Spatial Edge Services - From Coordination Model to Actual Applications

Giovanna Di Marzo Serugendo

Spatial Edge Services - From Coordination Model to Actual Applications

Giovanna Di Marzo Serugendo
giovanna.dimarzo@unige.ch

Centre Universitaire d'Informatique, Institute of Services Science,
University of Geneva, SWITZERLAND

Abstract Ubiquitous and context-aware sensors are increasing in number and aim at providing comfort and better life quality. They are spatially distributed and their computation capacity is still under-exploited. *Spatial Edge Service* are a new generation of services exploiting IoT and spatially distributed data. They result from collective and decentralised interactions of multiple computing entities. They rely on a logic and chemical-based coordination model. Spatial edge services provide innovation capabilities for the software industry, connected objects manufacturers and edge computing industry. This paper provides and overview of Spatial Edge Services, their underlying coordination model, a set of development tools, a series of case studies scenarios and future visions.

1 Introduction

Mobile phones, laptops, tablets locally connected to each other form huge computing and storage infrastructures, currently under-exploited, but available on motorways, city-centers, inside buildings, etc. Those infrastructures pave the way for a new category of services based on data propagation among devices, e.g. car traffic control service through propagation of data from one car to another, information dissemination in a crowd to better steer the crowd towards points of interest or emergency exits, and alternative communication infrastructures in case of environmental disasters. Such services are time-related: they may last just for a very short time for a specific purpose exploiting current contextual data, as well as space-related: these services have a meaning because the data they rely on or the data they spread is spatially distributed over a geographic area.

Traditional service-oriented approaches allow programmers to combine together into a new service diverse functionalities provided themselves under the form of services. Typical approaches involve Web services or rely on APIs mashups. Composition of services is generally foreseen at design-time, adaptation happens by choosing the actual services at run-time. Current service-oriented approaches are not suited for the new generation of services, working on smart environments or exploiting Internet of Things scenarios. They do not cope with the dynamicity involved by the underlying mobile and changing computing infrastructures, the spatiality of the considered data, or time-related issues. There is a need for a paradigm shift in the notion of services along two dimensions: (1) services make sense because they are spatially distributed, their functionality is provided as the result of the interactions among several entities, possibly distributed across several computational nodes; (2) services are built on demand, e.g. users can query diffused clouds of data in a smart city, to retrieve the closest vacant parking place, sensors, "things" and services spontaneously collaborate to provide the answer to the query. We call such services *spatial edge services*. This paper summarises our research works related to the notion of spatial services.

2 Spatial Edge Services

Spatial edge services are a new generation of services that exploit spatially distributed data, enable smart environments, or exploit Internet of Things (IoT) scenarios. This is a new category of decentralised services based on data propagation among mobile devices and where the functionality of the service is provided as the result of the collective interactions among multiple entities, involving processes and calculations taking place across several spatially (geographically) distributed computational entities (i.e. sensors, mobile or stationary computational nodes, actuators). Spatial edge services are built and composed on-demand. Spatial edge services are based on Spatial Computing [29]. Spatial computing thrives on decentralised computation, in a way similar to edge computing, fog computing or jungle computing, where computation is pushed at the edge of the network, away from centralised clouds, closer to end-users [23]. Spatial services bring an engineering dimension to spatial computing as they provide ready to use services that can be deployed on-demand over physical environments by higher-level services or applications. A spatial service is built dynamically through collaboration with other services [17]. Spatial edge services have time- and space-enabled capabilities: deposit data at geographic locations, retrieve data, aggregate data, provide information to users, evaporate information, or act on the environment. Geographically distributed data collectively provides a specific meaning (e.g. artificial gradient).

Bio-inspired mechanisms. Self-organisation, as exhibited by natural systems, provides an appealing approach to engineer spatial services where the functionality and coordination arises from the interactions among several autonomous entities; no central control is required; and some robustness properties and adaptation in dynamic large-scale environments are naturally provided. Self-organisation is achieved by the use of self-organising mechanisms [21] (Figure 1), i.e., rules that autonomous entities employ to coordinate their behaviour, usually following information gathered from their local environment. One of the main ideas behind spatial service is to provide the above mechanisms under the form of services, ready to use by other services. Therefore, among spatial services, we distinguish on the one hand, Spatial User Services (like finding a vacant parking place), and Spatial System Services, those services that implement spatial self-organisation mechanisms like spreading or gradient for example.

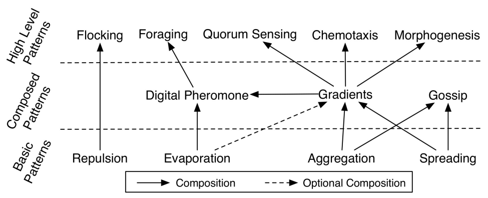


Figure 1: Self-organising mechanisms

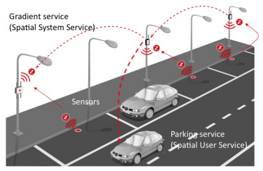


Figure 2: Spatial Services

Spatial System Services [17] bring an engineering dimension to spatial computing as they provide ready to use services that can be deployed on-demand over physical environments by higher-level services or applications. Spatial system services are based on self-organising mechanisms [21], they are time- and space-related, work on a decentralised and autonomous manner, and are naturally robust to some environmental perturbations, even though not to any perturbation or environmental change [15]. Figure 1 depicts these self-organising mechanisms as well as their inter-relations (e.g. gossip uses spreading and aggregation).

Spatial User Services are the services offered at the application level (for example the services offered by a smart city like help for parking, visually impaired visitor guided across a campus). They are provided through dynamic selection of underlying spatial system services. Figure 2 shows an example of such service, parking place sensors communicate with the nearest street lamp. Information about empty places spread, thanks to the Gradient system service among street lamps, until it reaches the user requesting that information.

Chemical-based coordination platform. Spatial services rely on the SAPERE coordination model that is built on chemical reactions and active shared tuple spaces [30] and its corresponding middleware [4]. Depending on their nature and location, objects are attached to diverse computational nodes (e.g. cars, smartphones, raspberry pi, etc.). Each computational node runs an instance of the coordination platform. Software agents run in the node and interact on behalf of objects (e.g. lamps, cameras, etc.), services, or applications with the coordination platform. They inject or retrieve information to/from the platform, and are sensitive to data present in the platform. Software agents also act on the objects (e.g. switch on/off a lamp, turn surveillance camera), update data spread by a spatial service, or provide a parking place location. Figure 3 depicts the case of a computational node, running an instance of the coordination platform. Computational nodes are connected through low-level connection mechanisms.

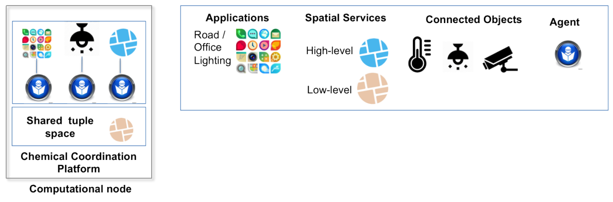


Figure 3: Coordination Model

3 Design Patterns

Bio-inspired mechanisms. In biological systems, complex emergent behaviour can be achieved by letting agents follow a set of basic rules while having only partial knowledge of the environment. Typical bio-inspired self-organising mechanisms are those using stigmergy, like ant foraging for coordinating behaviour, schooling and flocking for coordinating movements, or gradients based systems [5,18].

Bio-inspired design patterns. To ease the use of bio-inspired mechanisms and to apply them more systematically, a series of authors proposed descriptions of these mechanisms under the form of design patterns [27,25,1,14,22]. Results show that the bio-inspired decentralised approaches achieve better results (in terms of robustness, optimisation and adaptability) than more traditional approaches based on formal modelling. To ease engineering of artificial bio-inspired systems, we described a catalogue of bio-inspired mechanisms in terms of modular and reusable design patterns organised into different layers [21], shown on Figure 1. The mechanisms presented are uniformly described and framed using a software design pattern structure identifying when and how to use each pattern, and describing the relationship between the different mechanisms and their respective boundaries. This catalogue of mechanisms is a step forward to engineering self-organising systems in a systematic way.

4 Suite of Spatial Services

Despite the interest in building self-organising systems, there is currently no established way of designing and programming self-organising systems or applications in a systematic way. Applications are developed in an ad-hoc manner and the functionality of the application is closely entangled with the underlying self-organising mechanism. To implement a crowd steering application using digital pheromones, in addition to programming the crowd steering application functionality it is also necessary to implement the behaviour of the digital pheromone itself. We then consider that for engineering self-organising systems it is important to separate the concerns at different levels: (1) Separate functionality of an application from the underlying mechanisms it uses; (2) Delegate underlying mechanisms and their non-functional, self-managing aspects to the underlying environment.

Self-organising mechanisms as services. In consequence, we propose that self-organising mechanisms (e.g. gradients, digital pheromone or gossip) be provided to higher-level services or applications as ready-to-use services. This involves the design and implementation of a computational environment that provides reliable low-level self-organising mechanisms in the form of services, reusable by applications. This allows the implementation of self-organising applications in a modular way, favours reuse of mechanisms by decoupling them from the application functionality and delegating responsibilities for them to the computational environment.

Core service. Following the relationship among self-organising mechanisms, Core services provided at the level of the infrastructure or computational environment are: Spreading, Aggregation, Evaporation, Static Gradients. Higher-level services, developed by re-using and extending low-level self-organising services provided by the infrastructure, are: Dynamic Gradients, Chemotaxis, Remote Query and Retrieval (a composed service that spreads a query using the Gradient or Dynamic Gradient service and uses the Chemotaxis service to bring back information from remote nodes [19]). We propose to program self-organising systems using services that implement self-organising mechanisms, abstracting away the implementation of those low-level mechanisms for the developer and favouring the re-use of code.

Higher-level services. At the lower level, the computational environment provides reliable "Core" services, i.e. low-level self-organising mechanisms and proposes them to higher-level services or to applications in the form of ready-to-use "services" (e.g. spreading or evaporation). The implementation of higher-level self-organising services (e.g. digital pheromone or dynamic gradient) uses and exploits those low-level services. Higher-level services are themselves ready to be used by actual applications. Finally, applications exploit both high-level and low-level self-organising services. Their implementation happens then in a modular way, reusing self-organising mechanisms, decoupling them from the application functionality, as shown in Figure 4.

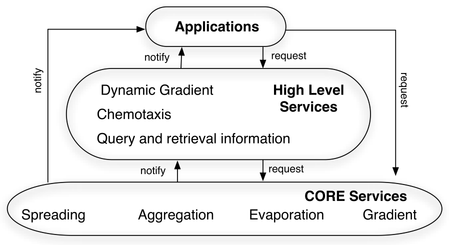


Figure 4: Spatial Services

5 Coordination Models

Coordination models have proven useful for designing and implementing distributed systems. They are appealing for developing collective adaptive systems working in a decentralised manner, interacting with their local environment, since the shared tuple space on which they are based is a powerful paradigm to implement bio-inspired mechanisms (e.g. stigmergy). Coordination infrastructures provide the basic mechanisms and the necessary middleware to implement and deploy coordinated systems. Different categories of coordination models have been developed: chemically-inspired models work as rewrite systems, where states are interpreted as chemical substances, and where molecules represent the coordinated entities that interact according to some reaction rules [30]. Physics-based models, are instead inspired by the way physical masses and particles move and self-organise according to gravitational and electromagnetic fields [26,2].

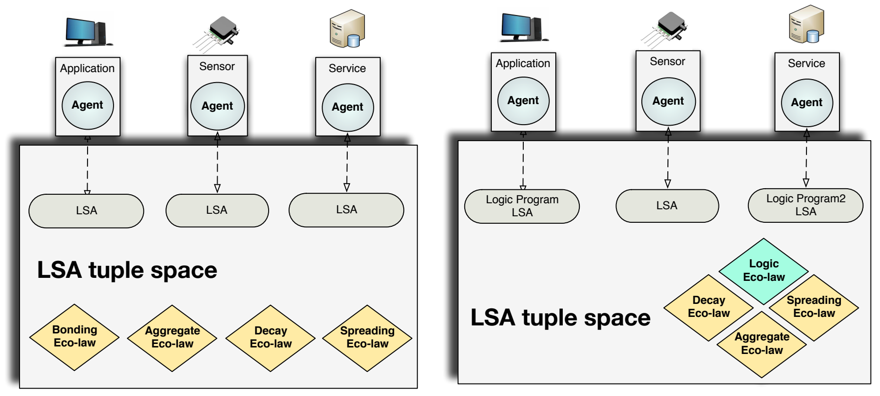
5.1 SAPERE

Our work derives from the SAPERE model [3], a coordination model for multi-agent pervasive systems inspired by chemical reactions. It is based on four main concepts: *Live Semantic Annotations* (LSAs), *LSA Tuple Space*, *agents* and *eco-laws*. LSA are tuples of types $(name, value)$ used to store applications data. For example, a tuple of type $(date, 04/04/1988)$ can be used to define a hypothetical date. LSAs belonging to a computing node are stored in a shared container named LSA Tuple Space. Each LSA is associated with an agent, an external entity that implements some domain-specific logic. For example, agents can represent sensors, services or general applications that want to interact with the LSA space - injecting or retrieving LSAs from the LSA space. Inside the shared container, tuples react in a virtual chemical way by using a predefined set of coordination rules named *eco-laws*, which can (i) instantiate relationships among LSAs (*Bonding* eco-law), (ii) aggregate them (*Aggregate* eco-law), (iii) delete them and (*Decay* eco-law) and (iv) spread them across remote LSA Tuples Spaces (*Spreading* eco-law). When a tuple is modified by an *eco-law*, its relative agent is notified. The implementation of the SAPERE model (Figure 5(a)), named SAPERE middleware permitted the development of several kinds of real distributed self-adaptive and self-organising applications [30].

5.2 Logic-Based Chemical Coordination Model (LFCM)

A well-known difficulty with self-organising systems stems from the analysis, validation and verification (at design-time or run-time) of so-called emergent properties - i.e. properties that can be observed at a global level but that none of the interacting entities exhibit on its own. Few coordination models integrate features supporting the validation of emergent properties, none of them relying on the chemical metaphor. Recently we extended SAPERE and defined a coordination model based on logic inference named Logic Fragments Coordination

Model (LFCM) [10] as well as its semantics [12] (Figure 5(b)). Logic Fragments are combinations of logic programs defining interactions among agents distributed over the nodes of the system. They are able to accommodate various types of logics, ranging from classical up to many-valued paraconsistent ones. The logical formalisation makes it possible to express coordination in a rigorous and predictable way, both at design-time and run-time. Our logic-based coordination model allows agents to inject logic fragments into the shared space. An additional eco-law, the logic fragement eco-law interprets those fragments based on the current tuples in the tuple space (including neighbouring ones). Those fragments actually define on-the-fly ad hoc chemical reactions that apply on matching data tuples present in the system, removing tuples and producing new tuples, possibly producing also new logic fragments. Our model is defined independently of any specific logic, an actual instantiation and implementation of the model can use its own logic(s). We also defined a spatial language to verify graph-based spatial properties of self-organising systems [6], a multi-valued logic [7] and an approximate reasoning [13]. The language encapsulates Logic Fragments in statements that are evaluated in a distributed manner at run-time, involving several system entities at the same time. The corresponding middleware for two-valued logic is available. LFCM is composed of the following elements (Figure 6) :



(a) SAPERE Coordination Model (b) Logic Fragment Coordination Model
Figure 5: Coordination models

• *Agents:* autonomous entities representing applications, services and sensors.
• *Tuple space:* shared space for all the agents running on the same node of the system. The Tuple Space stores information under the form of tuples, which can be *passive* (e.g. contextual-information, data input for service invocation, etc.) and *active* (logic fragments).
• *Logic fragments:* combinations of logic programs manipulating passive data stored in the Tuple Space and introduced either by Agents or generated by other logic fragments. They can be injected, removed and copied among nodes of the

system at run-time. We consider many-valued logics with *true*, *false*, *unknown* and *inconsistent* logical values.

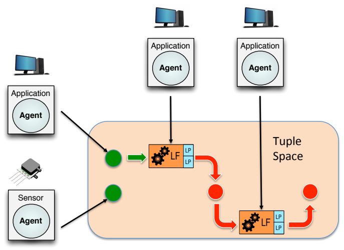


Figure 6: Components of the Logic Fragment Coordination Model. Green tuples represent passive data injected by Agents, red ones depict information generated by logic fragments.

LFCM combines the benefits of chemical-based coordination models and formal methods, offering a framework to rigorously design and implement self-organising systems. In the LFCM logical formulae are represented by combinations of logic programs, whereas the execution flow is controlled through predicates and operators defining how logic programs are interpreted and how data is manipulated through predicates (e.g. to create, aggregate, delete and spread information).

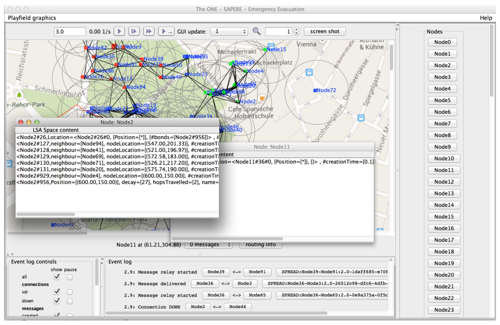
– When designing complex coordination mechanisms, great emphasis is put on the definition of the logic part, enforcing a rigorous formalisation of the algorithm while hiding most of implementation details. In such a way, code of system units becomes an essential set of logical rules defining the behaviour of components with respect to the distributed state of the system. This aspect ensures that *the set of spontaneous interactions taking place in the system is the one that is logically entailed from the rules.*
– The usage of logic programming to design system units makes it possible to adopt a uniform formal framework to specify, verify, synthesise system components. This is possible because pure logic can be used as a language for system specifications. When designing complex distributed systems these properties become extremely important because *they can be exploited to enforce correctness at design-time.*
– Given that system components are encapsulated into combinations of logic programs, they can be easily added, removed and shared among nodes. *This feature enables the installation and removal of ad-hoc coordination primitives on-the-fly.*
– By resorting to logic programming, interactions among components are driven by logic inference and system components can be considered as units performing formal reasoning processes on the distributed state of the system. By

resorting to this feature, *components can be exploited to evaluate distributed system properties at run-time [11].*

6 Prototyping platforms

TheOne-SAPERE. TheOne-SAPERE is a prototyping tool [20] that integrates the SAPERE middleware within The Opportunistic Network Environment (The One) simulator [24], allowing us to prototype and validate applications with realistic scenarios before deploying them. Indeed, it allows on the one hand to simulate a large number of computational nodes movements and their communications, placing them in various configurations allowing stochastic evaluation of parameters. On the other hand, each node is equipped with the actual SAPERE middleware (actual code), allowing to execute from within the simulation actual spatial system services (gradient, spreading, evaporation, etc.), thus providing actual results relating to spatial system services behaviour (Figure 7). The corresponding video is available on-line [1].

TheOne-LFCM. Following the above idea, we similarly developed a corresponding simulator, called TheOne-LFCM, involving the actual LFCM middleware in each simulated node.



```
<Node2#26,Location=<Node2#26#0, {Position=[*]}, {#bonds=[Node2#956]}> , #
<Node2#127,neighbour=[Node94], nodeLocation=[(547.00,201.33)], #creationTi
<Node2#128,neighbour=[Node41], nodeLocation=[(521.00,196.97)], #creationTi
<Node2#129,neighbour=[Node69], nodeLocation=[(572.58,183.00)], #creationTi
<Node2#130,neighbour=[Node71], nodeLocation=[(526.21,217.20)], #creationTi
<Node2#131,neighbour=[Node20], nodeLocation=[(575.74,190.00)], #creationTi
<Node2#929,neighbour=[Node4], nodeLocation=[(600.00,150.00)], #creationTin
<Node2#956,Position=[(600.00,150.00)], decay=[27], hopsTravelled=[2], name=
```

```
ion=<Node11#36#0, {Position=[*]}, {}> , #creationTime=[0.1]
```

```
2.9: Message relay started  Node39 <-> Node91  SPREAD:Node39-Node91:2.0-1daff685-e706
2.9: Message delivered  Node36 <-> Node3  SPREAD:Node36-Node3:2.0-26012c98-dfc6-4dfb-
2.9: Message relay started  Node36 <-> Node65  SPREAD:Node36-Node65:2.0-0e9a375a-0f5d
2.9: Connection DOWN  Node2 <-> Node46
```

Figure 7: TheOne-SAPERE

7 Developed Applications

We discuss here a series of applications actually deployed on tablets and smartphones. For more IoT applications, see [16].

Confidential Channel. The Confidential Channel service [8] provides an encrypted channel for routing confidential communication between devices in a

[1] http://youtu.be/EGbPq2rejmM

mobile ad-hoc network. The computational power and memory available in mobile devices, is increasing dramatically, and allows now the use of traditional encrypting techniques that some years ago were available only in wired computers, such as laptops or PCs. Therefore, the Secure Channel service uses RSA encryption on top of Dynamic Gradient and Chemotaxis services in order to ensure and maintain confidential communication and routing even in complex mobile networks. Using a Gradient service, node A sends a public key and a request for information. They are both propagated to all nodes in the network along with additional gradient information about the sender's distance (e.g. number of hops). When the Gradient reaches node B, the latter retrieves the query and the public key. B encrypts the information using the received public key and sends back the encrypted information using the Chemotaxis service using the shortest available path. The gradient structure is updated periodically in order to deal with network's topology changes. We subsequently developed a chat over the confidential channel.

SmartContent This is a novel approach for content protection and privacy. Documents are active and context-aware documents that sense and analyse their current context, e.g. location, noise, neighbouring devices, social network, expiration time, etc. Based on user provided policies, they grant, deny or limit access and manipulation actions, or destroy themselves if necessary. The implementation leverages the SAPERE middleware. Context information (GPS location, camera picture, etc.) is provided as tuples in the SAPERE middleware. Specific agents, wrapping documents, enforce users' policies. They are sensitive (bond) to context information relevant to the document they protect and enact when necessary the privacy policy (revealing or not the document) [9].

Context-aware data flows. The goal of this demonstration is to show-case self-organising context-aware data flows with a particular emphasis on the self-organising mechanisms [21] that are at the heart of it: (a) diffusion - propagates information among the nodes; (b) aggregation - allows the system to reduce the amount of information spread in the system or obtained from the environment; (c) evaporation - periodically decreases the relevance of the information in order to get rid of outdated information.

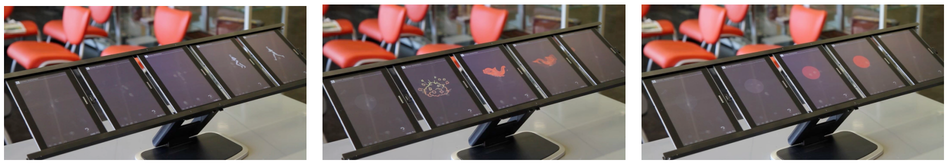
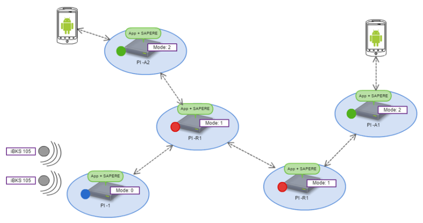

(a) Diffusion (b) Aggregation (c) Evaporation

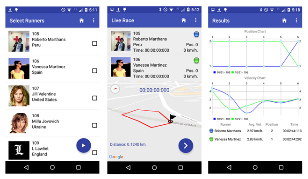
Figure 8: Self-organising mechanisms - concepts

In the first part of the demonstration, data flows of pictures spread, in a P2P fashion, from one tablet to the other. Data flows are sensitive to their context as follows: tapping tablets causes a new data flow to spread from one tablet to the neighbouring one (Figure 8 (a)); similarly swiping creates a new flow in the direction of the swipe; several flows meeting at the same time in the same tablet aggregate with each other. Here a simple aggregation is chosen, where one of the two flows is stopped (shown as an explosion in the demonstration) (Figure 8 (b)). Figure 8(c) shows an example of evaporation, with time data relevance fades away and ultimately information disappears. In the second part of the demonstration, data flows become services used to retrieve a friend in a crowd. A query first spreads from tablet to tablet following the gradient principle - a type of diffusion with a notion of distance [21]; once the message has reached the friend, the answer follows back the path created by the gradient . This demonstration has been featured at the ICT'13 Conference in Vilnius, the video is available here[2].

Tracking mobile objects/runners - querying IoT. Runners are sensed along the racetrack and their positions spread continuously in order to provide a real-time tracking. A variety of devices (beacons), and nodes (raspberry pi and smartphones) are exchanging data using Bluetooth Low Energy (BLE) to track mobile objects in a peer-to-peer network. Let us consider that each runner is equipped with a small device that emits continuously a signal. We also deploy several nodes along the path sensing and relaying these signals. People in the crowd with a smartphone and an appropriate mobile application request informations about their favourite runners and receive a response in real-time. In other words, people track their favourite runners while they are running and follow their performance during the race.



(a) Live Scenario



105
Roberto Marthans
Peru

106
Vanessa Martinez
Spain

107
Jill Valentine
United States

108
Milla Jovovich
Ukraine

109
L Lawliet
England

105
Roberto Marthans
Peru
Time: 00:00:00:000
Pos. 0
0 km/h.

106
Vanessa Martinez
Spain
Time: 00:00:00:000
Pos. 0
0 km/h.

00:00:00:000

Distance: 0.1240 km.

Runner	Avg. Vel.	Position	Time
Roberto Marthans	2.97 km/h.	2	00:02:44:113
Vanessa Martinez	2.83 km/h.	1	00:02:44:292

(b) Live Application

Figure 9: Tracking runners

Each competitor runs with a beacon that emits a BLE signal. A beacon identifies a runner by communicating its runner identifier to its surrounding environment. Several nodes, running a SAPERE application built over the SAPERE middleware, are deployed along the path to sense BLE signals and compute their position in the race, their location on the map, and the time they have

[2] http://youtu.be/4S4J5wYvdNk

been running. Every node spreads received signals and people in the crowd interact across a peer-to-peer network to request information about their favourite runners (Figure 9(a)). People track their favourite players while they are running and know about their performance live during the race (Figure 9(b)). The information remains spatially available in the area only during the time when the event takes place. A complete description and full details can be found here [28].

Assessing IoT properties on-the-fly. We show here how to resort to the Logic Fragment Coordination Model to design and implement a distributed monitoring spatial edge service for an industrial warehouse in order to avoid safety hazards arising from potential interactions among chemical compounds. The model of the warehouse system includes: Containers (e.g. tanks);

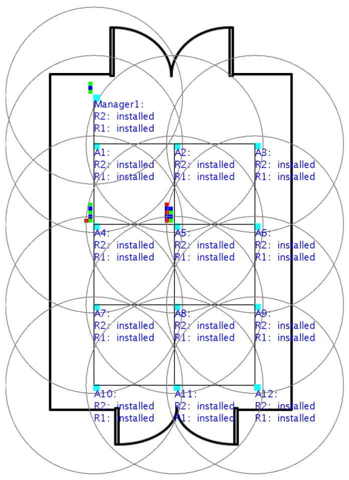


Figure 10: Warehouse and communicating shelf units.

Shelf units supporting the physical storage of containers in a particular point of the warehouse; Policies constraints on physical positions of containers implementing safety regulations (e.g. proximity relations among chemical compounds, etc). Policies can be established, replaced or removed at run-time. By resorting to a many-valued logic, policies can be *strongly violated*, *weakly violated*, *weakly accepted* and *strongly accepted*. Our warehouse model is composed of 12 shelf units connected in a grid network; shelf units must be considered as autonomous agent entities, able to perform computations and communicate with each other (Figure 10). The monitoring service triggers an alarm signal when one or several policies are violated on a particular shelf unit.

Whenever a policy is violated on a shelf unit, its associated logic fragment generates a tuple of type *Alam(Rule)*, where *Rule* is the identifier of the policy. An additional logic fragment (controller of the system) aggregates all tuples of type *Alarm(Rule)*, generating a tuple *Alarm* with a logical level that represents the level of criticality of the violated policies.

8 Future works

Future works involve multiple paths: integrating learning aspects within agents, analysing QoS for each of the different spatial services involved in an application and their impact on each other, pursuing current efforts on privacy and security, actually building spatial services on demand, arising from existing services, sensors and actuators.

Acknowledgements

This work has been supported by the EU-FP7-FET Proactive project SAPERE Self-aware Pervasive Service Ecosystems, under contract no. 256873. It is the result of the joint work of the SAPERE team. I would like to particularly thank the Geneva team, namely: Akla Esso Tchao, Jose Luis Fernandez-Marquez, Francesco Luca De Angelis, Houssem Ben Mahfoudh, Roberto Tomaylla.

References

1. Babaoglu, O., Canright, G., Deutsch, A., Caro, G.A.D., Ducatelle, F., Gambardella, L.M., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A., Urnes, T.: Design patterns from biology for distributed computing. ACM Trans. on Autonomous and Adaptive Sys 1, 26–66 (2006)
2. Beal, J., Bachrach, J.: Infrastructure for engineered emergence on sensor/actuator networks. IEEE Intelligent Systems (2006)
3. Castelli, G., Mamei, M., Rosi, A., Zambonelli, F.: Pervasive middleware goes social: The sapere approach. In: Proceedings of the 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops. pp. 9–14. SASOW '11 (2011)
4. Castelli, G., Mamei, M., Rosi, A., Zambonelli, F.: Engineering pervasive service ecosystems: The sapere approach. ACM Trans. Auton. Adapt. Syst. 10(1), 1:1–1:27 (Mar 2015), http://doi.acm.org/10.1145/2700321
5. de Castro, L.N.: Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications (Chapman & Hall/Crc Computer and Information Sciences). Chapman & Hall/CRC (2006)
6. De Angelis, F., Di Marzo Serugendo, G.: A logic language for run time assessment of spatial properties in self-organizing systems. In: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops. pp. 86–91 (2015)
7. De Angelis, F., Di Marzo Serugendo, G., Szałas, A.: Graded rule-based reasoning. Accepted to International Journal of Approximate Reasoning to appear (2017)
8. De Angelis, F., Fernandez-Marquez, J.L., Di Marzo Serugendo, G.: Secure channel for manets - demonstration. In: IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO2013). IEEE Computer Society (2013)

9. De Angelis, F.L., Di Marzo Serugendo, G.: Smartcontent - self-protected context-aware active documents for mobile environments. Electronics 6(1) (2017), http://www.mdpi.com/2079-9292/6/1/17
10. De Angelis, F.L., Di Marzo Serugendo, G.: Coordination Models and Languages: 17th IFIP WG 6.1 International Conference, COORDINATION 2015, chap. Logic Fragments: A Coordination Model Based on Logic Inference, pp. 35–48. Springer International Publishing, Cham (2015)
11. De Angelis, F.L., Di Marzo Serugendo, G.: A logic language for run time assessment of spatial properties in self-organizing systems. 9th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), Springer (2015)
12. De Angelis, F.L., Di Marzo Serugendo, G.: Logic fragments: coordinating entities with logic programs. In: 7th International Symposium On Leveraging Applications of Formal Methods Verification and Validation (ISOLA), (2016)
13. De Angelis, F.L., Di Marzo Serugendo, G., Dunin-Keplicz, B., Sza?as, A.: Heterogeneous approximate reasoning with graded truth values. In: International Joint Conference on Rough Sets (2017)
14. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. Engineering Self-Organising Systems 4335, 28–49 (Feb 2007)
15. Di Marzo Serugendo, G.: Robustness and dependability of self-organising systems - a safety engineering perspective. In: Int. Symp. on Stabilization, Safety, and Security of Distributed Systems(SSS). LNCS, vol. 5873, pp. 254–268. Springer, Berlin Heidelberg, Lyon, France (2009)
16. Di Marzo Serugendo, G., Abdennadher, N., Ben Mahfoudh, H., De Angelis, F.L., Tomaylla, R.: Spatial edge services. Global IoT Summit (2017)
17. Di Marzo Serugendo, G., Fernandez-Marquez, J.L., De Angelis, F.L.: Engineering spatial services: Concepts, architecture, and execution models. In: Ramanathan, R., Raja, K. (eds.) Handbook of Research on Architectural Trends in Service-Driven Computing, pp. 136–159. IGI Global (2014)
18. Di Marzo Serugendo, G., Gleizes, M.P., Karageorgos, A. (eds.): Self-Organising Software - From Natural to Artificial Adaptation. Natural Computing Series, Springer, 1st edition edn. (2011)
19. Fernandez-Marquez, J.L., Tchao, A.E., Di Marzo Serugendo, G., Stevenson, G., Ye, J., Dobson, S.: Analysis of new gradient based aggregation algorithms for data-propagation in distributed networks. In: 1st Int. Workshop on Adaptive Service Ecosystems: Nature and Socially Inspired Solutions (ASENSIS) at 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO2012) (2012)
20. Fernandez-Marquez, J.L., Angelis, F.D., Serugendo, G.D.M., Stevenson, G., Castelli, G.: The one-sapere simulator: A prototyping tool for engineering self-organisation in pervasive environments. In: SASO. pp. 201–202. IEEE Computer Society (2014), http://dblp.uni-trier.de/db/conf/saso/saso2014.html#Fernandez-MarquezASSC14
21. Fernandez-Marquez, J.L., Serugendo, G.D.M., Montagna, S., Viroli, M., Arcos, J.L.: Description and composition of bio-inspired design patterns: a complete overview. Natural Computing pp. 1–25 (2012)
22. Gardelli, L., Viroli, M., Omicini, A.: Design patterns for self-organizing multiagent systems. In: Proceedings of EEDAS (2007)
23. Hajibaba, M., Gorgin, S.: A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing. CIT - Journal of Computing and Information Technology 22(2) (2014)

24. Keränen, A., Ott, J., Kärkkäinen, T.: The ONE Simulator for DTN Protocol Evaluation. In: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST, New York, NY, USA (2009)
25. Mamei, M., Menezes, R., Tolksdorf, R.and Zambonelli, F.: Case studies for self-organization in computer science. J. Syst. Archit 52, 433–460 (2006)
26. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications: The tota approach. ACM Trans. Softw. Eng. Methodol. 18(4), 15:1–15:56 (Jul 2009), http://doi.acm.org/10.1145/1538942.1538945
27. Nagpal, R.: A catalog of biologically-inspired primitives for engineering self-organization. In: Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering. pp. 53–62. Springer (2004)
28. Tomaylla, R.: Bio-Inspired approach for tracking mobile entities in a Peer to Peer MANET. An application of The SAPERE Project in Road Running events . Master's thesis, Centre Universitaire d'Informatique, University of Geneva, Geneva, Switzerland (2016)
29. Zambonelli, F., Mamei, M.: Spatial computing: An emerging paradigm for autonomic computing and communication. In: Proceedings of the First International IFIP Conference on Autonomic Communication. pp. 44–57. WAC'04, Springer-Verlag, Berlin, Heidelberg (2005), http://dx.doi.org/10.1007/11520184_4
30. Zambonelli, F., Omicini, A., Anzengruber, B., Castelli, G., Angelis, F.L.D., Serugendo, G.D.M., Dobson, S., Fernandez-Marquez, J.L., Ferscha, A., Mamei, M., Mariani, S., Molesini, A., Montagna, S., Nieminen, J., Pianini, D., Risoldi, M., Rosi, A., Stevenson, G., Viroli, M., Ye, J.: Developing pervasive multi-agent systems with nature-inspired coordination. Pervasive and Mobile Computing 17, Part B, 236 – 252 (2015), http://www.sciencedirect.com/science/article/pii/S1574119214001904, 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian