social in some sense, if nothing else, because players often retell their experiences.

Finally, to better understand the landscape of social gaming, we should remember that *gameplay* is the specific way in which players interact with a game and, in particular, with a video game. We can adopt one of the many available definitions of gameplay as follows:

> Gameplay is the formalized interaction that occurs when players follow the rules of a game and experience its system though play. [20]

Once we are happy with the fact that the play activity and games are social in nature, we need to discuss how so called *social online games*, or *social games* for short, differentiates from other forms of games. This discussion has lead us to identifying salient characteristics of social games that any social gaming platform like AMUSE is demanded to provide.

B. A Characterization of Social Gaming

A primitive approach is to take the platform perspective and mark as social any game that use a *social network platform*. These are the so called *social network games* and the pervasiveness of online social networks in our society makes them one of the most important examples of social games. Any game delivered via, e.g., Facebook, is a social network game but unfortunately this is by far not enough to allow us to descend any salient characteristic of a social gaming platform.

A less primitive approach leads to a notable body of literature that identifies many dimensions of social gaming. Here we restrict to a threefold characterization that relates to the *timing of social interactions* and to the *type of social relationship* [19]. Together, these characteristics encapsulate the social interactions of most online games, including the two extremes of the range, namely, *MMOs* (*Massive Multi-player Online games*), that group hardcore players in large and long-lasting games, and *casual games*, targeted at and used by a mass audience of casual players for short burst. A real-world social gaming platform should be able to provide support for the whole, or at least for a large part, of this spectrum, and the scalable design of AMUSE ensures this.

In summary, the three characteristics of social gaming that we consider here, and that we detail below, are [19]:

- *Synchronous vs. asynchronous player interaction.* Do interactions occur simultaneously in real time or at different times as in a turn-based game?
- *Symmetrical vs. asymmetrical relationship formation.* Does forming a relationship require input from both parties or can they be formed unilaterally by a single party?
- *Strong tie vs. loose tie relationship evolution.* Do relationships tend to become deep and long lasting or are they more likely to be light and transitory?

The remaining of this section is devoted to analyzing such characteristics and providing example of how they are concretely adopted in social games.

Synchronous vs. Asynchronous Interaction. The superficial understanding is that MMO games feature synchronous, real-time play while casual social games are asynchronous with interaction occurring at disconnected times. However, all MMOs also feature important asynchronous features like in-game messages, and some Facebook game employs synchronous features such as a chat. Rather than an absolute position, current social games tend to offer a mix of synchronous and asynchronous interactions. Some games may highlight one or the other, but there are many that use both to establish a richer layer of engagement and retention.

The idea of synchronous gameplay is intuitively easy: players interact in real time rather than taking turns. Examples of synchronous social interactions include text chat, voice chat, video chat, and game elements like battles. Synchronous interactions can scale from two players to large groups.

The term *asynchronous game* might at first remind images of something slower and less intriguing, but asynchronous games can be just as engaging as synchronous ones, e.g., think of playing chess with a remote friend. Asynchronous social games come in different basic flavors, with some of the more common being:

- *Turn-based shared games.* They work well socially because each move is a mini game and there is social pressure to come back and complete the next turn. Moreover, bite-sized gameplay is easy to fit into schedules and players can play multiple games at once.
- *Turn-based challenge games.* Essentially, one of such games quickly becomes a set of two separate matches. Player 1 challenges and then player 2 responds; aggregate score determines the winner. They work socially because of the social pressure to return challenge and there is less waiting than shared turn-based since each player can complete his/her entire game independently.
- *Score-based challenge games.* These are the traditional *beat my high score* format. These games work socially because of the social pressure to return challenge and there is less waiting than turn-based options since players can try for their high scores anytime. Obviously, these types of games can be less interactive than other types.
- *Open-world asynchronous games.* In many ways, this is the most common Facebook game model. It works socially because the model supports a variety of game modes, including single-player and multi-player. It can variably approximate MMO experience without incurring into the technical issues of real-time play. Moreover, it still offers convenience of more casual games, i.e., players can play at different times and for short bursts.

Having said this, it is worth noting that chats are a powerful synchronous tool for player engagement and retention in both casual games and MMOs that deserves special attention, especially from the platform point of view. The chat, as a part of the game experience, always has a similar effect: boosting player engagement and facilitating long-term retention. When there is a real, vibrant support community present, players come back to a game more often and are less likely in search for other games.

Symmetrical vs. Asymmetrical Relationship. Perhaps the clearer example for understanding this characteristic of social gaming is the formation of social connections in Facebook versus Twitter. Facebook social relationships are symmetric: a member of the community asks to be a *friend* of another member and the latter must agree in turn for the relationship to exist. This approach as the advantage of making the acknowledgment mutual between parties and thus allowing for deeper sharing. On the contrary, the interaction is often limited to confirmed friends and friend relationships require (sometimes complex) management tools.

Examples of symmetric social interactions in online gaming include friending, gifting, trading, and private chatting on an individual scale, and parties, alliances, and manual multiplayer matchmaking on a group scale.

Twitter social relationships are asymmetric: a member of the community can *follow* anyone, without their reciprocation. This approach enables a widespread broadcasting and facilitates rapid dissemination of information. On the contrary, it requires less investment in social relationship and can be more prone to unsolicited interactions because communication filters are necessarily less sophisticated.

Examples of asymmetric social interactions include following, broadcasting, tweeting, and blogging on an individual scale, and public quests, factions, and random matchmaking on a group scale.

Although Facebook games are less known for such symmetric relationships, they do exist in many games. The *neighbor* approach prevalent in many games is a symmetrical social relationship. Even players who are in the same game and that are already Facebook friends still need to become *neighbors*.

Facebook games also feature numerous asymmetric social interactions. Instead of the neighbor approach, many games simply add a player's Facebook social graph directly to his/her game without requiring the permission of friends. These types of relationships are shallower than the ones originating from the neighbor approach but, because they are so broad, they lower the barriers to interaction and they create a high-density of ties among players.

Asymmetric relationships also exist in MMOs. A great example of this is the *public quest*. For example, if a public enemy is attacking the area, anyone who comes within a certain range is automatically considered to be participating in the public quest to capture him/her. Players can quickly and easily get a taste of group play and then go their own ways afterwards. The low social barrier allows for more frequent cooperation.

Strong Tie vs. Loose Tie. The former symmetry characteristic describes how relationships form but it does not necessarily dictate how they evolve. Ultimately, the relationship depends on what happens after the relationship itself has just been established. This characteristic is a simple measures of the evolution of a social relationship with a focus on the depth of interaction.

Examples of strong-tie gaming relationships range from the smallest scale, i.e., two players co-operatively play, to the group scale. Examples of loose-tie relationships also range from the smallest scale, e.g., game neighbors, to the group scale.

III. The AMUSE Platform

AMUSE (*Agent-based Multi-User Social Environment*) is an open-source development platform that can be downloaded from JADE Web site [16] and that is intended to tackle specific issues of social games, as discussed previously in this paper. The platform can be easily deployed in an *in house* setting, but it is designed to give service providers a tool to implement a PaaS with specific features of social games.

A. The Architecture of AMUSE

The characterization of social games sketched in Section II does not provide any means to concretely implement the desired features in a platform. In other words, we can classify a part of a game as employing symmetrical or asymmetrical relationships, but we have no best-practice tool to offer to developers to implement either symmetrical of asymmetrical relationships. *Gameplay design patterns* [10] are a pattern language that summarizes best practices in video game development and they are good reference for a list of features that a social gaming platform should provide.

Not all the over 700 gameplay design patterns identified in the literature and collected by the *Gameplay Design Pattern Project* [12] are interesting from the point of view of AMUSE. Some pattern is not related to the social aspect of games, while others are intended to provide specific features to games and they do not identify platform-level abstractions. We restrict here only to the best known gameplay design pattern that can contribute to the identification of the features that a social gaming platform like AMUSE should provide.

Before going into the details of the gameplay design pattern that AMUSE adopted, we need to clarify some underlying design decisions. First, we always assume the availability of a lower-level infrastructure for managing social relationships between users. This can be either a third-party infrastructure, like Facebook, providing a rich user profile and counting a large number of relationships between users; or it can be a private infrastructure accessible only from within AMUSE games, and normally providing a restricted user profile and a restricted set of relationships. AMUSE provides a generic interface that hides to the developers whether the infrastructure is third-party or private, thus ensuring scalability and allowing for the right infrastructure to be adopted for each game.

Another early decision that was taken is that AMUSE should provide a very flexible and highly scalable environment capable of scaling up with the success of a game. This enables early prototypes and low-cost experiments that can scale up to huge phenomena. *WADE* (*Workflows and Agents Development Environment*) [3], [24], the open-source platform for *agent-based BPM* (*Business Process Management*), is the ideal tool for ensuring such a high level of scalability and flexibility because one of its key characteristics is that it can be easily deployed on commodity computers and networks, and it can also be smoothly scaled up to huge services like nationwide network and service management [22].

Having said this, we simply decided to follow the best practices of WADE development and identified a set of *back-end agents* running on the server-side platform that communicate with *front-end agents* in charge of managing the actual interaction with the user.

Back-end and front-end agents implement the social gaming features of the platform depending on the centralization required by each and every single feature. For example, involving players in a mobile game does not always require a centralized authority: the agent on the user's mobile device contacts the respective agents of other users by means the users' profile stored locally in the device. AMUSE provides such a feature by assigning specific responsibilities to front-end agents. On the contrary, the management of a *table* in a *room* to let players engage in synchronous card game implies some centralized management of tables and rooms, as AMUSE actually provides.

In summary, the design of AMUSE comprises the following types of back-end agents that cooperatively deliver the functionality of the platform together with front-end agents:

- *UMA* (*User Manager Agent*). A socially inclined evolution of user manager agents of many other agent-based systems that manages the profile of single users and his/her relationships with other users. It relies on the underlying social network infrastructure for the concrete storage and discovery of profiles and relationships.
- *GRA* (*Games Room Agent*). It is an agent in charge of managing the shared game space in games with synchronous interaction. It can be effectively used to deliver asynchronous interaction if a back-end support is needed.
- *AMA* (*Application Manager Agent*). Taking the PaaS perspective, it is the agent in charge of managing the provided games and their lifecycle.
- *MTA* (*Match Tracer Agent*). It serves the needs of games that require a persistent game state and that needs restart options.

Finally, AMUSE provides a generic *MMA* (*Match Manager Agent*) in charge of interfacing the client application with back-end agents and to deliver the features that do not require a back-end support. For the current design, it is the only front-end agent that AMUSE provides.

B. Adopted Gameplay Design Patterns

After this brief sketch of the architecture of the multi-agent system that implements AMUSE functionality, we can briefly enumerate the gameplay design pattern that AMUSE uses internally, and that implement the features for game developers. Not all the following gameplay design pattern are currently implemented in AMUSE, but they are all important to grasp where AMUSE intends to go in the long term.

Some of the gameplay design patterns adopted in the design of AMUSE deal with the state of the game and with its evolution over time. These are implemented by the *GRA*, and the *GRA* itself provides an abstract view of the game state:

- *Private game spaces.* The game has parts of the game world that only a single player can manipulate directly.
- *Massively single-player online games.* These are games that make use of other players' games instances to provide input to the game state.

Together with the *MTA*, the *GRA* also implement the *persistent game world* design pattern, intended to make the game state independent from individual players' game and play session. The game state is available continuously and in continuous evolution.

Other gameplay design patterns that the *GRA* implements regard the management of how time is correlated with the evolution of the game, as follows:

- *Tick-based games.* The game progresses according to real-time, but in discrete steps.
- *Events timed to real world.* Gameplay events are initiated by specific real-time events occurring.

The *UMA* implements with no specific cooperation with other agents the *extra-game events broadcasting*, that allows game events to be broadcast in a medium where others can perceive them. This pattern is implemented by the *UMA* because it is the only agent that can interact with the underlying online social network infrastructure and its notification services.

Moreover, the *UMA* together with the *GRA* implement the following design patterns:

- *Drop-in/Drop-out.* Players entering and leaving ongoing game sessions are welcome.
- *Public player statistics.* The platform provides a means to publicize the player statistics inside and outside the game to users. The underlying online social network infrastructure is used to access the users' relationships and to publicize the statistics.
- *Visits.* Under the application of *UMA* policies, the *GRA* can provide temporary access to other players' private game spaces.
- *Invites.* Under the application of *UMA* policies, and taking into account the logics of the underlying social network infrastructure, the *GRA* allows inviting new players to a game as game actions.

Finally, only the *non-player help* design pattern request the cooperation of the *UMA*, the *GRA* and the *MMA* to ensure that players can receive help in the games by actions from those not playing.

C. The Implementation of AMUSE So Far

The current implementation of AMUSE does not yet provides all features described in the previous section, even if a clear plan is drawn to achieve full functionality briefly.

At the time of writing AMUSE provides a *UMA* with restricted functionality that manages a private online social network infrastructure. It is also able to manage various possible game involvement schemes and it is the final responsible for stored user profiles, which also include public game statistics.

Prototype *GRA* and *MMA* are provided to implement games with synchronous or asynchronous interactions. For the moment, the type of interaction that characterize the game is statically assigned and a game cannot have diverse parts with diverse types of interactions, nor it can dynamically change the type of interaction.

The available *MMA* is also in charge of providing a text channel that players can use for social interactions parallel to the actual play activity.

The current *GRA* prototype provides a game state representation that has proven valid for a wide range of games and it is based on the abstractions of *rooms* and *tables*, that players share and that provide the principal means of interaction during the game.

Taking the PaaS perspective, AMUSE now provides a prototype *AMA* that manages the interaction with the underlying WADE platform and that enables developers to choose between *decentralized* (or *client-only*) and *server-based* types of deployment. For the case of server-based deployment, the AMA already provides the features needed to leverage the flexible deployment schemes of WADE, which ensures that the server would not become a system bottleneck.

Moreover, the available *AMA* provides all needed administration services to quickly set up a private gaming infrastructure that, thanks to WADE, can scale up far over the initial deployment.

Finally, AMUSE includes a prototype *MTA* that is in charge of using WADE persistency support to implement persistent and restartable game state.

Front-end agents are now restricted to Android agents or desktop agents because, at the moment, no porting of JADE is available for other platforms. All in all, the flexible and somehow standardized agent communication protocol makes the interface between back-end and front-end agents fully device agnostic, and we see no problems is accommodating new and unexpected user devices in the architecture.

It is worth noting that current AMUSE prototype already includes Web games because we assume that such games can be structured into a lightweight client module connected to a heavyweight server module. We can already have server-side agents, running inside JADE/WADE containers, that communicate with the lightweight-client user interface via one of the available Web communication protocols, e.g., WebSockets.

IV. Conclusions

This paper presents the basic ideas that guided the development of AMUSE, a novel agent-based social gaming platform. The initial motivation for this work is that we felt the urge for a sharable tool capable of providing horizontal features of social gaming and we thought that agent technology, and WADE in particular, would have been ideal for this. Agent technology has already been applied to foster collaboration (see, e.g., [9]) and, more recently, it was used to address large-scale social networks (see, e.g., [6]), thus providing a solid base for the coordination of large communities. The initial vague idea that stimulated this work eventually turned into a complex architecture that encompasses back-end agents and front-end agents cooperatively providing social gaming features to developers.

AMUSE leverages the power of WADE to provide game developers and social gaming service providers with a scalable architecture with applicability ranging from initial prototypes to large-scale deployment. We think that this is a very important feature of AMUSE because it restricts the time-to-market and it extends the range of possible AMUSE developers to the open-source community, which is provided with fully open-source tools.

At present AMUSE uses WADE only for its proven features of flexibility and scalability in deployment. It does not really take advantage of the other major feature of WADE, namely its workflow-based development approach. This ensures that no game developers needs to understand and appreciate the flexibility of workflow-based development, but it also allows advanced developers to make an effective use of workflows to implement very dynamic games where parts of the game can be visually programmed, possibly by players.

At the moment we are investigating the possibility of providing a generic lightweight Web client using the *GWT* (*Google Web Toolkit*) [15] to give Web developers a minimalistic JADE implementation to adopt for their games. All in all, this is like using the Web browser as a single-agent JADE/WADE container using a proprietary protocol that is bridged to the common agent communication protocols by the Web server. A very similar approach has been available in JADE for more than a decade under the name of *split container*.

At the time of writing, AMUSE has been tested and validated by means of *four* mobile games and at least two of them will be released open-source together with AMUSE. Such games have been chosen to put in practical usage most of the functionality that AMUSE provides and to testbed the usability of AMUSE to implement fully fledged mobile games.

The first game that we developed, codename *Numblers*, is a number board game with asynchronous interactions that closely follows the lessons learned from largely appreciated games like *Ruzzle*. The dynamics of game engagement and gameplay does not need the support of back-end servers, and this game can be ideally deployed with no server support.

The second game, codename *TwentyOne*, is a variant of Numblers based a different game challenge, and it was chosen to try different ways for delivering similar functionality.

The third game developed so far, codename *BattleSpheres*, is a synchronous, real-time game that has been developed with the help of *AndEngine* [2]. In BattleSpheres, player A challenges player B by throwing virtual balls towards him/her and B is expected to block such balls before they reach the bottom of his/her screen. This game uses the experimental real-time features of AMUSE.

Finally, the fourth game is *Wadeoku*, a synchronous variant of Sudoku puzzles intended to have a group of players synchronously sharing a Sudoku board and gaining points for every good assignment of a number to an empty cell. This game does need a significant support from back-end agents and it is close to the real use of AMUSE that we foresee in the near future.

The development experience gathered with such four games can be considered positive and the early experimentation on concrete examples provided significant feedback on core platform-level decisions. In addition, early experimentation allowed us to identify interesting best practices in the utilization of AMUSE that were not initially envisaged.

AMUSE is open-source and it can be downloaded from JADE Web site [16].

References

[1] A. J. Agnello. *Investment in social gaming drops by $1 billion in 2012*. Available at http://www.digitaltrends.com

[2] AndEngine Web site. http://www.andengine.org

[3] M. Banzi, G. Caire, D. Gotta WADE: A software platform to develop mission critical, applications exploiting agents and workflows. Procs. Int'l Conf. *Autonomous Agents and Multi-Agent Systems*, 2008.

[4] F. Bellifemine, G. Caire, D. Greenwood. *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology, 2007.

[5] F. Bellifemine, A. Poggi, G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software: Practice & Experience*, 31:103–128, 2001.

[6] F. Bergenti, E. Franchi, A. Poggi. Selected models for agent-based simulation of social networks. Procs. *Symposium on Social Networks and Multiagent Systems*, 2011.

[7] F. Bergenti, A. Poggi. Ubiquitous Information Agents. *Int'l J. Cooperative Information Systems*, 11(3–4):231–244, 2002.

[8] F. Bergenti, A. Poggi, B. Burg, G. Caire. Deploying FIPA-compliant systems on handheld devices. *IEEE Internet Computing*, 5(4):20–25, 2001.

[9] F. Bergenti, A. Poggi, M. Somacher. A collaborative platform for fixed and mobile networks. *Communications of the ACM*, 45(11):39–44, 2002.

[10] S. Björk, S. Lundgren, J. Holopainen. Game Design Patterns. Procs. *Digital Games Research Conference*, 2003.

[11] G. Caire, E. Quarantotto, M. Porta, G. Sacchi. WOLF - An Eclipse Plug-in for WADE Procs. IEEE Int'l Workshops *Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2008.

[12] Gameplay Design Pattern Project Web site. http://gdp2.tii.se

[13] T. Grant. *The Art of Videogames*. Wiley-Blackwell, 2009.

[14] J. Huizinga. *Homo Ludens: A Study of the Play Element in Culture*. Beacon Press, 1955.

[15] GWT (Google Web Toolkit) Web site. http://www.gwtproject.org

[16] JADE (Java Agent DEvelopment framework) Web site. http://jade.tilab.com

[17] J. Juul. *Half-Real: Video Games between Real Rules and Fictional Worlds*. The MIT Press, 2005.

[18] S. Mlot. *Infographic: How Do You Get Your Mobile Gaming Fix?* Available at http://www.pcmag.com

[19] M. Ricchetti. *What Makes Social Games Social?* Available at http://www.gamasutra.com

[20] K. Salen, E. Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2004.

[21] E. Swallow. *What Makes a Good Social Game?* Available at http://www.forbes.com

[22] Telecom Italia S.p.A. Relazione Finanziaria Annuale 2012. Available at http://www.telecomitalia.com

[23] L. Trione, D. Long, D. Gotta, G. Sacchi. Wizard, WeMash, WADE: Unleash the power of collective intelligence. Procs. Int'l Conf. *Autonomous Agents and Multiagent Systems*, 2009.

[24] WADE (Workflows and Agents Development Environment) Web site. http://jade.tilab.com/wade

Replaceable Implementations for Actor Systems

Agostino Poggi
Dipartimento di Ingegneria dell’Informazione
Università degli Studi di Parma
Parma, Italy
agostino.poggi@unipr.it

***Abstract* — CoDE is an actor-based software framework aimed at both simplifying the development of large and distributed complex systems and guarantying an efficient execution of applications. This software framework takes advantage of a concise actor model that makes easy the development of the actor code by delegating the management of events (i.e., the reception of messages) to the execution environment. Moreover, it allows the development of scalable and efficient applications through the possibility of using different implementations of the components that drive the execution of actors. This paper introduces the software framework and shows how the performance of applications can be optimized by choosing the best combination among the alternative implementations of its components.**

Keywords - Actor model, software framework, concurrent programming, distributed systems.

I. Introduction

Distributed and concurrent programming have lately received enormous interest because multi-core processors make concurrency an essential ingredient of efficient program execution and because distributed architectures are inherently concurrent. However, distributed and concurrent programming is hard and largely different from sequential programming. Programmers have more concerns when it comes to taming parallelism. In fact distributed and concurrent programs are usually bigger than equivalent sequential ones and models of distributed and concurrent programming languages are different from familiar and popular sequential languages [12][14].

Message passing models seem be the more appropriate solution because they replace the sharing of data with the exchange of messages. One of the well-known theoretical and practical models of message passing is the actor model. Using such a model, programs become collections of independent active objects (actors) that exchange messages and have no mutable shared state [1][2][9]. Actors can help developers to avoid issues such as deadlock, live-lock and starvation, which are common problems for shared memory based approaches. There are a multitude of actor oriented libraries and languages, and each of them implements some variants of actor semantics. However, such libraries and languages use either thread-based programming, which makes easy the development of programs, or event-based programming, which is far more practical to develop large and efficient concurrent systems, but also is more difficult to use.

This paper presents an actor based software framework, called CoDE (Concurrent Development Environment), that has the suitable features for both simplifying the development of large and distributed complex systems and guarantying scalable and efficient applications. The next two sections introduce the software framework and its implementation. Section 4 shows how the possibility of configuring an application with different implementations of its components allows coping with performance and scalability problems. Section 5 introduces two simple applications and shows their execution times obtained with different configurations. Section 6 introduces related work. Finally, section 7 concludes the paper by discussing the main features of the software framework and the directions for future work.

II. CoDE

In CoDE a system is based on a set of interacting actors that perform tasks concurrently. An actor is an autonomous concurrent object, which interacts with other actors by exchanging asynchronous messages. Communication between actors is buffered: incoming messages are stored in a mailbox until the actor is ready to process them. Each actor has a system-wide unique identifier called its address that allows it to be referenced in a location transparent way. An actor can send messages only to the actors of which it knows the address, that is, the actors it created and of which it received the addresses from other actors. After its creation, an actor can change several times its behavior until it kills itself. Each behavior has the main duty of processing a set of specific messages through a set of message handlers called cases. Therefore, if an unexpected message arrives, then the actor mailbox maintains it until a next behavior will be able to process it.

An actor can perform five types of action:

- It can send messages to other actors or to itself.
- It can create new actors.
- It can update its local state.
- It can change its behavior.
- It can kill itself.

In particular, an actor has not explicit actions for the reception of messages, but its implementation autonomously extracts the new messages from the actor mailbox and then executes the actions for their processing.

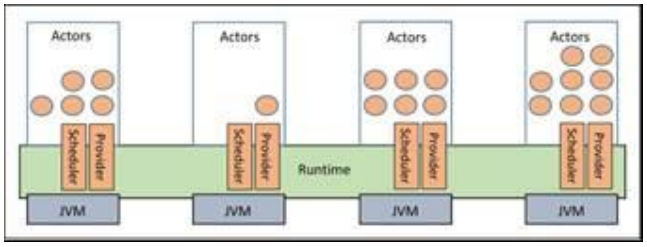


Fig. 1. CoDE distributed system architecture.

An actor can set a timeout for waiting for the next message and then execute some actions if the timeout fires. However, it has not explicit actions for monitoring the firing of such a timeout: its implementation autonomously observes the firing of the timeout and then executes the actions for its management.

Depending on the complexity of the application and on the availability of computing and communication resources, one or more actor spaces can manage the actors of the application. An actor space acts as "container" for a set of actors and provides them the services necessary for their execution. In particular, an actor space takes advantages of two special actors: the scheduler and the service provider. The scheduler manages the concurrent execution of the actors of the actor space. The service provider enables the actors of an application to perform new kinds of action (e.g., to broadcast a message or to move from an actor space to another one). Fig. 1 shows a graphical representation of the architecture of a CoDE distributed application.

III. Implementation

CoDE is a software environment implemented by using the Java language and takes advantage of preexistent Java software libraries and solutions for supporting concurrency and distribution. CoDE has a layered architecture composed of an application and a runtime layer. The application layer provides the software components that an application developer needs to extend or directly use for implementing the specific actors of an application. The runtime layer provides the software components that implement the CoDE middleware infrastructures to support the development of standalone and distributed applications.

A. *Actor Implementation*

An actor can be viewed as a logical thread that implements an event loop [4][13]. This event loop perpetually processes events that represent: the reception of messages, the behavior exchanges and the firing of timeouts. In particular, an actor is defined by five main components: a reference, a mailer, a behavior, a state and an execution manager. Fig. 2 shows a graphical representation of the architecture of an actor.

A reference supports the sending of messages to the actor it represents. Therefore, an actor needs to have the reference of another actor for sending it a message. In particular, an actor has have the reference of another actor if:

- It created such an actor (in fact, the creation method returns the reference of the new actor).
- It received a message from such an actor (in fact, each message contains the reference of the sender) or whose content enclosed its reference.

A reference has an attribute, called actor address, that allows to distinguish itself (and then the actor it represents) from the references of the other actors of the application where it is acting. To guarantee it and to simplify the implementation, an actor space acts as "container" for the actors running in the same Java Virtual Machine (JVM) and an actor address is composed of three components:

- An actor identifier that is different for all the actors of the same actor space.
- An actor space identifier that is different for all the actor spaces of the same computing node.
- The IP address of the computing node.

A mailer provides a mailbox for the messages sent to its actor until it processes them, and delivers its messages to the other actors of the application. As introduced above, a behavior can process a set of specific messages leaving in the mailbox the messages that is not able to process. Such messages remain into the mailbox until a new behavior is able to process them and if there is not such a behavior they remain into the queue for all the life of the actor. A mailbox has not an explicit limit on the number of messages that can maintain. However, it is clear that the (permanent) deposit of large numbers of messages in the mailboxes of the actors may reduce the performances of applications and cause in some circumstances their failure.

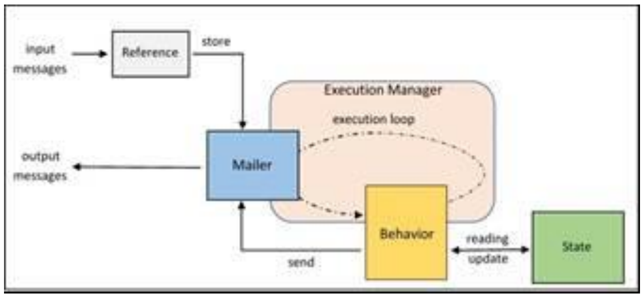


Fig. 2. Actor Architecture.

The original actor model associates a behavior with the task of messages processing. In CoDE, a behavior can perform three kinds of tasks: its initialization, the processing of messages and the management of message reception timeouts. In particular, a behavior does not directly process messages, but it delegates the task to some case objects, that have the goal of processing the messages that match a specific (and unreplaceable) message pattern.

Often the behaviors of an actor need to share some information (e.g., a behavior may work on the results of the previous behaviors). It is possible thank to a state object. Of course, the kind of information that the behaviors of an actor need to share depends on the type of tasks they must perform in an application. Therefore, the state of an actor must be specialized for the task it will perform.

A message is an object that contains a set of fields maintaining the typical header information and the message content. Moreover, each message is different from any other one. In fact, messages of the same sender have a different identifier and messages of different senders have a different sender reference.

An actor has not direct access to the local state of the other actors and can share data with them only through the exchange of messages and through the creation of actors. Therefore, to avoid the problems due to the concurrent access to mutable data, both message passing and actor creation should have call-by-value semantics. This may require making a copy of the data even on shared memory platforms, but, as it is done by the large part of the actors libraries implemented in Java, CoDE does not make data copies because such operations would be the source of an important overhead. However, it encourages the programmers to use immutable objects (by implementing as immutable all the predefined message content objects) and delegates the appropriate use of mutable object to them.

As introduced above, an actor behavior processes the received messages through a set of case objects and each of them can process only the messages that match a specific message pattern. In CoDE, a message pattern is an object that can apply a combination of constraint objects on the value of all the fields of a message and on the actor state. It improves the adaptability of actors to the changes of the environment they live. In fact, an actor can react to the changes by either moving to another behavior or by enabling, disabling or changing the cases that process the received messages depending on their current state.

An execution manager implements the basic functionalities of an actor on the top of the services provided by the runtime layer. In particular, it manages the life cycle of the actor by initializing its behaviors, by processing the received messages and the firing of message reception timeouts, and by moving it from a behavior to another one. The type of the implementation of an execution manager is one of the factors that mainly influence the attributes of the execution of an application. In particular, execution managers can be divided in two classes that allow to an actor either to have its own thread (from here named active actors) or to share a single thread with the other actors of the actor space (from here named passive actors).

B. Actor Space Implementation

An actor space has the duty of supporting the execution of the actions of its actors and of enhancing them with new kinds of action. To do it, an actor space takes advantage of some main runtime components (i.e., factory, dispatcher and registry) and of the two special actors: the scheduler and the service provider.

The factory has the duty of creating the actors of the actor space. In particular, it also creates their initial behavior, chooses their most appropriate execution manager and delegates the creation of their references to the registry.

The dispatcher has the duty of supporting the communication with the other actor spaces of the application. In particular, it creates connections to/from the other actor spaces, maps remote addresses to the appropriate output connections, manages the reception of messages from the input connections, and delivers messages through the output connections. This component works in collaboration with another component called connector.

A connector has the duty of opening and maintaining connections toward all the other actor spaces of the application. In particular, the connector of one of the actor spaces of the application plays the role of communication broker and has the additional duty of maintaining the information necessary to a new actor space for creating connections towards the other actor spaces of the application.

The registry supports the work of both the factory and the dispatcher. In fact, it creates the references of the new actors and supports the delivery of the messages coming from remote actor by proving the reference of the destination actor to the dispatcher. In fact, as introduced in a previous section an actor can send a message to another actor only if it has its reference, but while the reference of a local actor allows the direct delivery of messages, the reference of a remote actor delegates the delivery to the dispatchers of the local and remote actor spaces (see Fig. 3).

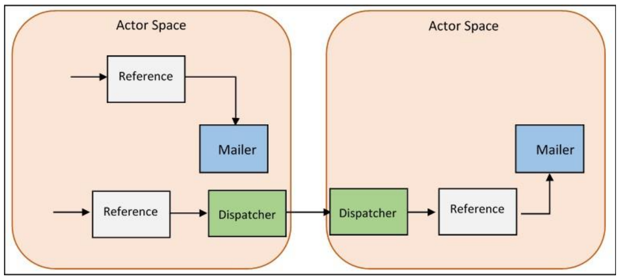


Fig. 3. Message dispatching.

The scheduler is a special actor that manages the execution of the actors of an actor space. Of course, the duties of a scheduler depend on the type of execution manager and, in particular, on the type of threading solutions associated with the actors of the actor space. In fact, while Java runtime environment mainly manage the execution of active actors, CoDE schedulers completely manage the execution of passive actors.

The service provider is a special actor that offers a set of services for enabling the actors of an application to perform new kinds of actions. Of course, the actors of the application can require the execution of such services by sending a message to the service provider.

Moreover, an actor space can enable the execution of an additional runtime component called logger. The logger has the possibility to store (or to send to another application) the relevant information about the execution of the actors of the actor space (e.g., creation and deletion of actors, exchange of messages, processing of messages and timeouts, exchange of behaviors). The logger can provides both textual and binary information that can be useful for understanding the activities of the application and for diagnosing the causes and solving the possible execution problems. Moreover, the binary information contain real copies of the objects of the application (e.g., messages and actor state); therefore, such an information can be used to feed other applications (e.g., monitoring and simulation tools).

Finally, the actor space provides a runtime component, called configurator, which simplifies the configuration of an application by allowing the use of either a declarative or a procedural method (i.e., the writing of either a properties file or a code that calls an API provided by the configurator).

IV. CONFIGURATION

One of the most important features of CoDE is the possibility of configuring an application with different implementations of the runtime components. For example, CoDE supports the communication among the actor spaces through four kinds of connector that respectively use ActiveMQ [23], Java RMI [15], MINA [3] and ZeroMQ [11]. Moreover, the service provider actor can offer an extensible set of services for enhancing the set of actions that the actors can perform. The current implementation of the software framework provides services for supporting the broadcast of messages, the exchange of messages through the "publish and subscribe" pattern, the mobility of actors, the interaction with users through emails and the creation of actors (useful for creating actors in other actor spaces).

However, the most important components that influence the quality of the execution of an application are the execution manager and the associated scheduling actor. In fact, the use of one or another couple of execution manager and scheduling actor causes large differences in the performance and in the scalability of the applications.

CoDE provides three types of execution managers and four types of execution scheduling actors. The first two types of execution manager respectively support the implementation of active and passive actors (active and passive executors). The third type provides a special implementation of passive actors in which the actors receive message from a shared queue (shared executors). The first two types of scheduling actors manage the execution of the actor spaces, which contain either active or passive actors (active and passive schedulers). The third type manages the execution of the actor spaces where passive actors share the message queue (shared schedulers). Finally, the forth type manages the execution of the actor spaces where both active and passive actors are present (hybrid schedulers).

The identification of the best couple of execution manager and scheduling actor for a specific application mainly depends on the number of actors, the number of exchanged messages, the preeminent type of communication used by actors (i.e., point-to-point or broadcast) and the presence of a subset of actors that consume a large part of the computational resources of the application. Table 1 shows what should be the best choices for binary partition of the values of the previous parameters. In particular, the third column indicates the preeminence of either point-to-point communication (P) or broadcast communication (B), the forth column indicates the presence/absence of a subset of heavy actors and the word "any" is used when the value of the associate parameter has not effect on the choice of execution manager and scheduler.

TABLE 1

actors	**messages**	**P/B**	**Heavy**	**scheduler**
few	any	any	any	active
many	any	P	no	passive
many	few	B	no	passive
many	many	B	no	shared
many	any	any	yes	hybrid

V. EXPERIMENTATION

The performances of the different types of execution managers and scheduling actors can be analyzed by comparing the execution times of two simple applications on a laptop with an Intel Core 2 - 2.90GHz processor, 16 GB RAM, Windows 8 OS and Java 7 with 4 GB heap size.

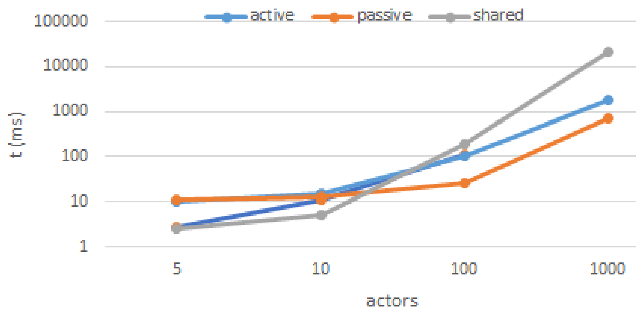


Fig. 4. Point-to-point message exchange example performances.

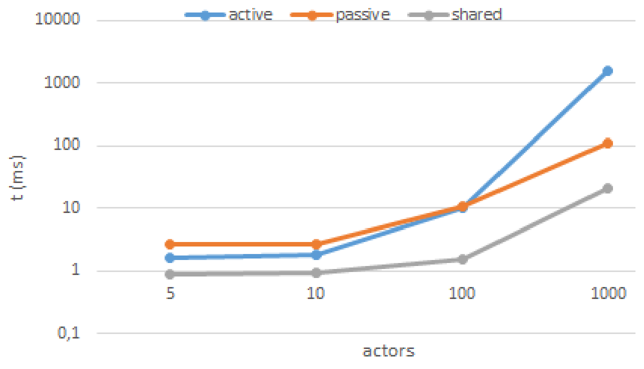


Fig. 5. Broadcasting example performances.

The first application is based on the point-to-point exchange of messages between the actors of an actor space. The application starts an actor that creates a certain number of actors, sends 1000 messages to each of them and then waits for their answers. Fig 4 shows the execution time of the application for 5, 10, 100 and 1.000 actors and, as introduced in table 1, the best performances are obtained with a passive executor and scheduling actor couple when the number of actors increases.

The second application is based on the broadcasting of messages to the actors of an actor space. The application starts an actor that creates a certain number of actors and then sends a broadcast message. Each actor receives the broadcast message, then, in its response, sends another broadcast message and finally waits for all the broadcast messages. Fig. 5 shows the execution time of the application for 5, 10, 100 and 1.000 actors and, as introduced in table 1, the best performances are obtained with a shared executor and scheduling actor couple.

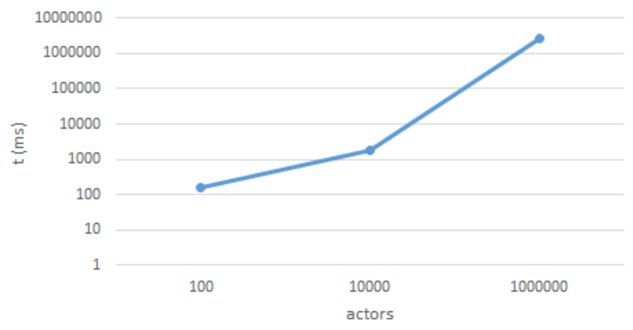


Fig. 6. Game of life performances.

Moreover, the use of passive actors allows the development of applications that scale to a large number of actors. In particular, the current implementation of the framework allows to scale up to a million of actors. Fig.6 show the execution times for 100 cycles of simulation of the game of live [8] for 100, 10.000 and and 1.000.000 actors.

VI. Related Work

Several actor-oriented libraries and languages have been proposed in last decades and a large part of them uses Java as implementation language. The rest of the section presents some of the most interesting works.

Salsa [27] is an actor-based language for mobile and Internet computing that provides three significant mechanisms based on the actor model: token-passing continuations, join continuations, and first-class continuations. In Salsa each actor has its own thread, and so scalability is limited. Moreover, message-passing performance suffers from the overhead of reflective method calls.

Kilim [24] is a framework used to create robust and massively concurrent actor systems in Java. It takes advantage of code annotations and of a byte-code post-processor to simplify the writing of the code. However, it provides only a very simplified implementation of the actor model where each actor (called task in Kilim) has a mailbox and a method defining its behavior. Moreover, it does not provide remote messaging capabilities.

Scala [9] is an object-oriented and functional programming language that provides an implementation of the actor model unifying thread based and event based programming models. In fact, in Scala an actor can suspend with a full thread stack (receive) or can suspend with just a continuation closure (react). Therefore, scalability can be obtained by sacrificing program simplicity. Akka [26] is an alternative toolkit and runtime system for developing event-based actors in Scala, but also providing APIs for developing actor-based systems in Java. One of its distinguishing features is the hierarchical organization of actors, so that a parent actor that creates some children actors is responsible for handling their failures.

Jetlang [22] provides a high performance Java threading library that should be used for message based concurrency. The library is designed specifically for high performance in-memory messaging and does not provide remote messaging capabilities.

AmbientTalk [4] is a distributed object-oriented programming language whose actor-based and event driven concurrency model makes it highly suitable for composing service objects across a mobile network. It provides an actor implementation based on communicating event loops [13]. However, each actor is always associated with its own JVM thread and so it limits the scalability of applications on the number of actors for JVM.

VII. Conclusions

This paper presented a software framework, called CoDE, which allows the development of efficient large actor based systems by combining the possibility to use different implementations of the components driving the execution of actors with the delegation of the management of the reception of messages to the execution environment.

CoDE is implemented by using the Java language and is an evolution of HDS [19] and ASIDE [20] from which it derives the concise actor model, and takes advantages of some implementation solutions used in JADE [16]. CoDE shares with Jetlang, [22] Kilim [24] and Scala [9] the possibility to build applications that scale applications to a massive number of actors, but without the need of introducing new constructs that make complex the writing of actor based programs. Moreover, CoDE has been designed for the development of distributed applications while the previous three actor based software were designed for applications running inside multi-core computers. In fact, the use of structured messages and message patterns makes possible the implementation of complex interactions in a distributed application because a message contains all the information for delivery it to the destination and then for building and sending a reply. Moreover, a message pattern filters the input messages on all the information contained in the message and not only on its content.

Current research activities are dedicated to extend the software framework to offer it as means for the development of multi-agent systems. Future research activities will be dedicated to the extension of the functionalities provided by the software framework and to its experimentation in different application fields. Regarding the extension of the software framework, current activities have the goal of providing a passive threading solution that fully take advantage of the features of multi-core processors, of enabling the interoperability with Web services and legacy systems [18], and of enhancing the definition of the content exchanged by actors with semantic Web technologies [21]. Moreover, future activities will be dedicated to the provision of a trust management infrastructure to support the interaction between actor spaces of different organizations [17][25]. Current experimentation of the software framework is performed in the field of the modeling and simulation of social networks [6], but in the next future will be extended to the collaborative work services [4] and to the agent-based systems for the management of information in pervasive environments [4].

REFERENCES

[1] G.A. Agha, Actors: A Model of Concurrent Computation in Distributed Systems, Cambridge, MA: MIT Press, 1986.

[2] G.A. Agha, I.A. Mason, S.F. Smith and C.L. Talcott, "A Foundation for Actor Computation," J. of Functional Programming, vol. 7, no. 1, pp. 1-69, 1997.

[3] Apache Software Foundation. (2013). Apache Mina Framework [Online]. Available: http://mina.apache.org

[4] F. Bergenti and A. Poggi, "Ubiquitous Information Agents," Int. J. on Cooperative Information Systems, vol. 11, no. 3-4, pp. 231-244, 2002.

[5] F. Bergenti, A., Poggi and M. Somacher, "A collaborative platform for fixed and mobile networks," Communications of the ACM, vol. 45, no. 11, pp. 39-44, 2002.

[6] F. Bergenti, E. Franchi and A. Poggi, "Selected models for agent-based simulation of social networks," in 3rd Symposium on Social Networks and Multiagent Systems (SNAMAS'11), York, UK: Society for the Study of Artificial Intelligence and the Simulation of Behaviour, 2011. pp. 27-32.

[7] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt and W. De Meuter, "Ambient-oriented programming in ambienttalk," in ECOOP 2006 – Object-Oriented Programming, Berlin Heidelberg: Springer, 2006, pp. 230-254.

[8] M. Gardner, The fantastic combinations of John Conway's new solitaire game Life. Scientific American 223:120-123, 1970.

[9] P. Haller and M. Odersky, "Scala Actors: Unifying thread-based and event-based programming," Theoretical Computer Science, vol. 410, no. 2-3, pp. 202–220, 2009.

[10] C. E. Hewitt, "Viewing controll structures as patterns of passing messages," Artificial Intelligence, vol. 8, no. 3, 1977, pp. 323–364.

[11] P. Hintjens, ZeroMQ: Messaging for Many Applications, Sebastopol, CA: O'Reilly, 2013.

[12] C. Leopold, Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches, New York, NY, USA: John Wiley & Sons, 2001.

[13] M. S. Miller, E. D. Tribble and J. Shapiro, "Concurrency among strangers," in Trustworthy Global Computing, Berlin Heidelberg: Springer, 2005, pp. 195-229.

[14] M. Philippsen, "A survey of concurrent object-oriented languages," Concurrency: Practice and Experience, vol. 12, no. 10, pp. 917-980, 2000.

[15] E. Pitt and K. McNiff, Java.rmi: the Remote Method Invocation Guide. Boston, MA, USA: Addison-Wesley, 2001.

[16] A. Poggi, M. Tomaiuolo and P. Turci, "Extending JADE for agent grid applications," in 13th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2004), Modena, Italy, 2004, pp. 352-357.

[17] A. Poggi, M. Tomaiuolo and G. Vitaglione, "A Security Infrastructure for Trust Management in Multi-agent Systems," in Trusting Agents for Trusting Electronic Societies, Theory and Applications in HCI and E-Commerce, LNCS, vol. 3577, R. Falcone, S. Barber, and M. P. Singh, Eds. Berlin, Germany: Springer, 2005, pp. 162-179.

[18] A. Poggi, M. Tomaiuolo and P. Turci, "An Agent-Based Service Oriented Architecture", in Proc of. WOA, Genova, Italy, 2007, pp. 157-165.

[19] A. Poggi, "HDS: a Software Framework for the Realization of Pervasive Applications," WSEAS Trans. on Computers, vol. 10, no. 9, pp. 1149-1159, 2010.

[20] A. Poggi, "ASiDE - A Software Framework for Complex and Distributed Systems," in Proc. of the 16th WSEAS Int. Conf. on Computers, Kos, Greece, 2012, pp. 353-358.

[21] A. Poggi, "Developing ontology based applications with O3L," WSEAS Trans. on Computers, vol. 8 no. 8, pp. 1286-1295, 2009.

[22] M. Rettig. (2013). Jetlang software [Online]. Available: http://code.google.com/p/jetlang/

[23] B. Snyder, D. Bosnanac and R. Davies, ActiveMQ in action, Westampton, NJ, USA: Manning, 2001.

[24] S. Srinivasan and A. Mycroft, "Kilim: Isolation-Typed Actors for Java," in ECOOP 2008 – Object-Oriented Programming, LNCS, vol. 5142, J. Vitek, Ed. Berlin ,Germany: Springer, 2008, pp. 104-128.

[25] M. Tomaiuolo, "dDelega: Trust Management for Web Services," Int. J. of Information Security and Privacy, vol. 7, no. 3, pp. 53-67, 2013.

[26] Typesafe. (2013) Akka software [Online]. Available: http://akka.io

[27] C. Varela and G.A. Agha, "Programming dynamically reconfigurable open systems with SALSA," SIGPLAN Notices, vol. 36, no. 12, pp. 20-34, 2001.

Trust Negotiation for Automated Service Integration

Filippo Agazzi, Michele Tomaiuolo
Dipartimento di Ingegneria dell'Informazione
Università di Parma
Parma, Italy
{agazzi, tomamic}@ce.unipr.it

Abstract—**This paper presents a generic Trust Negotiation framework for Web services, based on the WS-Trust standard. It allows users to create trust incrementally, by disclosing credentials step by step. This way, services and resources can be shared in an open environment, and access can be realized on the basis of peer-to-peer trust relationships. The paper also describes a practical implementation of the framework, which integrates a modular trust engine and a rule engine, which is used as a policy checker.**

Security; trust; Web services; rule-based systems

I. Introduction

The automatic or assisted management of trust relationships is a fundamental requirement to allow the provision and use of disparate services in an open environment. At the global scale, the assumption that all users are known in advance, or they can be easily managed through a traditional Access Control List, is not realistic. In fact, the potential user base of an application provided on the open Internet is still growing, with the mass adoption of social networking tools. Since nowadays contacts among people may develop fully online, possibly with no body of knowledge to associate with a name, more flexible schemes are needed. Currently, no general solutions are available for the problem of identity management, assuming a global database of names and personal profiles is both unfeasible and undesirable. Moreover, online interactions may involve human users together with software agents, possibly with a common understanding of the exchanged messages, on the basis of Semantic Web technologies [1]. Given such a new way people are using the Internet today, the approach of Automated Trust Negotiation (ATN) [2][3] is becoming relevant, because it allows unknown users and agents desiring to share any resource or service, to establish a level of trust in an incremental way through the exchange of credentials.

In this scenario, the open selection and composition of services is made possible, since ATN simplifies the creation and management of trust bounds. In fact, delegation and workflow composition [4] may only be applied on the basis of careful protection of resources and information. This requires a clear analysis of risks and opportunities associated with local trust bounds, on the basis of their socio-cognitive constituents [5], including competence, disposition, dependence and fulfillment. The problem of authorization can thus be solved in a fully distributed way, as access rights can be assigned and delegated on the basis of the local trust assumptions, in a typical Trust Management scheme [6].

This paper is organized as follows: Section II presents an overview and a literature review of ATN; Section III describes a generic trust negotiation framework for Web services, based on the WS-Trust standard; Section IV provides details about practical implementation and use of such a framework, including first performance results; finally, some concluding remarks are provided.

II. Background

A *credential* is generally defined as a digital certificate attesting, via a digital signature, the association of one or more attributes to an entity, identified through its public key. This entity, i.e. the certificate subject, can attest the ownership of the presented credential by demonstrating to possess the corresponding private key. Notably, the entity that originally issued and signed the certificate is not necessarily requested to participate directly in the verification process.

Attributes in a credential can be considered sensible or not. The case of non-sensible attributes does not require any particular care. On the contrary, for the case of sensible attributes, it is necessary to build a certain level of trust between negotiating parties via a structured list of release conditions. Such release conditions are generally known as *policies*. An *access policy* for a resource R is a boolean function, which allows or denies access depending on disclosed credentials. It can be written as: $f_R(C_1, C_2, \dots C_n)$, where each C_i is a credential which may be possessed by the requester. A credential itself often holds sensitive information, and it needs to be protected. Thus a *credential disclosure policy* should be defined for revealing a certain credential C. It will be a boolean function of the form: $f_C(C_1, C_2, \dots C_n)$. Finally, for selecting the credentials to disclose, a client could need to access a service policy P. But also this policy can be considered reserved. In this case, it should be associated with a *policy disclosure policy*: $f_P(C_1, C_2, \dots C_n)$. That is, credentials and policies are to be considered as sensitive resources, and thus they need to be protected by access policies, along other kinds of resources. Access control can be implemented on the basis of different kinds of security credentials, including X.509 certificates and SAML assertions. Moreover, different languages have been defined to represent policies [7][8] in an appropriate and expressive way.

These are the cases where trust negotiation provides its full benefits. Digital credentials are exchanged step by step, to increase the level of trust between involved parties, and the flow of credentials between two entities through a sequence of requests and releases is what is actually intended with trust negotiation.

A policy language for Trust Negotiation must allow to specify all these conditions. A policy has to be considered satisfied only when the requester discloses all the required credentials, and this verification requires to use a formal policy language, with precise semantics. Another important consideration is that, to fit the wide Internet, such language has to be comprehensible and agreed by all involved entities. In the last years, scholars and firms have proposed various languages, like the IBM Trust Policy Language or the Role-Based Trust Management Language (RT). All those languages, however, were related to some particular engines to compute and decide about certain policies. Moreover, a number of languages are being proposed by ongoing research works, but with a limited scope of application, to be shared by some nodes which interact using the same framework or the same software infrastructure, for example in the context of Web Services.

With respect to the management and computation of policies in a trust negotiation, a particularly important element is the policy compliance checker. Starting from a policy and a set of credentials, the policy compliance checker must be able to find the credentials which satisfy the policy, if they are effectively available as a subset of all disclosed credentials. For this purpose, it is also necessary to translate each credential from its original format into an assertion of the policy language. Considering the example of a client requesting a service, one of the problems to solve is how the client comes to know which credentials it is required to present, and how the policies protecting the service and the credentials are disclosed.

A. Negotiation Strategies

From the architectural point of view, each entity participating in an Automated Trust Negotiation has a Security Agent (SA). The SA has the fundamental responsibility of managing the negotiation, computing available policies and credentials, both the local ones and those disclosed by a remote entity, and taking the decision to authorize the disclosure of some credentials and policies at a given phase of a negotiation. These decisions, as well as the exchanged messages and the disclosure of policies and credentials, can be conducted in a number of ways, which is essentially unlimited. A negotiation strategy defines the protocol for the modality and decisions.

The main goal of a strategy is to reach a successful completion of the negotiation protocol, in the respect of certain requirements. A strategy decides when and which credentials must be disclosed and inserted into a message to send to the other party; how much computational load to dedicate to the negotiation (e.g., the maximum number of rounds) and other decisions about the behaviour to pursue during the negotiation. Moreover, a negotiation is not always possible, since for example one of the two parties does not possess sufficient credentials: the strategy has to determine the moment to abandon the negotiation, since it is not possible to conclude it with success.

The execution of a negotiation requires some agreement on a common protocol, with the intended agreement that each subject is free to apply a possibly different strategy. The characteristics of a negotiation are defined by the adopted strategies. Some of the tasks of such strategies are related to which credentials are released, when they are released, which parties are required to unlock the release of another credential and when the negotiation closes, successfully or not. The success of a negotiation is not always possible. One of the subjects could not have all the needed credentials, or one of the subjects could implement a policy imposing a cyclic dependency. Therefore, it is worth defining properties that should be expressed, in the best possible way, by a strategy:

- A strategy should bring a negotiation to success, when such a possibility exists. Strategy having such a property are said to be complete.
- Ideally, a strategy should avoid the release of information which is not strictly required to bring the negotiation to an end.
- A strategy should truncate a negotiation when it cannot bring to a successful conclusion.
- A strategy should recognize a cyclic dependency among credentials and policies.
- The strategy should be reasonably efficient.

There is a vast choice of possible negotiation strategies, each one with its peculiar features. An important distinction can be drawn upon the level of prudence in the disclosure of credentials and policies. In [2] and [7], the following strategies are considered:

Eager Strategy. This strategy is complete and efficient. Participants release all their credentials as soon as the relevant policy is satisfied, without waiting the credential to be requested. This strategy is very simple and brings the negotiation to success whenever it is possible. Nevertheless, it reveals more credentials than those strictly needed to create the minimum level of trust.

Parsimonious Strategy. In this strategy, the number of exchanged credentials is minimized. It is reasonably efficient and it concludes with success whenever it is possible. At the beginning, parties exchange credential requests, but not the credentials themselves. All possible release sequences are then explored. The path that brings the negotiation to success with the minimum number of exposed credentials is selected and followed. Unfortunately, due to the possible limitations in the level of cooperation between two subjects, the global minimum solution is not guaranteed.

Prudent Strategy. This strategy allows establishing trust without revealing irrelevant credentials, while remaining reasonably efficient. In [9] the communication complexity is shown to be $O(n^2)$, and the computational complexity to be $O(n^m)$, where n is the number of credentials and m is the size of the policy regulating the release of credentials.

In the heterogeneous world of Internet, each entity must be free to choose the strategy that is the best compatible with its own requisites and objectives. It is quite possible that two unknown entities will choose different strategies. Thus, there is a problem of how to make such strategies interoperable, and if it is possible. In [10], a family of strategies, called DST (Disclosure Tree Strategy), is proposed as a solution to this problem. A family of strategies is defined as a set of reciprocally compatible and interoperable strategies. An important advantage regards the fact that a Security Agent can choose, among a set of strategies belonging to the same family, the closest one to its own requisites. Moreover, this way it can adopt different strategies, during the different stages of a negotiation.

III. Application of ATN to WSs

This section describes a generic trust negotiation protocol for web services. The protocol is designed in conformance to relevant standards for Web services security. Thus, it first presents an overview of these standards.

A. Standard protocols for Web services security

SOAP Web services can exploit the SOAP header as an extensible container for message metadata, which provides developers with a set of options also covering the most typical security issues. The so-called WS-* specifications are designed in order to be composed with each other. *WS-Security* supports the definition of security tokens inside SOAP messages and uses XML Security specifications to sign or encrypt those tokens or other parts of a SOAP message. It provides a level of abstraction which allows different systems, using different security technologies, to communicate securely using SOAP in a way which is independent from the underlying transport protocol. This level of abstraction allows developers to use existing security infrastructure and established industry standards for authentication, encryption and signature, but also to incorporate new security technologies. Other specifications provide additional SOAP-level security mechanisms. *WS-SecureConversatio*n defines security contexts, which can be used to secure sessions between two parties. *WS-Trust* specifies how security contexts are issued and obtained. It includes methods to issue, validate, renew and forward security tokens, to exchange policies and trust relationships between different parties. *WS-Policy* allows organizations to specify various requirements and qualities about the Web services they expose. This specification provides a general purpose model and the corresponding syntax to describe the requirements and constraints of a Web service as policies, using policy assertions. *WS-SecurityPolicy* is based on the structure of WS-Policy and allows an entity to define, through a set of policy assertions, its own security constraints and requirements. Moreover, a set policy subjects can be associated with each specified assertion. WS-SecurityPolicy allows a Web Service to define a set of assertions, and thus its own security requirements, using a standard and interoperable format [8].

Apart from WS-* specifications, additional formats and protocols are being defined by OASIS, to provide a higher level of interoperability among services. The eXtensible Access Control Markup Language (*XACML*) is a language for specifying Role- or Attribute-Based Access Control policies. The Security Assertion Markup Language (*SAML*), in particular, is an open XML-based format to convey security information associated with a principal. The generic structure of a SAML assertion is very similar to what is usually called a "digital certificate", i.e., an issuer attests some properties about a subject, defines the validity limit of the claim, and digitally signs the document to prove its authenticity and to avoid tampering. SAML itself deals with three different kinds of assertions: (*i*) authentication, (*ii*) attribute, and (*iii*) authorization decision [8].

The WS-Trust standard [11] defines mechanisms for mediating trust relations among entities in the context of Web Services. It considers a security model in which a Web service can request that a received message proves a set of claims (e.g. name, key, privileges, etc) or, more commonly, that it carries a security token representing a relation between the sender and some other entity, trusted by the service provider. In this context, a service provider can request a client, before accessing its services, to present a token released by a trusted entity. A new client would probably not possess a proper token to access the service, in advance. For this reason, WS-Trust defines a protocol for allowing a client to contact an authority, trusted by the service provider, to request the token. Such an authority is defined as a Security Token Service (STS). An STS, on his turn, can define the requirements which clients have to satisfy to obtain the release of a token. As a STS is responsible for releasing those tokens, it is also known as a "token issuer".

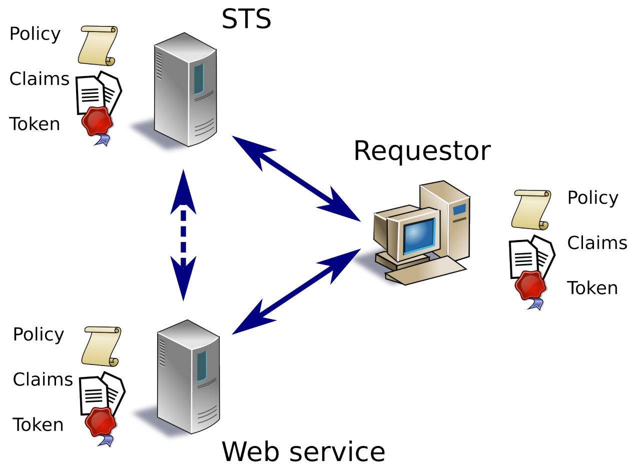


Figure 1. WS-Trust architecture

In Fig.1, arrows represent possible paths of communication among the Requestor (client), the Web service Provider, and the STS. The Requestor contacts the STS for receiving a token. The STS has the duty to verify that the Requestor possess the necessary attributes for obtaining a token. In the case if the policy of the STS is satisfied, the STS releases a token. At this point, the Requestor can send a message to the Web service Provider, attaching the obtained token.

The security token released by the STS must have some features, in particular: (*i*) being verifiable as effectively released by the STS, and (*ii*) effectively authorizing the requester to the use of some services. These features depend on the type of token being released: various technologies may be used to implement the token, such as X.509 and SAML. SAML is well fit for this scenario as it provides a secure way to make assertions about some subjects and their attributes. Otherwise these features may be guaranteed on the basis of a previous agreement, i.e., a secret, shared between the Web service and the STS bound to the service. In fact, an STS can be a platform-level Web service, bound to one or more Web services, for which it plays the role of a trusted authority. A Web service may trust the signature of the STS, or it may request an STS to validate the token, or validate it in autonomously.

A Requestor may be informed about the necessity to use a security token released by an STS, as the needed Web service can publish a policy where a certain *IssuedToken* is requested.

The interaction between a client and an STS occurs through a request-response protocol.

In particular, a *RequestSecurityToken* is used to request a token, and a *RequestSecurityTokenResponse* for responding to the request. Each request must be associated with an action which identifies the possible actions to request to an STS, as defines in the WS-Trust standard: to release, renew, cancel or validate a token. The requestor can also add claims, expressed in a certain "dialect" depending on the application. The requestor may also specify a service which the request applies to, if the STS is associated with multiple Web services; in this case, the exact endpoint reference of the Web service has to be specified.

The response may convey a token through a *RequestedSecurityToken* element. Additionally, it may convey other proofs through an *RequestedProofToken* element, containing data which the Requestor may use to demonstrate to be authorized for using the token. For example, it may contain a secret encrypted with the public key of the Requestor.

B. A Generic ATN Protocol for Web Services

An STS is normally integrated into a system using a single round of messages, i.e. a RequestSecurityToken (RST), sent from the requestor to the STS, followed by a RequestSecurityTokenResponse (RSTR), sent from the STS to the requestor. However, in some scenarios, more steps may be needed before a token is obtained. In fact, the WS-Trust standard foresees the extension of this basic mechanism, named "negotiation and challenge framework", which is depicted in Fig.2.

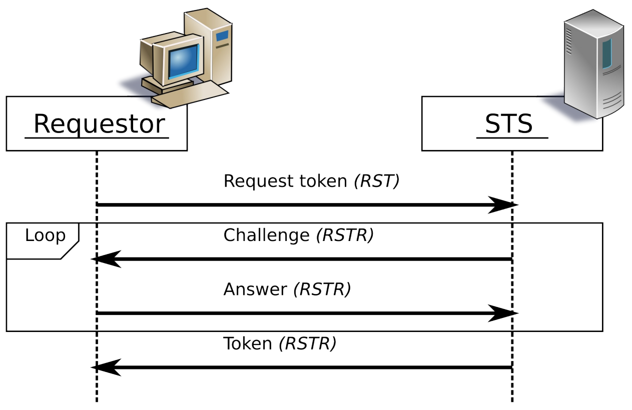


Figure 2. WS-Trust - Negotiation and challenge framework

The message exchange starts with a RST for requesting the token, then an arbitrary number of RSTR messages can be exchanged between the Requestor, or other entities, and the STS. Those RSTR messages may convey any additional information needed for completing the transaction, before finally transmitting the token. The WS-Trust standard defines some elements for proposing a "challenge" the other end, including: SignChallenge, BinaryExchange, KeyExchangeToken. However, it does not specify how to use such elements, or even other arbitrary elements, in a transaction. For example, Policy elements may be used by both parties to exchange their respective policies.

In this work, we propose a generic protocol for ATN. We decided to use some elements already proposed in [9], when possible. However, we organized the protocol and the content schemas to better distinguish the two fundamental phases of the negotiation: (*i*) the initialization, and (*ii*) the real exchange of credentials and policies.

In the initialization phase, the parties use an extensible TNInit element in a single turn of messaging. It contains information useful for defining the parameters of the following negotiation, and for verifying if there is the necessary compatibility, before beginning a real negotiation. A TNInit element can contain: a SignatureMaterial, for proving the possession of a private credential; a StrategyFamily, for identifying a supported family of strategies; a TokenFormat, for specifying the supported type of security token.

In the negotiation phase, the parties use an extensible TNExchange element. It can contain PolicyCollection and TokenCollection elements, for transporting policies and credentials, respectively, disclosed to the other party during the negotiation. Moreover, it can contain TokenType, RequestedSecurityToken and OwnershipProof, for conveying the requested token and other associated proofs.

IV. Implementation of a practical STS framework

Following the design of a generic Trust Negotiation protocol for Web services, a practical implementation has been realized. At this step, it is mainly an experimentation framework, for testing both the functionality and performance of the proposed protocol. However, part from prototype services and clients, most of its components are reusable for creating open SOA-based applications, especially in the case of dynamic service selection and composition.

The framework is available as part of the open source dDelega project [12], at https://github.com/tomamic/dDelega. dDelega is the result of ongoing work started with the development of a security layer for JADE, one the most widespread FIPA-compliant multi-agent systems [13].

In particular, it integrates a trust engine, in compliance to WS-Trust specifications. It also integrates an advanced rule engine for compliance checking against disclosure policies. These engines can be used by parties in a WS environment, by means of translator components that has been realized, in order to complete the integration. At a more basic level, the implementation exploits a number of frameworks developed under the Apache Foundation umbrella, including Axiom, Axis, Rahas, Rampart.

A. Integrating a modular trust engine

The trust engine must be able to evaluate which policies and credentials have to be inserted into the message at each round of the negotiation, on the basis of current state of negotiation and policies and credentials received at the previous round.

TrustBuilder2 (TB2) is a framework for trust negotiation, developed for providing a flexible and extensible tool in the context of research about this problem area. It is the second main version of the TrustBuilder tool and it has been developed at the DAIS (Database and Information Systems) Laboratory of the University of Illinois [14].

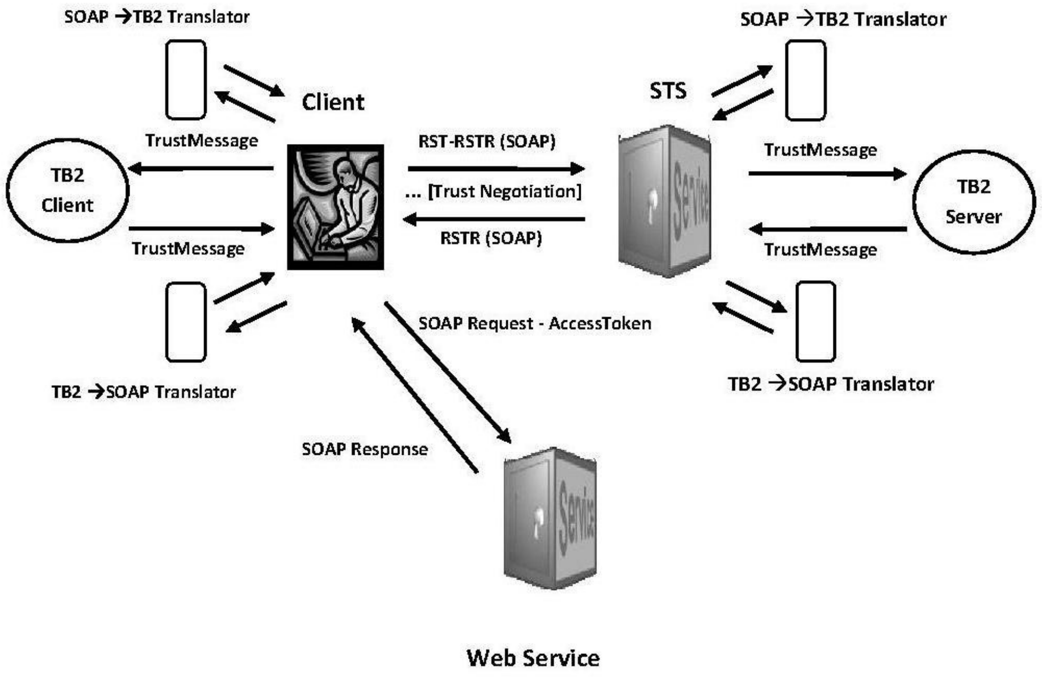


Figure 3. System architecture

TrustBuilder2 has not been realized for usage in the context of Web services, however his modular structure allows it to be extended for: (*i*) using different policy languages, (*ii*) implement different negotiation strategies, (*iii*) and provide support for different types of credentials.

In particular, after a proper translation we defined, it is able to evaluate policies expressed according to the WS-SecurityPolicy language. Starting from received policies and credentials, it is able to analyze them and take decisions about which credentials and policies to disclose, according to the chosen negotiation strategy. The framework uses a policy compliance checker, which has the duty of finding one or more minimal sets of credentials satisfying a given policy. In TB2, the main components of ATN are represented as interfaces, which can be implemented and extended to add new functionalities. They can be distinguished as:

- Strategy module: regarding negotiation strategies.
- Policy compliance checker: regarding the problem of finding a set of credentials satisfying a policy.
- Query interfaces: used to provide access to resources, including local policies and credentials.
- Credential chain module: used to build and validate chains of credentials, during the negotiation process.

TrustBuilder2 is designed according to a model of negotiation with two main phases. The first phase is characterized by the exchange of messages containing data structures, called InitBrick, for communicating the information needed to initialize a negotiation.

After this phase, the main negotiation rounds take place, characterized by the exchange of data structures called TrustMessage, i.e. objects containing policies and credentials to exchange during the negotiation.

In this research work TrustBuilder2 is used as a trust engine for automated trust negotiation. A mechanism has been realized for translating "TB2 messages", i.e. InitBrick and TrustMessage objects, into "WS-Trust messages", i.e. RSTR messages containing TNInit and TNExchange elements, which are exchanged in the context of a "negotiation and challenge framework", as defined by WS-Trust.

Policy and credentials are represented as abstract classes and credentials in TB2, in such a way to make the tool independent from the type of policies and credentials used. The authors of TB2 have also implemented the support for X.509 credentials; in fact, the implementation of this research work uses X.509 credentials. In TB2, credentials are organized in chains; i.e. when a credential, released by an authority, is sent, then the whole chain has also to be sent. In fact, TB2 does not process single credentials, but chains of credentials, through the *CredentialChainMediator* component, which uses algorithms to build and validate chains of credentials. This allows administrators to create decentralized authorities, valid for the different parties participating in a negotiation process; moreover, it allows TB2, when processing a chain, to verify the issuer of a credential released by an entity, starting from the verification of the root certificate of the chain.

Moreover, our implementation requires a user to specify, though configuration files, information about some components to be used by a client and a server, with respect to TB2 functionalities. This allows users to customize negotiation strategies, types of credentials and policy languages to be used in a certain application. The credential loader module can also be customized to load particular credentials into the system; it has access to a list of available credentials. The profile manager module uses the same customization to decide which class loader to use, according to the type of credentials used by the PolicyManager. A policy loader contains information for the PolicyManager, to decide which policy class loader to use.

B. *Using a rule engine as a policy checker*

A fundamental aspect of TB2 is the logic it uses for the functioning of its compliance checker component. In TB2, the problem of finding a set of credentials satisfying a policy is reformulated into the so-called "many pattern/many object match" problem, i.e., to find the objects matching the given patterns. Here, credentials are considered as objects and policies as patterns, in a problem which can be solved using a production rule engine. The rules of such engines have a standard format, with: an LHS (left hand side), the part of the rule defining the conditions; and an RHS (right hand side), the part of the rule defining the action to perform in the case when the conditions of the LHS are satisfied.

TB2 includes the Clouseau component, that is an expert system using the Jess (Java Expert System Shell) rule engine, which provides APIs for integration into a Java application. The rules, representing the policies of a trust negotiation process, define constraints on credentials. Jess implements the Rete algorithm [15], which allows to solve the "many pattern/many object match" problem. Using an engine of this kind in a trust negotiation process requires to introduce rules for representing policies, which specifies the patterns. The knowledge base, instead, is determined by acquired credentials. An inference can be realized by finding a set of credentials satisfying the policy, which is exactly the duty of the policy checker in TB2. Thus, a policy checker is nothing more than an expert system based on production rules.

Jess does not support natively any object for representing credentials or policies. Instead, to use credentials in Jess and to insert them into its working memory, it is necessary to define their format explicitly. Then, through JessComplianceChecker class, an assert command must be constructed and executed. This requires quite cumbersome code, for constructing thse command as an "assert(...)" string, starting from the object representing the credential.

Instead, in this work we have customized the TrustBuilder2, extending it for using a different rule engine as a policy checker. In particular, we used the *Drools* rule engine [16] for the policy checker component instead of Jess, supported by the currently available version of TB2. Drools is based on the so-called ReteOO algorithm, i.e., an adaptation of the Rete algorithm for object oriented systems. In Drools there are two main storage areas: a Production Memory, where rules are stored, and a Working Memory, where known facts are stored. For trust negotiation, the Production Memory can be used for storing the policies as rules, while the Working Memory can be used for storing the credentials as facts. An important advantage with respect to Jess is that facts in Drools are represented as Java objects, which can be put directly into the Working Memory. This has allowed us to develop a policy checker with a much leaner code than the Jess policy checker. Moreover, the tool is completely open-source, at the contrary of Jess; it is continuously updated, with the addition of new features, and it has the attentions of a vast and lively community of developers.

C. *Initial evaluation*

The ATN process, as described in the previous sections, was analyzed from the point of view of performance. The evaluation regarded the influence of the various components of the system and the conversions required by those components for communicating. For this tests, a scenario has been realized, in which:

- the client requests a token;
- the STS sends a policy requesting a chain of credentials;
- the client, on the other hand, protects one of the credentials in the chain with a policy, which he discloses to the STS;
- then, the STS discloses the credentials satisfying the client's policy;
- thus, the client discloses the credential chain initially requested by the STS;
- finally, the STS sends the requested token.

Including the initialization phase, the whole process takes 4 rounds, in which both the client and the STS send a message to the other party.

4 rounds, with Enc & Sign	6.0s
4 rounds, w/o Enc & Sign	4.8s
3 rounds, w/o Enc & Sign	4.0s
3 rounds, 1 credential requested by STS	3.0s
4 rounds, TB2, no WS	1.2s

Table 1. Initial performance results

As shown in Tab.1, the execution times vary around a mean value of 6 seconds, including the signature and encryption of SOAP messages, and 4.75 seconds without any signature and encryption. Considering instead a minimal negotiation process of three rounds, the execution time is around 4 seconds. Decreasing the credentials required by the policy from 3 to 1, the execution time does not vary proportionally, but it is reduced only by around 1 second. This means that a significant part of the computation load is absorbed by TB2, for the evaluation of policies, in addition to the basic workload imposed by the WS-* stack [17][18].

These qualitative results are in accordance with those conducted by some authors of TB2 [19], which report that almost half of the total time of execution is used by the policy checker. Another significant comparison is with the execution of a negotiation using only the TB2 tool, in which a TB2Client and a TB2Server communicate directly, through a dedicated socket, without any conversion, signature or encryption: in the same scenario with 4 rounds, as described above, the process takes 1.2s in TB2, against the 6s required by the whole Web services infrastructure implemented in this work.

It is worth noting that more efforts may be dedicated to the optimization and fine tuning of various components the system. Thus, performance may be improved in many aspects. For example, the inclusion of policy statements into Drools is now a process involving various steps and conversions. In future releases of the framework, this process will be streamlined, enabling a more direct inclusion of policies and improving efficiency.

V. CONCLUSIONS

This paper presented the design and implementation of a generic Trust Negotiation framework for Web services. It allows users to create trust automatically, by incrementally disclosing credentials. Modular applications can integrate services provided in an open environment, on the basis of peer-to-peer trust relationships. Interoperability among such services is guaranteed by the conformance to standard protocols for Web services. The realized ATN system is composed of various components and requires various format conversions for messages, policies and credentials. For these reasons, the complete execution of a negotiation process is quite costly and imposes a significant computational overhead. Thus, it is advisable to release tokens which can be used for accessing a number of cohesive services in a given time interval, without repeating the negotiation.

Besides using the framework in generic Web-based applications, further research work will also investigate the possibility of using an Automated Trust Negotiation protocol in distributed social platforms [20]. In fact, especially in the case of location-aware applications, unknown users may need to establish some level of trust before interacting, when meeting at a certain place or at a certain event.

In this sense, the framework described in this work will provide a solid ground for further analysis in different application scenarios, above all for its generality and modularity, which permit to exploit a powerful trust engine and a well known rule engine with very different kinds of protocols and credentials.

REFERENCES

[1] A. Poggi. Developing Ontology Based Applications with O3L. *WSEAS Trans. on Computers* 8(8): 1286-1295, 2009.

[2] Winsborough, W. H., Seamons, K. E., & Jones, V. E. (2000). Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (Vol. 1, pp. 88-102). IEEE.

[3] Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., & Yu, L. (2002). Negotiating trust in the Web. *Internet Computing, IEEE*, 6(6), 30-37.

[4] A. Negri, A. Poggi, M. Tomaiuolo, P. Turci. Dynamic Grid Tasks Composition and Distribution through Agents. *Concurrency and Computation: Practice and Experience*, 18(8):875-885, 2006.

[5] Venanzi, M., Piunti, M., Falcone, R., & Castelfranchi, C. (2011, July). Facing openness with socio-cognitive trust and categories. *In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence* – Vol. One (pp. 400-405). AAAI Press.

[6] Li, N., Mitchell, J. C., & Winsborough, W. H. (2005). Beyond proof-of-compliance: security analysis in trust management. *Journal of the ACM* (JACM), 52(3), 474-514.

[7] Yu, T., Ma, X., & Winslett, M. (2000, November). PRUNES: an efficient and complete strategy for automated trust negotiation over the Internet. In *Proceedings of the 7th ACM conference on Computer and communications security* (pp. 210-219). ACM.

[8] Bertino, E., Martino, L. D., Paci, F., & Squicciarini, A. C. (2010). Standards for web services security. In *Security for Web Services and Service-Oriented Architectures* (pp. 45-77). Springer Berlin Heidelberg.

[9] Lee, A. J., & Winslett, M. (2008, June). Towards Standards-Compliant Trust Negotiation for Web Services. In *Trust Management II: Proceedings of IFIPTM 2008* (Vol. 263, p. 311). Springer.

[10] Yu, T., Winslett, M., & Seamons, K. E. (2001). Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM conference on Computer and Communications Security* (pp. 146-155).

[11] Nadalin, A., Goodner, M., Gudgin, M., Barbir, A., & Granqvist, H. (2009). WS-Trust 1.4. *OASIS* (February 2009).

[12] Tomaiuolo, M. (2013). dDelega: Trust Management for Web Services. *International Journal of Information Security and Privacy (IJISP)*, 7(3), 53-67. ISSN:1930-1650.

[13] Poggi, A., Tomaiuolo, M., & Vitaglione, G. (2005). A Security Infrastructure for Trust Management in Multi-agent Systems. *Trusting Agents for Trusting Electronic Societies, Theory and Applications in HCI and E-Commerce*, LNCS, vol. 3577, R. Falcone, S. Barber, and M. P. Singh, Eds. Berlin, Germany: Springer, 2005, pp. 162-179.

[14] Lee, A. J., Winslett, M., & Perano, K. J. (2009). Trustbuilder2: A reconfigurable framework for trust negotiation. In *Trust Management III* (pp. 176-195). Springer Berlin Heidelberg.

[15] Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1), 17-37.

[16] Sottara, D., Mello, P., & Proctor, M. (2010). A Configurable Rete-OO Engine for Reasoning with Different Types of Imperfect Information. *IEEE Transactions on Knowledge and Data Engineering*, 22(11), 1535-1548.

[17] Novakouski, M., Simanta, S., Peterson, G., Morris, E., & Lewis, G. (2010). Performance analysis of ws-security mechanisms in soap-based web services (No. CMU/SEI9-2010-TR-023). *Carnegie-Mellon University, Software Engineering Institute*.

[18] Rodrigues, D., Pigatto, D. F., Estrella, J. C., & Branco, K. R. (2011). Performance evaluation of security techniques in web services. In *Proc. of the 13th International Conference on Information Integration and Web-based Applications and Services* (pp. 270-277). ACM.

[19] Lee, A. J. (2008). *Towards practical and secure decentralized attribute-based authorization systems*. ProQuest.

[20] Franchi, E., Poggi, A., Tomaiuolo, M. (2013). Supporting Social Networks with Agent-Based Services. *International Journal of Virtual Communities and Social Networking (IJVCSN)*, 5(1), 62-74. ISSN:1942-9010.

2COMM: A commitment-based MAS architecture

Matteo Baldoni, Cristina Baroglio and Federico Capuzzimati
Dipartimento di Informatica
Università degli Studi di Torino
c.so Svizzera 185, I-10149 Torino (Italy)
Email: {matteo.baldoni,cristina.baroglio,federico.capuzzimati}@unito.it

***Abstract*—Social expectations and social dependencies are a key characteristic of interaction, which should be explicitly accounted for by the agent platform, supporting the coordination of the involved autonomous peers. To this aim, it is necessary to provide a normative characterization of coordination and give a social meaning to the agents' actions. We focus on one of the best-known agent platforms, Jade, and show that it is possible to account for the social layer of interaction by exploiting commitment-based protocols, by modifying the Jade Methodology so as to include the new features in a seamless way, and by relying on the notion of artifact, along the direction outlined in the Mercurio proposal.**

This is a light revision of a paper presented a EMAS 2013.

I. Introduction and Motivation

Interaction creates *social expectations* and *dependencies* in the involved partners [38], [18], [34], [23]. These should be explicitly accounted for by the *agent platform* to allow the coordination of autonomous entities. In order to create social expectations on the agents' behavior, it is necessary to introduce a *normative* characterization of coordination and give a social meaning to the agents' actions. An agent that understands such a specification and that publicly accepts it (i.e. that declares it will behave according to it) allows reasoning about its behavior [21]. This is the key to the development of open environment systems, made of autonomous and heterogeneous components.

By not supplying such abstractions, current platforms do not supply agents the means for *observing* or *reasoning* about the social expectations involved by interaction, and do not supply the designers the means to explicitly *express* and *characterize* them when developing an interaction model. An important example is JADE [10], [11], which is a well-established development environment for multi-agent systems, FIPA-compliant, actually used for industrial applications, and which notoriously does not provide of a social and observational semantics.

One proposal for filling this gap is provided by the Mercurio framework [4], [3], where *commitments* and *commitment-based protocols*, which are well-known for featuring a social and observational semantics [34], [35], [41], are introduced in JADE. Mercurio, however, is an abstract proposal. In this work, we describe an implementation of a system along the lines proposed in Mercurio. Our starting point for introducing commitment-based protocols inside JADE is the JADE Methodology [30]. This methodology is particularly interesting because it is intrinsically agent-oriented – it is not the adaptation of an object-oriented methodology, and it combines a top-down approach with a bottom-up one, allowing the integration with legacy, non agent-based systems. It concerns two of the four main phases of the standard software development cycle: the *analysis phase* and the *design phase*.

Following [4], we rely on a form of indirect communication among agents that envisages the use of *artifacts*: commitment-based communication artifacts implement interaction protocols as well as monitoring functionalities for the verification that the on-going interaction respects the protocol, for detecting violations and violators, and so forth. Artifacts, therefore, encode the social layer of the multi-agent system: as a programmable communication channel an artifact contains what in the terminology of commitment protocols is called "the social state", and captures it as an interaction session among the parties. Artifacts also supply agents the social actions that are necessary to the interaction – that is, actions that allow agents to enter into and to comply with commitments – together with their social meaning. As a consequence, they capture the coordination rules of the protocol. The reification of commitment protocols allows agents to act on them, e.g. to examine them (for instance, to decide whether to play one of the foreseen roles), use them (which entails that they explicitly accept the corresponding regulation), negotiate their construction, specialize them, and compose them. The advantage of relying on indirect communication is that it allows more variegated ways of interacting, not hindering message exchange when necessary.

In this paper we show that our proposal can be integrated *seamlessly* within the JADE Methodology, simply by substituting the selection of JADE FIPA protocols with the selection/construction of appropriate communication artifacts. We also use the methodology to show the differences between these two alternatives with the help of an example from a financial setting.

Section II reports the relevant background, necessary to understand the proposal. Section III is the core of the paper, containing the original proposal. Section IV applies the concepts to an illustrative example, from a financial setting. A discussion also involving related works ends the paper.

II. Background

We briefly report the technical, methodological and theoretical background required for our work. We use the proposal in [4] as a high-level reference architecture. In this work, the authors outline the basic ideas for an interaction-oriented agent framework, grounding the social semantics of interaction on commitments, and proposing the A&A (Agents and Artifacts) Metamodel as a means to obtain a form of indirect, observable communication. Let us, then, explain the fundamental bricks to build our architecture, whose overview is reported in Figure 1.

a) JADE framework.: JADE is a popular and industry adopted agent framework. It offers to developers a Java middleware 100% FIPA-compliant (Foundation for Intelligent Physical Agents, [1]) plus a set of command-line and graphical tools, supporting development and debugging/testing activities. Its robustness and well-proven reliability makes JADE a preferred choice in developing MAS. It is currently used in many research and industrial projects jointly with its most popular and promising extension, WADE [17]. A JADE-based system is composed of one or more *containers*, each grouping a set of agents in a logical *node* and representing a single JADE runtime. The overall set of containers is called a *platform*, and can spread across various physical hosts. The resulting architecture hides the underlying layer, allowing support for different low-level frameworks (JEE, JSE, JME, etc.). The platform reference container is called *main container*, and represents the entry point to the system. JADE provides communication and infrastructure services, allowing agents, deployed in different containers, to discover and interact with each other, in a transparent way from the developer's logical point of view.

b) Commitment Protocols.: Agents share a social state that contains commitments and other literals that are relevant to their interaction. A *commitment* $C(x, y, r, p)$ denotes a contractual relationship between a debtor x and a creditor y: x commits to y to bring about the consequent condition p when the antecedent condition r holds. A commitment, when active, functions as a directed obligation from a debtor to a creditor. However, unlike a traditional obligation, a commitment may be manipulated, e.g., delegated, assigned, or released [37]. Importantly, commitments have a regulative value: the social expectation is that agents respect the commitments which involve them and, in particular, the debtor is considered responsible of realizing the consequent condition. Thus, the agents' behavior is affected by the commitments that are present in the social state. A *commitment protocol* usually consists of a set of actions, whose semantics is shared (and agreed upon) by all of the interacting agents [41], [40], [20]. The semantics of the social actions is given in terms of operations which modify the social state by, e.g., adding a new commitment, releasing another agent from some commitment, satisfying a commitment, see [41].

c) CArtAgO.: CArtAgO is a framework based on the A&A model. It extends the agent programming paradigm with the first-class entity of *artifact*: a resource that an agent can use, and that models working environments ([32]). In order to properly model a MAS, CArtAgO proposes to explicitly model the environment where pro-active agents live, work, act and communicate. It provides a way to define and organize *workspaces*, logical groups of artifacts, that can be joined by agents at runtime and where agents can create, use, share and compose artifacts to support individual and collective, cooperative or antagonistic activities. The environment is itself programmable as a dynamic first class abstraction, it is an active part of a MAS, encapsulating services and functionalities. The A&A model decouples the notion of agent from the notion of environment. The overall engineering of the MAS results more flexible, easy to understand, modular and reusable. CArtAgO provides an API to program *artifacts* that agents can use, regardless of the agent programming language or the agent framework used. This is possible by means of the *agent body* metaphor: CArtAgO provides a native agent entity, which allows using the framework as a complete MAS platform as well as it allows mapping the agents of some platform onto the CArtAgO agents, which, in this way, becomes a kind of "proxy" in the artifacts workspace. The developed agent is the mind, that uses the CArtAgO agent as a body, interacting with artifacts and sensing the environment. An agent interacts with an artifact by means of public *operations*. An operation can be equipped with a *guard*: a condition that must hold so that the operation will produce its effects. It is not an execution condition: when the guard does not hold the action is performed anyhow but without consequences.

III. Reifying Commitment Protocols with Artifacts

Artifacts naturally lend themselves to provide a suitable means for realizing mediated communication channels among agents. To this aim, it is necessary to encode inside a communication artifact a normative characterization to the actions it offers to agents and that allow them to interact. We propose to interpret commitment protocols as environments, within which agents interact. The public interface of artifacts allows agents to examine the encoded interaction protocol. As a consequence, the act of using an artifact can be interpreted as a declaration of acceptance of the coordination rules. This will generate social expectations about the agent's behavior and agrees with the characterization of norms in [21]. Moreover, the fact that the behavior of agents on artifacts is observable and that interaction only occurs through artifacts, agrees with the view that regulations can only concern observable behavior [22]. The resulting programmable environment provides a flexible communication channel that is suitable for realizing open systems. Notice that the use of a programmable environment does not entail that the social state will be centralized because an artifact can be composed by a distributed network of artifacts.

Figure 1 sketches the way in which we propose to use CArtAgO so as to account also for social commitments inside JADE. We named this first realization of the Mercurio architecture *2COMM* (standing for "Communication & Commitment")[1]. 2COMM realizes mediated interaction by means of communication artifacts, which, in our proposal, replace the JADE-based FIPA protocols and which reify commitment-based protocols [4]. At the bottom level, the JADE framework supplies standard agent services: message passing, distributed containers, naming and yellow pages services, agent mobility. When needed, an agent can *enact* a certain protocol role, thus using a communication artifact by CArtAgO. This provides a set of operations by means of which agents participate in a mediated interaction session. Each artifact (protocol enactment) maintains a *social state*, that is, a collection of *social facts* and *commitments* involving the roles of the corresponding protocol, following Yolum and Singh's commitment protocol model [40].

[1]The source files of the system and examples are available at the URL http://di.unito.it/2COMM.

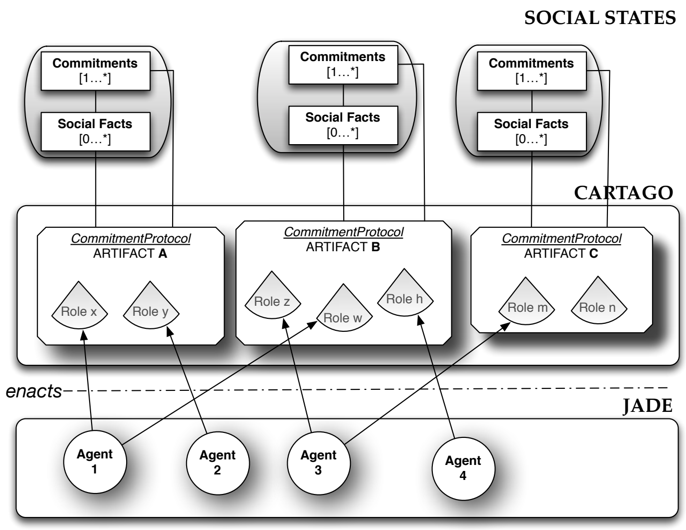


Fig. 1. A sketch of 2COMM.

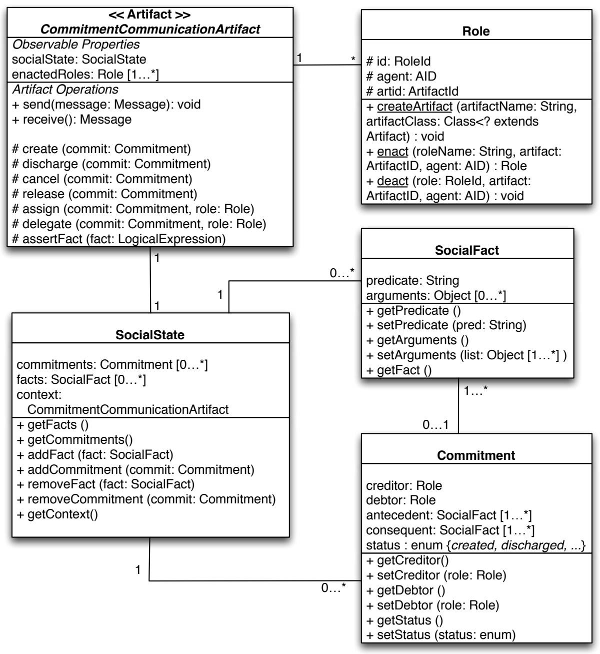

<< Artifact >>
CommitmentCommunicationArtifact
Observable Properties
socialState: SocialState
enactedRoles: Role [1…*]
Artifact Operations
+ send(message: Message): void
+ receive(): Message
# create (commit: Commitment)
# discharge (commit: Commitment)
# cancel (commit: Commitment)
# release (commit: Commitment)
# assign (commit: Commitment, role: Role)
# delegate (commit: Commitment, role: Role)
# assertFact (fact: LogicalExpression)
Role
# id: RoleId
# agent: AID
# artid: ArtifactId
+ createArtifact (artifactName: String, artifactClass: Class<? extends Artifact) : void
+ enact (roleName: String, artifact: ArtifactID, agent: AID) : Role
+ deact (role: RoleId, artifact: ArtifactID, agent: AID) : void
SocialFact
predicate: String
arguments: Object [0…*]
+ getPredicate ()
+ setPredicate (pred: String)
+ getArguments ()
+ setArguments (list: Object [1…*])
+ getFact ()
SocialState
commitments: Commitment [0…*]
facts: SocialFact [0…*]
context: CommitmentCommunicationArtifact
+ getFacts ()
+ getCommitments()
+ addFact (fact: SocialFact)
+ addCommitment (commit: Commitment)
+ removeFact (fact: SocialFact)
+ removeCommitment (commit: Commitment)
+ getContext()
Commitment
creditor: Role
debtor: Role
antecedent: SocialFact [1…*]
consequent: SocialFact [1…*]
status : enum {created, discharged, ...}
+ getCreditor()
+ setCreditor (role: Role)
+ getDebtor ()
+ setDebtor (role: Role)
+ getStatus ()
+ setStatus (status: enum)
1
*
1
1
1
0…*
1…*
0…1
1
0…*

Fig. 2. The UML Class diagram for the core of 2COMM.

A. Communication Artifact

We follow the ontological model for organizational roles proposed in [13], [14], which is characterized by three aspects: (1) *Foundation*: a role must always be associated with an institution it belongs to and with its player; (2) *Definitional dependence*: the definition of the role must be given inside the definition of the institution it belongs to; (3) *Institutional empowerment*: the actions defined for the role in the definition of the institution have access to the state of the institution and of the other roles, thus, they are called *powers*; instead, the actions that a player must offer for playing a role are called *requirements*.

Communication artifacts realize a kind of mediated interaction that is guided by commitment-based protocols. Figure 2 shows the UML schema of the super-type of communication artifacts implementing specific interaction protocols (e.g., Contract Net, Net Bill, Brokering): the *CommitmentCommunicationArtifact*. We call an instance of an artifact of type CommitmentCommunicationArtifact an *interaction session*. It represents an on-going protocol interaction, with a specific social state that is observable by the interacting agents, that play the protocol roles. The CommitmentCommunicationArtifact presents an observable property, *enactedRoles*, that is the collection of the roles of the protocol (definitional dependence [13], [14]). Actions have a social effect only when they are executed by the role they are assigned to, but actions are not defined at this super level, rather they are provided by the instantiations of the CommitmentCommunicationArtifact, i.e. by artifacts implementing specific protocols. Each protocol action is implemented as a public *operation*, which is associated to a specific role (institutional empowerment [13], [14]): the fact that the action was executed is registered in the social state together with its meaning. An action can have some additional guards, implementing *context preconditions*: such a condition specifies the context in which it makes sense that the action produces the described social effect. An artifact can be monitored by an observer agent, who, following the CArtAgO terminology, is *focusing* on that artifact, particularly on one or more public properties. A change of one of these properties causes a *signal*, from the artifact to the observer agents, about the property that changed: the agents *perceive* the new artifact state. In particular, when the creation of a commitment, involving an agent as a debtor, is signaled to it, this agent is expected to behave so as to satisfy the commitment. The agent is free to decide how (and if) it will handle the satisfaction of its commitments. Therefore, the requirement is that an agent has the capability to behave so as to achieve the involved conditions [13], [14]. An agent who does not show such capabilities is bound to violate its commitments.

CommitmentCommunicationArtifact provides a property, tracking the identity of the agents actually playing the various role. Two operations are provided, by class *Role*, in order to manage the association between an agent's identity and a role: *enact* and *deact*, by means of which an agent can explicitly assume/cease a protocol role (foundation [13], [14]). After enacting a role, the use of the associated operations on the artifact will have social consequences.

The communication artifact has an observable property, *social state*, that is a set of zero or more elements of type *Commitment* or *Social Fact*. As we can see in Figure 2, these structures are simple Java objects, representing the actual social state. The artifact is responsible to manage the Social State structure, i.e. the Commitments life-cycle, as well as the assertion or retraction of social facts, via methods called on commitment and on social fact objects. For Commitment management, we refer to the basic operations of commitment manipulation [40]: *create, discharge, cancel, release, assign, delegate*. The operations regarding the commitments life-cycle are implemented as artifact *internal operations*, therefore, the agents cannot modify commitments explicitly. The communication artifact exposes the social state, whose evolution is controlled by the agents via the protocol-provided actions. Finally, communication artifacts provide service operations, which can be performed only by the ArtifactManager Agent (see below) for managing the protocol roles and the identities of their players.

When the social state property changes, due to the execution of a protocol action (an artifact operation) on the communication artifact, all of the agents using the artifact will be notified, allowing them to react (or not) to the evolution of the interaction. This mechanism is a core part of the CArtAgO framework.

The *ArtifactManager Agent* plays the role of a Yellow Pages Agent for communication artifacts, or, in other terms, of an artifact broker. It has a crucial role: it is a "communication channel" broker, gathering requests for both focused or broadcasting calls for interaction. As such, it provides a collection of utility services. It supplies information about the interaction protocols (e.g. it provides the XML describing a given protocol, it allows a search for a protocol, a list of active communication channels, a list of interacting agents); it answers to requests about the status of an existing interaction session; it notifies the subscriber agents a particular session availability, and so on. Its main purpose is to prepare the communication artifact among the interacting agents, and to supply it to the requesting agents. It can also enable other interested agents to monitor, audit, or, more generally, observe the social state evolution. The communications between the ArtifactManager Agent and the requesting agents is realized via FIPA-ACL messages: when a requester sends a request ACL message to the ArtifactManager Agent, specifying the protocol and the role it wants to enact, the latter will do the following steps:

1) Check if the requested protocol is available;
2) Check if the requested role is foreseen by the protocol;
3) Create/retrieve a communication artifact of the requested type;
4) Set the requested artifact role field to the agent identifier (AID) of the requester;
5) Respond to the requester with the artifact's reference;
6) Possibly inform other interested agents of the availability of the communication artifact.

The initialization procedure is modeled as a simple FIPA Request Interaction Protocol, where the content of messages consists of the communication artifact request parameters. After this phase, the agent can use the *enact* operation to start playing the requested role. The use of an agent does not necessarily imply a centralization of the yellow pages: agents may directly create communication artifacts; yellow pages can be federated.

B. Using Mediated Communication at Design Time

We assume that MAS designers know a collection of communication artifacts, each representing a commitment-based protocol. Each protocol is enriched with an XML-based description of it, a *Protocol Manual*, available both at design- and at run-time. It is an add-on to the CArtAgO artifact manual, with orthogonal scopes and purposes. It can be used by MAS and agent designers as a guideline for understanding whether an agent is suitable for a protocol role as well as for understanding whether a protocol role suits the purposes of an agent. From a methodological point of view, the designer needs the Protocol Manual to know the social consequences of the actions supplied by an artifact, in terms of social facts and commitments, so he/she can design agent behaviors accordingly. Then, depending on the implemented behavior, the agent will decide how to use information about the social state evolution, how to fulfill commitments, which social action (i.e. a public artifact operation) to execute and when. Ideally, the designer should equip the agent with the behaviors that are necessary to bring about the conditions of the commitments it will possibly take. This protocol-centric design, jointly with the commitment nature of protocols, avoids a critical facet of JADE protocols. Here, a pattern of interaction is projected on a set of JADE behaviors, one for each role, thus making a global view of the protocol and its maintenance difficult, and binding the very interaction to ad-hoc behaviors. Consequently, the risk of conflicting behaviors, not devised at design time, increases. This way, the designer can leverage a library of programmable communication artifacts, focusing on the internal agent behavior without being concerned about ad-hoc shaped communication behaviors.

IV. JADE METHODOLOGY REVISED

The JADE Methodology is a JADE founded agent-oriented software engineering methodology. It proposes a fully agent-based approach, instead of adapting Object-Oriented techniques (like MASE [39], Adelfe [12] or MESSAGE [16]). It concerns the *analysis* and the *design* phases of the software development life cycle. The methodology considers agents as "pieces of autonomous code, able to communicate with each other" [30], thus following a weak notion of agency; it does not account for mentalistic/humanistic agents properties.

In the analysis phase, the first step is the identification of *use cases*, i.e. functional requirements of the overall system, which are captured as standard Use-cases UML Diagrams. Starting from this, the designer can point out an initial set of agent types: an agent type for each user/device and for each resource. The agent paradigm foresees that even external devices and software/hardware resources (e.g. legacy systems, databases, external data sources) are represented with an agent. The designer, then, identifies *responsibilities*, i.e. the activities provided by system each agent is responsible for; and *acquaintances*, that is relationships between agents aimed at fulfilling some responsibility. The results are a *Responsibility table* and an *Agent diagram* with initial acquaintances. No distinction is made between acquaintances and responsibilities: in fact, the mentioned table will contain both. The analysis is completed by executing activities related to agents/acquaintances refinement, to define discover services and to add management/deployment information. The design phase starts with the *interaction specification* step, where an interaction table is produced. It refers to the responsibility table in order to define interactions between JADE agents, specifying the interacting agents, the protocol and protocol role (e.g. Initiator or Responder), the reference responsibility, and a triggering condition.

It is suggested to use, when possible, standard JADE protocol behaviors, that must be added to an agent's behavior set to implement the corresponding protocol role. The subsequent steps focus on the specification of agent interactions with users and resources; the definition of a yellow page services, using the JADE Directory Facilitator; the implementation of agent behaviors, starting from JADE protocol behaviors related to responsibilities. A last effort is the definition of a shared, system-wide ontology.

We show how it is possible to integrate, within the JADE Methodology [30], an account of commitment-based protocols with the help of a real-world scenario, we call *FinancialMAS*. For brevity, we show only the fundamental steps needed to draft the system and to highlight the benefits of reifying commitment-based protocols by means of artifacts, and thus based on mediated interaction. By applying the steps of the methodology, we obtained an initial design prototype for FinancialMAS, concerning an initial set of agents and the so called *responsibility table* (Table I). In the terminology of the JADE Methodology, *responsibilities* amount to functional duties, agents are responsible for, from an overall MAS point of view. To handle them, agents possibly need to *interact* with one another. The result of this analysis is an *Interaction table* (Table II). At this point, instead of realizing protocols via distributed JADE behaviors, we implement them via *commitment-based communication artifacts*. We assume to have already designed artifacts for common interaction protocols, like the Contract Net Protocol, the Query Protocol, and the Request Protocol. The resulting model is depicted in Figure 3. For the sake of comparison, in Figure 4 we zoomed into the one of the commitment artifacts, the *Contract Net Protocol* artifact, reported as a UML diagram, while in Figure 5 we highlight the very same protocol, implemented via pure JADE behaviors.

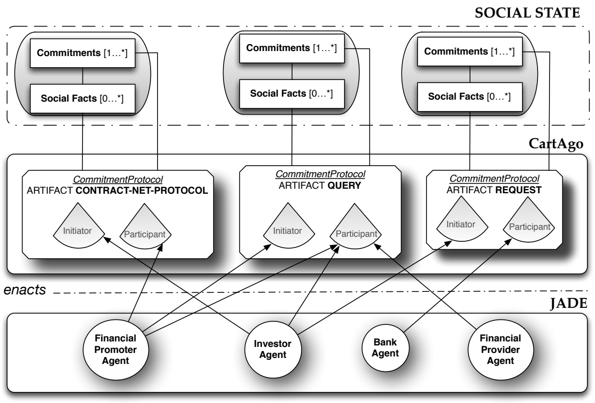


Fig. 3. FinancialMAS Commitment-based Interaction Architecture.

2COMM proposes a clear notion of *Role* that an agent must enact to participate in an interaction session, so the designer must only implement the behaviors for fulfilling the commitments caused by the execution of a protocol *actions*. We refer to the following description of CNP based on commitment protocols (this is just an example, alternatives and variants can be found in papers like [42], [24]):

cfp **means** create(*C(i, p, propose, accept* $\vee$ *reject)*)
accept **means** *none*
reject **means** release(*C(p, i, accept, done* $\vee$ *failure)*)

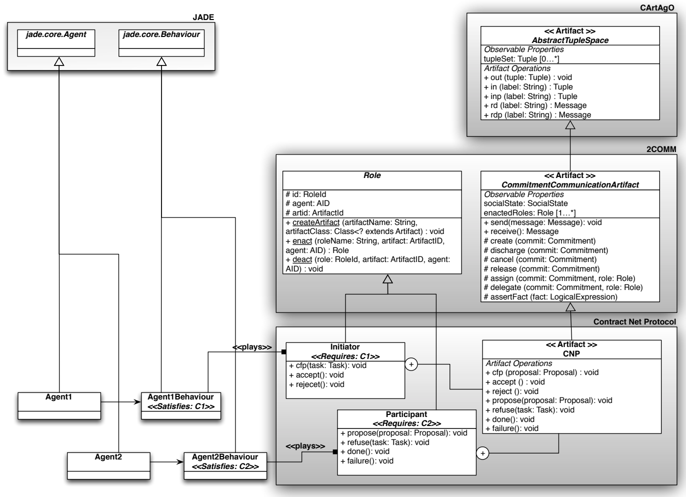

JADE
jade.core.Agent
jade.core.Behaviour
CArtAgO
<< Artifact >>
AbstractTupleSpace
Observable Properties
tupleSet: Tuple [0…*]
Artifact Operations
+ out (tuple: Tuple) : void
+ in (label: String) : Tuple
+ inp (label: String) : Tuple
+ rd (label: String) : Message
+ rdp (label: String) : Message
2COMM
Role
# id: RoleId
# agent: AID
# artid: ArtifactId
+ createArtifact (artifactName: String, artifactClass: Class<? extends Artifact) : void
+ enact (roleName: String, artifact: ArtifactID, agent: AID) : Role
+ deact (role: RoleId, artifact: ArtifactID, agent: AID) : void
<< Artifact >>
CommitmentCommunicationArtifact
Observable Properties
socialState: SocialState
enactedRoles: Role [1…*]
+ send(message: Message): void
+ receive(): Message
# create (commit: Commitment)
# discharge (commit: Commitment)
# cancel (commit: Commitment)
# release (commit: Commitment)
# assign (commit: Commitment, role: Role)
# delegate (commit: Commitment, role: Role)
# assertFact (fact: LogicalExpression)
Contract Net Protocol
<<plays>>
Initiator
<<Requires: C1>>
+ cfp(task: Task): void
+ accept(): void
+ rejecet(): void
<< Artifact >>
CNP
Artifact Operations
+ cfp (proposal: Proposal) : void
+ accept () : void
+ reject (): void
+ propose(proposal: Proposal): void
+ refuse(task: Task): void
+ done(): void
+ failure(): void
Agent1
Agent1Behaviour
<<Satisfies: C1>>
Participant
<<Requires: C2>>
+ propose(proposal: Proposal): void
+ refuse(task: Task): void
+ done(): void
+ failure(): void
<<plays>>
Agent2
Agent2Behaviour
<<Satisfies: C2>>

Fig. 4. The UML diagram for the 2COMM implementation of CNP.

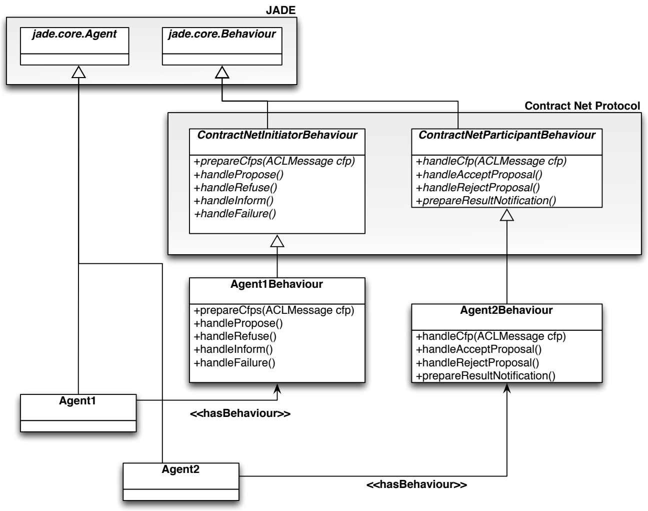


Fig. 5. UML diagram for the JADE implementation of CNP.

propose **means** create(*C(p, i, accept, done* $\vee$ *failure)*)
refuse **means** release(*C(i, p, propose, accept* $\vee$ *reject)*)
done **means** *none*
failure **means** *none*

In the case of CNP, two roles are foreseen, *Initiator* (i) and *Participant* (p). Playing a role gives an agent *powers*, in terms of social state modification (i.e. the state of the interaction session) as a consequence of its actions, and the agent designer can use them if, when and how he/she wants. For instance, for what concerns the update of the social state, when an agent playing the role Initiator executes the artifact action *cfp*, the social state is modified by creating the commitment *C(i, p, propose, accept* $\vee$ *reject)*. On the one hand, this change binds i to either accept or reject a proposal, if one is received; the agent is free to decide not only which course of action to take but also how to realize acceptance or rejection. On the other hand, this change is signaled to the agent playing the role

TABLE I. RESPONSIBILITY TABLE FOR FINANCIALMAS.

Agent Type	No.	Responsibility
Investor agent (IA)	1	Let investor search for investments proposals
	2	Assist investor in setting search parameters and data
	3	Support the individuation of the investor's risk profile
	4	Support in proposal acceptance
	5	Withdraw from an investment contract
Financial Promoter agent (FP)	1	Respond to investment searches
	2	Assist financial promoter in risk-classifying financial products
	3	Determine the investor's profile
	4	Support individuation of the investor's risk profile
Bank agent (BA)	1	Support bank in investment contract subscription
	2	Assist bank in investment conclusion
Financial Provider agent (FV)	1	Provide financial and aggregate news information
Integration agent (IntA)	1	Serve and support integration with legacy bank informative systems

TABLE II. INTERACTION TABLE FOR FINANCIALMAS: WHO INTERACTS WITH WHOM, TO FULFILL WHICH DUTY, BY USING WHICH PROTOCOL.

Interaction	R.ty	Interaction Protocol	Role	With	When
Investor Agent					
Search Investment	1	CNP	Initiator	FP	Investor searches an investment
Profiling	3	Query	Participant	FP	Investor chose a Financial Promoter
Proposal Acceptance	4	Query	Participant	BA	Investor chose a financial product
Withdraw	5	Request	Initiator	BA	After Investor accepted a proposal
Financial Promoter Agent					
Respond to Search	1	CNP	Participant	IA	Investor searches an investment
Profiling	3	Query	Initiator	IA	Investor chose a Financial Promoter
Fin. Prod. Classif.	2	Query	Initiator	FV	FP starts fin. prod. classif.
Bank Agent					
Proposal Acceptance	1	Query	Initiator	IA	Investor chose a financial product
Withdraw	3	Request	Participant	IA	After Investor accepted a proposal
Financial Provider Agent					
Fin. Prod. Classif.	1	Query	Participant	FP	FP starts fin. prod. classif.

Participant, who will handle it in some manner (depending on its behaviors) and decide whether sending a proposal. Instead, when a *accept* is executed the raised event automatically discharges a commitment created by a *cfp*.

This approach is illustrated in Figure 4. We modeled CNP as a Commitment Communication Artifact. Roles are *inner classes* within the artifacts, allowing JADE agents to use them. The protocol consists of a set of *social actions*, each of which has both an impact on the social state of the interaction and on the communication between agents. Actions are attributed to roles. For instance, action *cfp* is attributed to the role *Initiator*. For what concerns communication, the execution of a social action amounts to sending the content to be communicated through the tuple space provided by CArtAgO. This result is obtained by exploiting the method *send* of the CommitmentCommunicationArtifact. Commitments are handled as an instance of the class SocialState which is part of the CommitmentCommunicationArtifact. For example, consider the social action *cfp*, whose execution creates a commitment. This result is achieved through the execution of the following artifact operation:

```
@OPERATION
public void cfp(Task task, Role initiator,
 Role participant) {
    Message cfp = new Message();
   // setting of cfp parameters
    send(cfp);
    create(new Commitment(initiator,
    participant,
    new Fact(``propose''),
    new CompositeExpression(
         LogicalOperatorType.OR,
                          new Fact(``accept''),
                          new Fact(``reject'')))));
}
```

The first part of the operation manages the communication level, while the latter manages the creation of the commitment. The action *cfp* attributed to the role Initiator merely calls the described artifact operation.

An agent that will to play as a certain role can inspect the commitments that are required by the role itself, which are the commitments it will possibly be involved in as a debtor. In order to be able to satisfy them, the agent needs to have appropriate behaviors, otherwise its role execution is bound to fail. Notice that the agent is autonomous in selecting which social actions to execute and when as well as how to behave in order to satisfy its commitments.

Looking at Figure 5, the reader can perceive a major drawback of the original JADE approach: being part of an interaction protocol entails the adoption of an entire behavior, that must be added to the set of the internal agent behaviors. The resulting agent design breaks the autonomy of the agent, since the agent has an additional behavior for each role of each interaction it takes part to, increasing the possibility of conflicts between behaviors, and increasing the overall agent design complexity. In fact, being such behaviors FSMBehaviors, they implement Finite State Machines, i.e. they rigidly prescribe the sequences of actions that the agent is allowed to execute without any flexibility. Thus, it is not possible to intervene on the logic by which actions are sequentialized but only to realize the methods that the predefined behavior requires to redefine, which roughly correspond to decision points. Furthermore, this approach hinders the observability of the interaction, unless the designer adds specific sniffing or audit agents to log every message passed. In performance-critical applications, having more agents and producing a message overhead can produce undesirable scenarios.

V. Related works, discussion and future work

2COMM is a first step towards the implementation of the Mercurio architecture, proposed in [3], [4]. It realizes a programmable communication channel by means of artifacts, which is interaction-centric, exploits the social meaning of interaction supplied by commitment protocols, and enables the development of monitoring functionalities. The realization of roles is inspired by [8], [9]. The use of commitments gives a normative value to the encoded protocol, while the act of using a communication artifact amounts to the explicit acceptance, by the agent, of the rules of the protocol. This makes the current proposal very different from [7], whose aim was the introduction of the notion of role, as in [8], [9], inside JADE. The proposal conjugates the flexibility and the openness that are typical of MAS with the need of modularity and compositionality that are typical of design and development methodologies. The realization of commitment protocols as artifacts is an advancement of research on commitment-based approaches, w.r.t. approaches like [19], where commitment management resides in a middleware which, in turn, relies on a message-exchange communication infrastructure. Even though the function of the middleware recalls that of our artifacts, artifacts are, by their nature, distributed (and not centralized), they can be the result of the composition of other artifacts, can be manipulated and customized by the agents themselves. Moreover, the adoption of tuple spaces allows more variegated forms of communication where communication actions are not limited to utterances.

We believe that a commitment approach brings relevant advantages in terms of design and modeling flexibility, modularity and traceability. The resulting artifact explicitly provides a notion of *Role* that is decoupled from the interacting agent, instead of cabling it into an agent behavior (as in the JADE Methodology) or of composing different atomic roles to build an agent type (as in the GAIA Methodology [43]). Both approaches break into inner agent definitions, hindering the agent autonomy and the openness of the system. The artifact entity supplies a natural way for logging and audit purposes, leveraging the concept of social state (and its evolution). In a pure agent environment (like JADE), a similar result is obtained via a massive use of either message-sniffing agents and/or auditing agents, with a consequent overhead of the number of messages that are passed. This is, for example, the case of the proposal in [29]. By being an observable property, the social state provides the agent society a clear vision of who is responsible of what, in which protocol interaction, and when an agent acted so as to fulfill its commitments.

2COMM focuses on the interaction protocol layer, leaving aside issues concerning the society of agents in which the interaction takes place. Thus, it does not, for instance, tackle how to deal with violations of commitments. In order to properly handle these aspects it would be interesting to combine its use with proposals from the area of e-institutions. Concerning this field 2COMM would provide an improvement in that it would introduce the possibility to account for indirect forms of communication. As [25] witness, there is an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts, whose use is not always natural. Along this direction, it is relevant to mention the OCeAN meta-model for artificial institutions [26], which encompasses a notion of commitment, and for which a possible architecture is discussed in [31]. For what concerns organizations, instead, there are some attempts to integrate them with artifacts, e.g. ORA4MAS [27] and JaCaMo http://jacamo.sourceforge.net, which also accounts for BDI agents. Following the A&A perspective, artifacts are concrete bricks used to structure the agents' world: part of which is the organizational infrastructure, part amounts to artifacts introduced by specific MAS applications, including entities/services belonging to the external environment. In [27] the organizational infrastructure is based on $Moise^+$, which allows both for the enforcement and the regimentation of the rules of the organization. This is done by defining a set of conditions to be achieved and the roles that are permitted or obliged to perform them. The limit of this approach is that it cannot capture contexts in which regulations are, more generally, norms because norms cannot be restricted to achievement goals. Recently, the use of a communication infrastructure based on artifacts has been proposed to define, in an explicit and clear way, interaction in JaCaMo [33]. Nevertheless, the proposal does not supply a normative account of communication.

Finally, we think that our proposal can give significant contributions in industrial applicative contexts, for the realization of business processes and, in particular, of human-oriented workflows, whose nature is intrinsically social and where the notion of commitment plays a fundamental role [28]. In [36], the authors present LoST, a commitment-based model for the definition of declarative protocols, which is based on local history vectors of sent/received messages, associated to each of the interacting agents. LoST enables the representation and monitoring of (business) protocols when it is necessary to transfer local knowledge about occurring interactions between the agents. It works as an adapter for message transfer between agents. 2COMM, instead, provides agents an environment by which they communicate and, if this is requested, they can perform actions which do not amount to utterances but still entail social effects.

As a future work, we devise an extension of 2COMM for tackling a more expressive protocol language, with support for temporal constraints, see also [2]. This goal can easily be achieved by defining new artifact types that provide developers the appropriate protocol language primitives, such as those offered by 2CL [6][5].

Acknowledgments.: The authors would like to thank the reviewers for their comments and the participants to the EMAS 2013 workshop for the discussions.

References

[1] FIPA specifications. http://www.fipa.org.

[2] M. Baldoni and C. Baroglio. Some Thoughts about Commitment Protocols (Position Paper). In *Post-Proc. of the 10th Int. Workshop on Declarative Agent Languages and Technologies X, DALT 2012, Revised Selected and Invited Papers*, *LNAI 7784*, pages 190–196. Springer, 2013.

[3] M. Baldoni, C. Baroglio, F. Bergenti, E. Marengo, V. Mascardi, V. Patti, A. Ricci, and A. Santi. An interaction-oriented agent framework for open environments. *AI*IA 2011: Artificial Intelligence Around Man and Beyond*, 6934, 2011.

[4] M. Baldoni, C. Baroglio, E. Marengo, V. Patti, and A. Ricci. Back to the future: An interaction-oriented framework for social computing. In *First Int. Workshop on Req. Eng. for Social Computing, RESC*, pages 2–5. IEEE, 2011.

[5] M. Baldoni, C. Baroglio, E. Marengo, F. Capuzzimati, and V. Patti. A Generalized Commitment Machine for 2CL protocols and Its Implementation. In *Post-Proc. of the 10th Int. Workshop on Declarative Agent Languages and Technologies X, DALT 2012, Revised Selected and Invited Papers*, *LNAI 7784*, pages 96–115. Springer, 2013.

[6] M. Baldoni, C. Baroglio, E. Marengo, V. Patti, and F. Capuzzimati. Engineering commitment-based business protocols with 2CL methodology. *J. of Autonomous Agents and Multi-Agent Systems*, August 2013 (to appear).

[7] M. Baldoni, G. Boella, V. Genovese, A. Mugnaini, R. Grenna, and L. van der Torre. A Middleware for Modelling Organizations and Roles in Jade. In *Programming Multi-Agent Systems, 7th Int. Workshop, ProMAS 2009, Revised Selected Papers*, *LNAI 5919*, pages 100–117. Springer, 2010.

[8] M. Baldoni, G. Boella, and L. van der Torre. Bridging Agent Theory and Object Orientation: Agent-like Communication among Objects. In *Post-Proc. of the International Workshop on Programming Multi-Agent Systems, ProMAS 2006*, *LNAI 4411*, pages 149–164. Springer, 2007.

[9] M. Baldoni, G. Boella, and L. van der Torre. Interaction between Objects in `powerjava`. *Journal of Object Technology*, 6(2), 2007.

[10] F. Bellifemine and A. Poggi. JADE – A FIPA-compliant agent framework. *Proceedings of PAAM*, 1999.

[11] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software-Practice and Experience*, (July 1999):103–128, 2001.

[12] C. Bernon, M. P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: A methodology for adaptive multi-agent systems engineering. In *ESAW*, volume 2577 of *LNCS*, pages 156–169. Springer, 2002.

[13] G. Boella and L. W. N. van der Torre. An agent oriented ontology of social reality. In *Procs. of Formal Ontologies in Information Systems (FOIS)*. IOS Press, 2004.

[14] G. Boella and L. W. N. van der Torre. The ontological properties of social roles in multi-agent systems: definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law*, 15(3):201–221, 2007.

[15] O. Boissier, R. H. Bordini, J. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 2011.

[16] G. Caire, W. Coulier, Fr. J. Garijo, J. Gomez, J. Pavón, F. Leal, P. Chainho, Paul E. Kearney, J. Stark, R. Evans, and P. Massonet. Agent oriented analysis using message/uml. In *Proc. of AOSE 2001*, pages 119–135. Springer-Verlag, 2002.

[17] G. Caire, D. Gotta, and M. Banzi. Wade: a software platform to develop mission critical applications exploiting agents and workflows. In *AAMAS (Industry Track)*, pages 29–36. IFAAMAS, 2008.

[18] C. Castelfranchi. Principles of Individual Social Action. In *Contemporary action theory: Social action*, volume 2, pages 163–192, Dordrecht, 1997. Kluwer.

[19] A. K. Chopra and M. P. Singh. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proc. of ProMAS*, 2009.

[20] A.K. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing*. PhD thesis, North Carolina State University, Raleigh, NC, 2009.

[21] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous Norm Acceptance. In *Proc. of ATAL*, volume 1555 of *LNCS*, pages 99–112. Springer, 1998.

[22] M. Dastani, D. Grossi, Meyer. J.-J. Ch., and N. A. M. Tinnemeier. Normative Multi-agent Programs and Their Logics. In *KRAMAS*, pages 16–31, 2008.

[23] Jan L.G. Dietz. Understanding and modelling business processes with demo. In *Proc. of ER'99, 18th Int. Conf. on Conceptual Modeling*, volume 1728 of *LNCS*, pages 188–202. Springer, 1999.

[24] Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. Symbolic model checking commitment protocols using reduction. In *DALT*, volume 6619 of *Lecture Notes in Computer Science*, pages 185–203. Springer, 2010.

[25] N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.

[26] Nicoletta Fornara, Francesco Viganò, Mario Verdicchio, and Marco Colombetti. Artificial institutions: a model of institutional reality for open multiagent systems. *Artif. Intell. Law*, 16(1):89–105, 2008.

[27] J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents: "Giving the organisational power back to the agents". *Autonomous Agents and Multi-Agent Systems*, 20, 2009.

[28] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. *Inf. Soc.*, 9(4):391–404, 1993.

[29] M. T. Nguyen, P. Fuhrer, and J. Pasquier-Rocha. Enhancing e-health information systems with agent technology. *Int. J. Telemedicine Appl.*, 2009:1:1–1:13, 2009.

[30] M. Nikraz, G. Caire, and P. A. Bahri. A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. (May), 2006.

[31] D. Okouya, N. Fornara, and M. Colombetti. An Infrastructure for the Design and Development of Open Interaction Systems. In *Proc. of the 1st International Workshop on Engineering Multi-Agent Systems, EMAS 2013, held in conjuction with AAMAS 2013*, pages 128–143, St. Paul, Minnesota, USA, May 2013.

[32] A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, 2011.

[33] T. F. Rodrigues, A. C. da Rocha Costa, and G. P. Dimuro. A Communication Infrastructure Based on Artifacts for the JaCaMo Platform. In *Proc. of the 1st Int. Workshop on Engineering Multi-Agent Systems, EMAS 2013*, pages 97–111, St. Paul, Minnesota, USA, May 2013.

[34] M. P. Singh. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

[35] M. P. Singh. A social semantics for agent communication languages. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 31–45. Springer, 2000.

[36] M. P. Singh. LoST: Local Transfer - An Architectural Style for the Distributed Enactment of Business Protocols. *Proc. of the 9th Internactional Conference on Web Services*, pages 57–64, IEEE Computer Society, 2011.

[37] P. R. Telang and M. P. Singh. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE Transactions on Services Computing*, pages 1–14, 2011.

[38] T. Winograd and F. Flores. *Understanding computers and cognition - a new foundation for design*. Addison-Wesley, 1987.

[39] M. F. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology. In *AOSE*, *LNCS 1957*, pages 207–222. Springer, 2000.

[40] P. Yolum and M. P Singh. Designing and executing protocols using the event calculus. *Proc. of the 5th Int. Conf. on Autonomous agents - AGENTS '01*, pages 27–28, 2001.

[41] P. Yolum and M. P. Singh. Commitment Machines. In *Intelligent Agents VIII, 8th International Workshop, ATAL 2001*, *LNCS 2333*, pages 235–247. Springer, 2002.

[42] Pinar Yolum. Design time analysis of multiagent protocols. *Data Knowledge Engineering*, 63(1):137–154, 2007.

[43] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.

Author Index