This is the peer reviewd version of the followng article:

Developing pervasive multi-agent systems with nature-inspired coordination / Zambonelli, Franco; Omicini, Andrea; Anzengruber, Bernhard; Castelli, Gabriella; De Angelis, Francesco L.; Serugendo, Giovanna Di Marzo; Dobson, Simon; Fernandez Marquez, Jose Luis; Ferscha, Alois; Mamei, Marco; Mariani, Stefano; Molesini, Ambra; Montagna, Sara; Nieminen, Jussi; Pianini, Danilo; Risoldi, Matteo; Rosi, Alberto; Stevenson, Graeme; Viroli, Mirko; Ye, Juan. - In: PERVASIVE AND MOBILE COMPUTING. - ISSN 1574-1192. - STAMPA. - 17:(2015), pp. 236-252. [10.1016/j.pmcj.2014.12.002]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

12/10/2024 06:02

(Article begins on next page)

Developing Pervasive Multi-Agent Systems with Nature-Inspired Coordination

Franco Zambonelli[1], Andrea Omicini[2], Bernhard Anzengruber[3], Gabriella Castelli[1], Francesco L. DeAngelis[4], Giovanna Di Marzo Serugendo[4], Simon Dobson[5], Jose Luis Fernandez-Marquez[4], Alois Ferscha[3], Marco Mamei[1], Stefano Mariani[2], Ambra Molesini[2], Sara Montagna[2], Jussi Nieminen[3], Danilo Pianini[2], Matteo Risoldi[4], Alberto Rosi[1], Graeme Stevenson[5], Mirko Viroli[2], Juan Ye[5]

[1] *Università di Modena e Reggio Emilia, Italy*
[2] *Università di Bologna, Italy*
[3] *University of Linz, Austria*
[4] *University of Geneva, Switzerland*
[5] *University of St Andrews, UK*
[1]*: Corresponding Author: email: franco.zambonelli@unimore.it*
Telephone: +39-0522-522215
Fax: +39-0522-522209

Abstract

Pervasive computing systems can be modeled effectively as populations of interacting autonomous components. The key challenge to realizing such models is in getting separately-specified and -developed sub-systems to discover and interoperate with each other in an open and extensible way, supported by appropriate middleware services. In this paper, we argue that nature-inspired coordination models offer a promising way of addressing this challenge. We first frame the various dimensions along which nature-inspired coordination models can be defined, and survey the most relevant proposals in the area. We describe the nature-inspired coordination model developed within the SAPERE project as a synthesis of existing approaches, and show how it can effectively support the multifold requirements of modern and emerging pervasive services. We conclude by identifying what we think are the open research challenges in this area, and identify some research directions that we believe are promising.

Key words: Pervasive computing, multi-agent systems, coordination

models, self-organization.

---

1. Introduction

The multitude of pervasive computing devices that populate our everyday environments – embedded sensors and actuators, smart phones, personal see-through displays, interactive public displays, and smart objects, to mention only a few – are leading to the emergence of a dense, decentralized infrastructure, which can provide a wealth of services in an un-intrusive manner [1, 2, 3]. This infrastructure will necessarily be open to contributions from a range of vendors and other users, and will be used to deliver and access services for interacting with the surrounding physical world and with the social activities occurring within it [4, 5, 6, 7].

To support such a vision, a great deal of research activity in pervasive computing has been devoted to addressing the problems associated with the development of effective pervasive service ecosystems. The research issues include supporting context-aware composition of service components [8, 9]; enforcing self-adaptability and self-organization in services [10]; and ensuring that service frameworks can be flexible enough to tolerate service diversity and evolution [11]. While a general-purpose approach to support the development of such systems is still missing [12], the key requirements that such an approach should support have been extensively analyzed [11]. They can be re-formulated as follows:

Situatedness — Pervasive services are typically time- and space-dependent, and feature physically- or socially-situated activities. Components of pervasive systems should be able to interact with the surrounding physical and social world by adapting their behaviour accordingly.

Autonomy and self-adaptivity — While individual components should be autonomous in the face of the inherent dynamics of their operational environment [13], pervasive systems should also feature *system-level autonomy* to deal globally with the unpredictability of the environment, providing properties such as self-adaptation, self-management, and self-organization [14].

Prosumption and diversity — Infrastructures for pervasive systems must promote open models of component integration, to be able to

take advantage of the injection of new services and components [15]. This is particularly true in the context of *socio-technical systems* [16], where human users and software agents act as *prosumers* – both consumers and producers – of devices, data, and services.

Eternity — As well as short-term adaptation, a pervasive systems infrastructure should allow for the long-term evolution of organizations, components, and patterns of usage, in order to accommodate technological advances as well as the mutable needs of users without requiring extensive re-engineering effort [17]. In fact, pervasive systems are better conceived as *eternal* systems, engineered for continuous, unlimited service, upgrading, and maintenance over time.

The way to identify suitable approaches for the engineering and development of complex pervasive service systems is to state the problem in terms of *engineering the coordinated activities of a multitude of decentralized autonomous components*: that is, as a dynamic coordination problem for multi-agent systems [15] (MAS from now on). The different hardware and software components that provide pervasive services are decentralized and embedded in a dynamic environment, that is typically controlled by multiple stakeholders (municipalities, industries, private users) and is intrinsically dynamic due to mobility and intermittent availability. Accordingly, their behaviour cannot be subjected to a predictable flow of control, and they should be rather modeled in terms of components exhibiting observable autonomous behaviour, that is as agents [15, 18]. The same considerations clearly apply to those components that by their very nature have autonomous internal decision-making, such as mobile robots, self-driving cars, and humans. Consequently, composite services cannot be engineered on the basis of static patterns of orchestrated activities such as happens in the business process domain. Rather, they must base their activities on a suitable *coordination model* [19] that captures the appropriate dynamic composition patterns, adaptable to the context and situations the system finds itself in, and offering fully decentralized orchestration, within an associated *coordination infrastructure*. *Coordination* has been defined as the science of managing the space of interaction [20]. Coordination models and their associated middleware infrastructures provide the basic abstractions and technologies for dealing with the engineering of complex component interactions, especially in large-scale, open MASs [21, 22] such as are found in pervasive systems.

Among the variety of possible approaches to coordination in modern pervasive systems [12, 19, 23, 24], a very promising one is to take inspiration from nature. Indeed – given the very different kinds of natural systems that effectively coordinate the activities of their components in a context-aware, self-adaptable, and flexible way [11] – one could argue that a coordination model that uses natural metaphors to express distributed algorithms and coordination patterns provides a "natural" way to think about distributed computation.

In recent years a variety of nature-inspired algorithmic solutions and patterns have been applied to deal with specific aspects of self-adaptability and openness in specific distributed and pervasive contexts [14, 25, 26]. Yet a comprehensive framework defining a general-purpose coordination approach to pervasive systems is still missing.

Against this background, this paper presents the following contributions and insights. In Section 2 we introduce the key concepts underlying coordination models and infrastructures, and analyze the suitability of nature-inspired coordination models for the engineering of complex pervasive multi-agent ecosystems. We survey the most relevant proposals in the area of nature-inspired coordination models, and the associated proposals for middleware infrastructures in Section 3. Section 4 presents the SAPERE approach [27], that synthesizes and generalizes existing approaches and was specifically conceived to engineer and support the execution of complex pervasive MASs. As discussed in Section 5, SAPERE makes it possible to easily program a variety of nature-inspired self-organization patterns. We identify some remaining open challenges in Section 6, and identify some promising research directions, before offering some overall conclusions in Section 7.

2. Towards Nature-Inspired Coordination

In this section we introduce the key concepts behind coordination models and infrastructures, focusing in particular on tuple-based coordination models, and motivate the suitability of nature-inspired coordination models for pervasive MASs.

2.1. Coordination Models, Languages, and Technologies

Expressive models, technologies, and methodologies are required to allow programmers and designers to command the complexity and diversity of today's computational systems, which nowadays typically consist of large,

dynamic ensembles of distributed components that are heterogeneous in nature, structure, and behaviour. These include socio-technical components, humans, mobile and embedded devices. Building coherent and dependable systems in such environments is the main concern of research on *coordination models and languages* [28, 29, 30].

Most of the complexity in computational systems comes from *interaction* [20, 31, 32], from harnessing the relations and inter-dependencies between the various system components (humans included). *Coordination models* [24] provide the basic abstractions for harnessing system interaction [33]. Sitting on top of such a model, coordination *languages* and coordination *infrastructures* provide the basic mechanisms and the necessary middleware to implement and deploy coordinated systems. Clearly, coordination models, languages, and technologies have the potential to play a key role in the engineering of pervasive systems [34].

It is worth noting that other meaningful definitions of coordination model have been proposed, each highlighting some key issue of coordination:

- A coordination model is the glue that binds separate activities into an ensemble [19];
- A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed [24]. A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time.

According to the simple meta-model presented by Ciancarini [24], a coordination model defines:

- the *coordination entities*, whose mutual interaction is ruled by the model, also called the *coordinables*;
- the *coordination media*, the abstractions enabling and ruling agent interactions; and
- the *coordination laws*, the rules that govern the space of interaction, controlling the observable behaviour of coordinables, and the computational behaviour of coordination media as well;

In a coordinated MAS, agents play the role of the coordinables, and the coordination media play the role of the social abstractions governing agent interaction, and enforcing coordination laws.

Nowadays, tuple-based coordination models [35] – deriving from the common ancestor of the original LINDA [23] model – represent the most prominent and widespread coordination models, with several implementations from both academia and industry including T Spaces [36], JavaSpaces [37], TuCSoN [38], and GigaSpaces[1]. In a tuple-based coordinated MAS, agents synchronize, co-operate, and compete based on *tuples*, which are simple data structures representing information chunks. Tuples are made available in *tuple spaces*, which are non-indexed shared information spaces working as the *coordination medium*. Computation occurs by accessing, consuming, and producing tuples in an *associative* way, relying on the actual content of tuples and not on any form of naming, addressing, or indexing. An interesting survey of the technologies and platforms for tuple-based coordination by Menezes and Tolksdorf [39] analyzes the suitability of tuple-based coordination systems in supporting openness, unpredictable changes in distributed environments, and several aspects related to adaptiveness: requirements that are of primary importance in pervasive systems.

Most importantly for the current discussion, tuple-based coordination forms an effective basis for nature-inspired coordination.

2.2. Nature-inspired Coordination for Pervasive Systems

As many of the most characteristic human artifacts are *not* nature-inspired, the first question to be answered is: why should we care about nature-inspired models when building computational systems? – and pervasive systems in particular. *Nature-inspired computing* researchers provide a simple yet powerful answer [26, 40]: Natural systems are good in dealing with the *complexity* of coordinating large-scale systems of autonomous agents [41]. Equally important, many properties of complex coordinated natural systems – such as physical, chemical, biochemical, biological, and ethological systems – are essential for computational systems: notably openness, robustness, fault tolerance, and self-adaptation. The issue of properly modeling the dynamics of the interactions to tame the complexity of a system [32] does not only arise in computational and natural systems, but span across many other domains,

[1] `http://www.gigaspaces.com`

from economics to sociology and organization sciences [42].

Yet it is in natural systems that the most suitable answers to the issues of pervasive systems can be found. For instance Grassé [43] notes that in termite societies the coordination of tasks and the regulation of constructions are not directly dependent from the workers, but from the specific mechanism by which workers indirectly coordinate with each other, and that such mechanisms provided for openness, adaptiveness and situatedeness. Many other works on social insects [44] and on biochemical and physical systems [11] show that nature has found elegant solutions to coordination models and mechanisms supporting situatedness openness, and adaptiveness – mechanisms over and above those of long-term evolution through natural selection.

Research activities in the field of natural models reveal some common characteristics that nature-inspired coordination models for pervasive systems should feature: *autonomy*, at the component level to preserve openness; *spontaneous* context-aware triggering of interactions between components; supporting *situated* behaviour; and leading to an overall *self-organising* and *self-adaptive* behavior at the system level [45]. In spite of the fact that tuple-based coordination is definitely *not* a nature-inspired model *per se* [23], it exhibits the characteristics that make it suitable as a basic building block to transpose natural coordination models in computational terms. These include full support for autonomy of components, associative access to tuples by components to support spontaneous triggering of interactions, and support for distributed and context-aware (situated) implementation [46].

3. Overview of Nature-inspired Coordination Models

A number of different nature-inspired models have been proposed in the literature, inspired by a variety of different natural metaphors through which to express coordination patterns suitable for solving some specific coordination problems. Typically, these nature-inspired patterns can be supported either by off-the-shelf middleware or – better – by specifically-developed language and middleware infrastructures, typically realized by extending tuple-based coordination models.

3.1. Stigmergy

The notion of *stigmergy* represents the first acknowledged mechanism of spatial coordination in natural systems, introduced by Grassé [43] as the fundamental coordination mechanism in termite societies. Nowadays, the most

widely-studied example of stigmergic coordination in insect societies is probably that of ant colonies [47]. The basic mechanism is based on *pheromones* that are released in the environment by the ants that find food on their way back to the nest, thus building pheromone trails towards food that other ants are then stimulated to follow. The pheromones act as environment markers for specific social activities, driving both the *individual* and the *social* behaviour of ants.

In the computer science literature, the term "stigmergy" refers to a set of nature-inspired coordination mechanisms mediated by the environment [48, 49]. Agents deposit data into a distributed, shared, environment so as to collectively (yet implicitly) build distributed data structures that can help them navigate in such environments. For instance, *digital pheromones* [50, 51] have been fruitfully exploited as the basic mechanism for coordinating the movements of robot swarms and modular robots [52], and for helping people find directions in an unknown environment [53]. In the area of networking, stigmergy has been exploited to realize effective routing mechanisms in dynamic networks [54, 55].

In terms of middleware infrastructures, stigmergic coordination can be effectively supported by distributed shared network technologies, and in particular distributed tuple spaces [38, 46, 12]. Indeed, middleware infrastructures for distributed network environments have been proposed that extend tuple-based model with the the specific aim of facilitating the expression of stigmergic features (for example through pheromone evaporation and diffusion) [52]. In the case of pervasive systems, stigmergic coordination has also been supported by exploiting pervasive devices such as sensor networks [56] or RFID tags [53, 57].

3.2. Chemical coordination

Another early source of inspiration for coordination was provided by *chemistry*. Chemical reactions can be seen as simple laws regulating the evolution of extraordinarily complex physical phenomena, in some way coordinating the behaviours of a huge number of components, along with the global evolution of complex systems such as biological organisms and meteorological systems.

Gamma [58] was the first and the most prominent example of a *chemically-inspired coordination* model. In Gamma, coordination is conceived as the evolution of a space governed by chemical-like rules, globally

working as a rewriting system [59]. In the CHAM (chemical abstract machine) model [60], states are interpreted as chemical solutions, where floating molecules (representing coordinated entities) interact according to some reaction rules, and where *membranes* constrain the execution of reactions.

Beside having been conceived as general computational model, chemically-inspired computing can be an effective starting point to realize schemes of dynamic service composition [61] or knowledge aggregation [62]. Network protocols for data distribution and aggregation according to chemical models have also been explored, as in the Fraglets approach [63, 64].

Several proposals exist to support service composition based on chemical models [65], including proposals specifically conceived for adaptive pervasive services [66], or to support the adaptive organization of knowledge [62].

3.3. Physical coordination

Inspired by the way in which physical masses and particles move and self-organize according to gravitational and electromagnetic fields, new models of coordination have recently been proposed using a *field-based* metaphor [67]. In field-based coordination models, computational "force fields" – generated either by coordinated components or by the coordination middleware – propagate across a spatial environment, leading to distributed data structures that affect the actions and motions of the agents in that environment.

Co-fields [68] proposes exploiting composite computational fields to coordinate the motion of users and robots in an environment. Other approaches suggest the adoption of virtual force fields to control the shape of self-assembly material [69]. From the perspective of middleware infrastructures, TOTA is a tuple-based middleware explicitly conceived to support field-based coordination for adaptive context-aware and spatially-aware activities in pervasive computing scenarios [70]. The Proto language and middleware [71] exploits field-based coordination to orchestrate the activities of sensor-actuator networks, and the "field calculus" proposed in [72] is accordingly used as a ground for formally studying properties of self-organisation.

3.4. Biochemical coordination

The chemical nature of Gamma is flawed in principle: while their *structure* is reminiscent of chemical reactions, their *behaviour* is that of a rewriting system that is far from resembling real chemical laws. This is not the case with *chemical tuple space* models [73]: in these systems, data, devices, and software agents are uniformly represented in the form of chemical reactants,

and system behaviour is expressed by means of full-fledged chemical-like laws that are both *time-dependent* and *stochastic*, thereby more accurately capturing the essential properties of chemical processes. Chemical tuple spaces can then also embed *probabilistic coordination laws*, thus promoting stochastic behaviour in coordinated MASs.

More specifically, *biochemical tuple spaces* [74] enhance chemical tuple spaces by shaping coordination through distribution and topology. They introduce *compartments* and *diffusion* (somehow borrowing from physical coordination models) as first-class notions for coordination, along with a notion of *neighbourhood*, making it possible to structure the coordination topology by making local spaces available for the execution of chemical reactions.

Biochemical coordination models appear very flexible in enabling the spatial formation of both localised and distributed activity patterns, and have been exploited in many special-purpose pervasive frameworks, including crowd mobility management [75] and participatory sensing [76]. The amorphous computing model can be considered an example of biochemical coordination model [77].

However, to the best of our knowledge no general-purpose middleware infrastructure has previously been developed to support a biochemical coordination model. The SAPERE model and infrastructure supply this, making also possible to support stigmergic, chemical, and physical coordination within a single model and framework.

4. The SAPERE Approach

SAPERE[2] is a nature-inspired coordination model and framework to support the design and development of composite pervasive service systems. Its reference architecture and coordination model synthesize from existing nature-inspired approaches, and is based on an assumption of spatial, local interactions (to be realized via a network of distributed tuple spaces), which is in line with all nature-inspired approaches. Its coordination laws make it possible to express and deploy general nature-inspired distributed algorithms and coordination patterns.

[2]The model was developed as part of an EU-funded research project. See `http://www.sapere-project.eu/`.

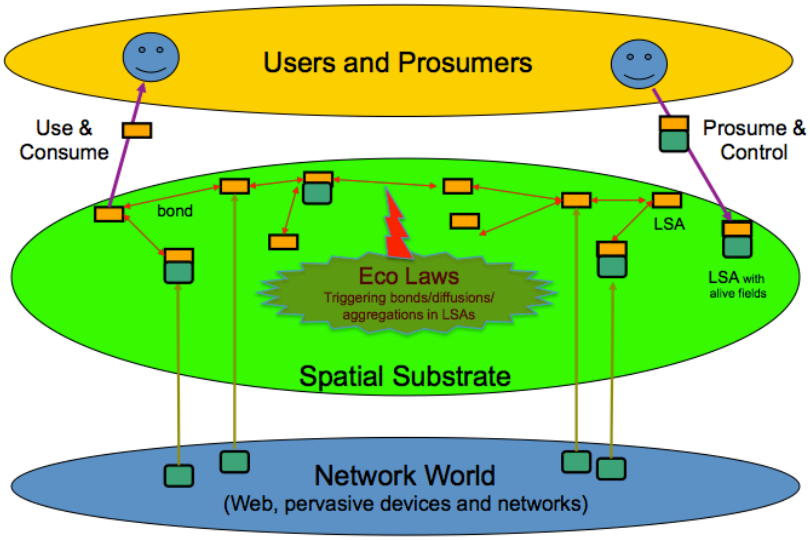


Figure 1: The SAPERE Reference Architecture.

4.1. Reference Architecture and Model Overview

SAPERE abstracts a pervasive environment as a non-layered *spatial substrate* deployed upon a dense network of connected heterogeneous ICT devices (Figure 1). SAPERE acts as a shared coordination medium embodying the basic laws of coordination. These coordination laws govern the interactions between SAPERE agents: we call them *eco-laws*, in line with a nature-inspired terminology. SAPERE agents include all those autonomous components that can provide or request resources and services.

In SAPERE spatial substrate, agents can interact and combine with each other to serve their own individual needs as well as those of the overall environment, so as to define a computational ecosystem. Interactions are mediated through the eco-laws that typically take into consideration the spatial relationships between agents. Users can access the ecosystem in a decentralized way to use and consume data and services, and they can also act as "prosumers" (that is, as both producers and consumers of services) by injecting new agents (data or service components) into the ecosystem.

For the agents in the ecosystem, SAPERE provides common modeling and common treatment. All agents (whether sensors, actuators, services, users, data, or resources in general) have an associated semantic representa-

tion, which is a basic ingredient for enabling dynamic interactions between components. Such annotations, called *Live Semantic Annotations* or LSAs for short, are tightly associated to the agent they describe, and are capable of dynamically reflecting in their values its current situation and context (hence their "liveness"). LSAs act as observable interfaces of resources as well as the basis for enforcing semantic forms of dynamic interactions both for service aggregation and composition, and for data and knowledge management. From a nature-inspired viewpoint, the LSAs of an agent express the *stigma* (analogously to a pheromone or chemical/physical trace) reflecting the existence of an agent in an environment and its current (and possibly past) activities.

The eco-laws define the basic coordination laws driving the virtual "biochemical" interactions among the LSAs of the various agents of the ecosystem. In particular, the idea is to limit interaction on a spatial basis, and rely on diffusive mechanisms to get longer-range networking and composition of data and services. The set of eco-laws includes:

Bonding is the basic mechanism for local interactions and exchange of information between agents. This eco-law acts as a virtual chemical bond (based on semantic pattern-matching) between two LSAs (and thus, between their associated agents);

Spreading diffuses LSAs on a spatial basis, and is necessary to support the propagation of information and interactions among remote agents and to realize coordination structures such as fields and chemical gradients;

Aggregation provides catalysis among LSAs, to support distributed data aggregation;

Decay mimics chemical evaporation to garbage collect data.

It is perhaps surprising that these four eco-laws are sufficient to realize a wide variety of nature-inspired coordination schemes, including those physically-, chemically-, and biologically-inspired, within the *same* framework and with the *same* basic programming approach.

Following its natural inspiration, adaptivity in the SAPERE approach is not in the capability of individual components, but rather in the overall self-organizing dynamics of the ecosystem. In particular, any change in the system, and any change in its components or their context, are reflected by

dynamic changes in their LSAs and so will be reflected in the firing of new eco-laws. This can lead to new bonds or aggregations, and/or to the breaking of existing bonds between components.

4.2. The SAPERE Middleware

To turn this reference architecture and model into an operational one, a middleware infrastructure needs to provide an active space in which to store the continuously-updating LSAs of agents, and to support the matching process that triggers eco-laws that allow the components of the system to interact.

At its lowest level, the SAPERE middleware is formed by a network of nodes, each hosting a local LSA space, with neighbor relations typically shaped by spatial or network relations. The LSA space is a local tuple space which hosts LSAs represented as tuples. The shape of the actual network of connections is determined by a reconfigurable component, which can be based on a strategy that connects nodes based on spatial proximity, social proximity or both [78]. The shape of such a network determines the paths along which LSAs on a node can propagate and diffuse to other nodes: nothing fundamentally precludes long-range, non-spatial neighborhoods.

Whenever an agent (corresponding to a device, a sensor, a service, or an application agent) is executed in a node, its own LSA is automatically injected into the LSA space of that node, making the component part of that space and of its local coordination dynamics. The LSA can also indicate its desire to propagate and diffuse through the network of LSA spaces. When an agent is removed from a node, its LSA is automatically removed from that space.

In the current implementation of the middleware (available under open-source licence for desktop and server Java platforms[3], and for Android[4]) any device with sufficient computational power to handle an LSA space can serve as a node of the network. A network can therefore consist of personal computers, tablets or smartphones, interactive displays, or embedded sensors (of adequate power). From the operational perspective all SAPERE nodes are at the same level, since the middleware code they run can support the same services and provide the same set of functions. Despite such notional

[3] `http://bitbucket.org/gcastelli/saperemiddleware-javase`
[4] `http://bitbucket.org/gcastelli/saperemiddleware-android`

equality, for practical purposes nodes might implement different behaviours to account for particular service requirements, hardware capabilities or limitations. For example, a node on a mobile device might assume a behaviour accounting for constrained power supply and computational power, setting device sensors to tolerable sample rates.

For individual agents, the middleware provides a simple API to let agents advertise themselves *via* an LSA and support its continuous updating. In addition, the API enables agents to detect local events, such as the modification of LSAs or the triggering of eco-laws.

From the viewpoint of the underlying infrastructure, the middleware transparently absorbs dynamic changes caused by the arrival and departure of the devices without affecting the perception of the spatial environment by individual agents. The middleware is also able to detect events on LSAs and trigger the applicable eco-laws. Coordination therefore occurs when a change in LSA causes an eco-law to fire and bring about an interaction concerning one or more LSAs in the local LSA space.

Eco-laws are realized as a set of rules embedded in SAPERE nodes. For each node, the same set of eco-laws applies. Eco-laws' computations will be described more in detail in the following section.

We emphasize that the SAPERE middleware can host additional libraries in the forms of agents to be installed in the nodes to enrich services offered by the middleware. Such library agents access the system by means of the same API and in terms of LSAs, at the same level as any other agent.

4.3. A Coordination Model Based on Eco-laws

The eco-laws support self-adaptive and self-organising activities in the ecosystem. Eco-laws operate on a pattern-matching schema: they are triggered by the presence of LSAs matching with each other, and manipulate the matching LSAs (and the fields within them) according to a mechanism of artificial chemistry [11, 79].

We now describe the structure of LSAs, the middleware API, and the principles underlying the set of basic eco-laws.

4.3.1. LSAs and API

Any component or agent that takes part in a SAPERE ecosystem is represented by one or more LSAs, a representation that is strictly linked to its corresponding agent. On the one hand, the LSA reflects in real time the internal state of its associated agent, enabling the entity to actively take

part to the system dynamics and ecosystem evolution over time. On the other hand, LSAs are manipulated by the eco-laws running the ecosystem that can (for instance) establish or dissolve connections with other LSAs.

LSAs are realized as descriptive tuples made up of a number of fields in the form of name/value pairs. For instance, the following LSA: `LSA1(sensor-type = temperature; accuracy = 0.1; temp = 45)` could represent the LSA of a temperature sensor, expressing the actual temperature value and its accuracy.

By building on and extending tuple-based models [23], the values in an LSA can be *actual* (yet possibly dynamic and changing over time, as it is the case with the temp field in `LSA1`, which makes LSAs live) or *formal*, not tied to an actual value and representing a dangling connection (typically represented with a "`?`").

The fields in an LSA can be organized in a hierarchical fashion: the value of a property can be a set of properties again. Also, in the actual SAPERE system, field names and values can be expressed using ontologies. We omit these details for clarity, and forward the reader to [80] for a detailed description.

In the SAPERE model, the idea is that each agent takes care of initializing at least one LSA, injecting it into the LSA space, and keeping the values of the LSAs fields updated to reflect its current state. Each agent can modify *only* its own LSAs, but can read any LSAs which have been linked to its own LSAs by the bonding eco-law. LSAs can also be manipulated by eco-laws, as explained in the following sections.

Pattern matching between LSAs – which is the basis for triggering eco-laws – happens when the values of properties with corresponding names match. As in classical tuple-based approaches, a formal value matches with any corresponding actual value.

As an example, the presented `LSA1` could match an `LSA2` of the kind `LSA2(sensor-type = temperature; temp = ?)`, which expresses a request for acquiring the current temperature value. Any additional properties present in `LSA1` (e.g., `accuracy`) are not taken into account by the matching function because it considers only an inclusive match.

At the middleware level, an API is provided to agents to let agents inject an LSA into the tuple space (`injectLSA(LSA myLSA)`); to let agents atomically update some fields of an LSA to keep it alive (`updateLSA(field = new-value)`); and to execute event handlers (`onEcoLawEvent(...)`) to sense whatever events occur on its LSAs (spe-

```
AgentTemperatureSensor {
    init() {
        float tl = sample();
        injectLSA(sensor-type = temperature; accuracy = 0.1; temp = t1)
    }

    run() {
        while(true) {
            sleep (100);
            float tl = sample();
            updateLSA(temp = tl)
        }
    }
}
```

Figure 2: An agent that acts as a temperature sensor, injects an LSA with the temperature level during initialization, and periodically updates it.

cialized in different ways such as the `onBond(LSA mylsa)` method to detect when an LSA is being bonded to another). As an example, Figure 2 represents the code (simplified for the sake of readability) of a temperature-sensing agent that injects the LSA with the temperature level and periodically updates it.

Unlike LINDA, SAPERE does not distinguish between tuples and anti-tuples (or templates). Consequently, the `injectLSA()` operation subsumes LINDA's `rd` and the `out` operations thanks to the specific operation of the bond eco-law, described next.

4.3.2. The Bonding Eco-law

Bonding is the primary form of interaction among co-located agents (within the same LSA space) in SAPERE. In particular, bonding can be used to locally discover and access information, as well to get in touch with and access local services – all of which is achieved by a single, versatile mechanism. The bonding eco-law realizes a virtual link between LSAs whenever two LSAs (or sub-descriptions within) match.

The bonding eco-law is triggered by the presence of formal values in at least one of the LSAs involved. Upon a successful pattern-match between the formal values of an LSA and actual values of another LSA, the eco-law creates the bond between the two. The link established by bond in the presence of "`?`" formal fields is bi-directional and symmetric. Once a bond is established both the agents holding the LSAs are notified of the new bond and can trigger actions accordingly. After bond creation, the two agents holding the LSAs

```
Agent AccessTemperatureInformation {
    init() {
        injectLSA(sensor-type = temperature; temp = ?);
    }
    onBond(LSA b) {
        float tl = b.temp;
        print("current temperature = "+ tl);
    }
}
```

Figure 3: An agent that injects an LSA which matches with that of the temperature sensor and enables it to access the corresponding temp information.

can read each other's LSAs. This implies that once a formal value of an LSA matches with an actual value in an LSA it is bound to, the corresponding agent can access the actual values associated with the formal ones as well as any other properties in the bonded LSA.

As an example, and recalling the temperature-sensing agent of Figure 2, the agent in Figure 3 injects an LSA that matches the LSA in Figure 2, thus enabling it to access the corresponding temperature information.

In the same way that bonding is automatically triggered upon match presence, bond disruption takes place automatically whenever a change in the "live" values of a bonded LSA mean that the match conditions no longer hold.

In addition to the "`?`" formal field, which establishes a one-to-one bidirectional bond between components, SAPERE also makes it possible to express a "`*`" formal field, which leads to a one-to-many bond with multiple matching LSAs.

Finally, the "`!`" formal field expresses a field that is formal unless the other "`?`" field has been bonded. This makes it possible for an LSA to express a parameterised services, where the "`?`" formal field represents the parameter of the service, and the "`!`" field represents the answer that it is able to provide once it has been filled with the parameters.

As a very simple example, consider an agent that is able to classify temperature values into labels expressing the comfort level, such as "cold" an "warm". To do that, the agent would need a temperature value as input and would return a label as output. This could be expressed by the LSA `LSA3(agent-type = temp-converter; temp = ?; comfort = !)`. In the presence of a matching LSA, such as the presented `LSA1`, `LSA3` would bind

with `LSA1`. Consequently, the agent holding `LSA3` would automatically turn the `comfort` field into an actual value on the basis of the information read in `LSA1`.

The bonding eco-law mechanism enables two agents to discover each other spontaneously and exchange information, all with a single operation after both have injected an LSA into the space, and without any prior information about each other's existence other than a shared ontological space of field names. Unlike in traditional discovery of data and services [81], bonding makes it possible to compose services without distinguishing between the roles of the involved agents, and subsumes the traditionally separated phases of discovery and invocation.

4.3.3. The Aggregation Eco-law

The ability to aggregate information to produce high-level digests of some contextual or situational facts is a fundamental requirement for adaptive and dynamic systems. In fact, in open and dynamic environments, one cannot know *a priori* which information will be available: some information sources may disappear, others may appear. Providing a means to extract a summary of all available information without having to explicitly discover and access the individual information sources is therefore very important.

The aggregation eco-law is intended to aggregate LSAs together to form summaries of the current system's context. An agent can inject an LSA with `aggregation_op` and `type` properties. The `aggregation_op` property identifies a function to base aggregation upon. The `type` property identifies which sorts of LSAs to aggregate. In particular it identifies a numerical property of LSAs to be aggregated.

For example, the LSA `LSA4(aggregation_op = max; type = temp)` will trigger the aggregation eco-law that selects all the LSAs having a `temp` type; computes the maximum value among them; and modifies the LSA with the result (i.e., creating a new field in the LSA space, `aggregation_result = 57` – 57 being the computed value). In the current implementation, the aggregation eco-law is capable of performing most common order- and duplicate-insensitive (ODI) aggregation functions [82].

The aggregation eco-law supports separation of concerns and allows the re-use of previous aggregations. On the one hand, an agent can request an aggregation process without dealing with the actual code to perform the aggregation; on the other, the LSA resulting from an aggregation can be read (by means of bonding) by any other agent that wishes to acquire the

pre-computed result.

Beside ODI functions, which can be performed by the aggregation eco-law in isolation, other forms of distributed computations can be realized by properly combining the activities of eco-laws and agents.

4.3.4. The Decay Eco-law

The decay eco-law enables the removal of components from the SAPERE environment. The decay eco-law applies to all LSAs that specify a `decay` property, updating their time-to-live according to a specific decay function, and removing LSAs that, based on this decay property, have expired. For instance, the LSA `LSA5(sensor-type = temperature; temp = 10; decay = 10000)` will expire and be deleted ten seconds (10000ms) after injection.

The decay eco-law is the basis for a garbage collector capable of removing LSAs that are no longer needed in the ecosystem or that are no longer maintained by a component, for instance because they have been copied into an LSA space through spreading.

4.3.5. The Spread Eco-law

The eco-laws presented so far act on a local basis, on LSAs within a single LSA space. As the SAPERE model is based on a network of spaces, it is fundamental to enable interactions between spaces, specifically by providing a mechanism to send information to remote LSA spaces and make it possible to distribute information and results across LSA spaces, and consequently to promote coordination of activities among distributed agents.

In SAPERE design, we wanted to avoid direct actions in remote spaces, which introduces semantic and implementation problems that seem resistant to a clean and nature-inspired solution. To this end, in SAPERE we designed a spread eco-law to diffuse LSAs to remote spaces. One of the primary usages of the spread eco-law is to enable searches for components that are not available locally, and conversely to enable the remote advertisement of components.

For an LSA to be subject to the spread eco-law, it has to include a `diffusion` field, whose value (along with additional parameters) defines the specific type of propagation. The SAPERE framework implements two different types of propagation: direct propagation used to spread an LSA to a specified neighbor node,

`LSA6(...diffusion_op=direct; destination=node_x; ...)`; and general propagation to propagate an LSA to all neighboring nodes, `LSA7(...diffusion_op=general; hop = 10; ...)`, where the `hop` value can be specified to limit the distance of propagation of the LSA from the source node.

Clearly, the nature of the relations between the LSA spaces (whether related to spatial proximity, or social proximity, or other relations that can be supported by the middleware [78]), determine the actual semantic of the "hops" that determine the extent to which an LSA spreads.

Concerning scalability, it is clear that not giving any bound (or giving too large bound) to the distance at which LSAs could spread, may induce scalability problems: the resulting patterns of coordination would involve too many agents and too far from each other. However, the spatial nature of pervasive applications typically suggests the adoption of coordination patterns relying on spatial proximity and, consequently, reasonable limits on the spreading distance for LSAs.

General diffusion of an LSA to distances greater than one is a multicast that induces a large number of replicas of the same LSA to reach the same nodes multiple times from different paths. To prevent this, general diffusion is typically coupled with the aggregation eco-law so as to merge together multiple copies of the same LSA that arrive on a node from different paths, and/or with decay to remove replicas after some time.

5. From Eco-laws to Nature-inspired Patterns, Self-awareness, and Applications

The eco-laws form a necessary and sufficient core to support a large set of self-organizing, nature-inspired interactions [10].

On the one hand, our experience and analysis show that the four eco-laws are *necessary* to express known self-organization patterns. Bonding is the elementary mean to support adaptive local service interactions in SAPERE, subsuming the necessary phases of discovery and invocation of traditional service systems. Spreading is necessary to diffuse information in a distributed environment to enable distributed interactions. Aggregation and decay are necessary to support decentralized adaptive access to information without being forced to dynamically deploy code on the nodes of the system, which may not be possible in decentralized environments. Such informal considerations supply the lack of formal proofs that we will try to build in the

future.

On the other hand (and again relying on extensive experience rather than on formal arguments) the eco-laws are *sufficient*. That is, they are expressive enough to support a large variety of self-adaptive and self-organizing coordination schemes, including nature-inspired interactions in open pervasive service ecosystems and "unnatural" advanced forms of distributed situation awareness [83, 84]. This does not mean that *only* nature-inspired algorithms can be implemented. On the contrary, binding and spreading can be trivially used to realize local and distributed client-server interaction schemes, including those used to dynamically connect to sources of contextual information, as well as asynchronous models of interaction and information propagation. By coupling spreading with aggregation and decay it is possible to realize distributed data structures that effectively support all patterns of nature-inspired adaptive and self-organizing behaviour. Most of these patterns, in fact, rely on the possibility of computing distributed data structures to express virtual physical fields, or digital pheromones, or virtual gradients: structures that can be built quite intuitively as LSAs and eco-laws.

In this section we exemplify how, by combining eco-laws, it is possible to realize some exemplary nature-inspired self-organizing patterns, as well as a variety of self-awareness patterns, and that can ultimately be used to develop several applications.

5.1. Self-organization Patterns

The bonding eco-law is the basic interaction mechanism for composing *local* data and service structures. On top of it, the other three eco-laws can be used to compose *distributed* structures [10]. The aggregation and spreading eco-laws can be combined in different ways to form a variety of structures.

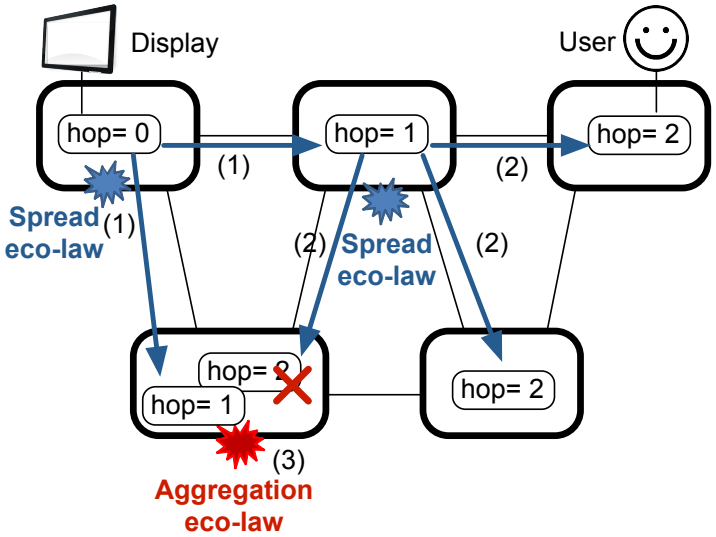
As a first example, aggregation applied to the multiple copies of diffused LSAs can reduce the number of redundant LSAs so as to form a distributed *gradient* structure, also known as *computational force field*. In general field data-structures are a useful tool to encode and spread information in a distributed system. The main point of using them is that they effectively provide adaptive spatial awareness to agents. Fields in fact provide both a measure of distance in the network (by means of hop count from the source) and a measure of the direction from where the information comes from (by considering the slope of the hop counts). Such information is very useful in a number of pervasive applications that are closely coupled with the location of the agents.

For example, a field data structure can be used to coordinate the motion of people in an environment [85], where agents running on users' smart phones give directions by accessing the information on LSAs that are spread as fields in the environment. Considering the example in Figure 4, the `Display` agent spreads a field with the `hop` number expressing distance (in terms of network hops) from it. The `User` agent can then follow the gradient of the `Display` LSA to reach the location of the display. This approach is based on strictly local information – the local shape of the field – and is adaptive to changing conditions – a moving display or a corridor made inaccessible – as the field of the `Display` would reshape itself accordingly.

As a second example, spreading and aggregation can be used together to produce distributed self-organized aggregation, dynamically computing some distributed property of the system (e.g., some value of sensors spread in the systems and represented as a set of distributed LSAs) and making the results of such computation available at each and every node of the system [82]. Distributed aggregation is a basic mechanism by which to realize forms of distributed coordination such as consensus-finding, distributed task allocation, and behaviour differentiation. As an example, Figure 5 shows how an agent X can trigger a distributed aggregation of temperature sensors, so as to determine what is the maximum value of the temperature sensed in the surroundings.

As a third example, gradients and fields can be the basis for building pheromone-like data structures driving agent activities [25]. These data structures, mimicking pheromone trails in natural systems like ant colonies, are deployed by mobile agents, and provide local information on how to explore the distributed environment. In particular, while general diffusion and progressive decay can be used to realize diffusible and evaporating pheromone-like data structures, direct propagation can be used to navigate by following pheromone gradients. For example, in a trade fair scenario with multiple displays providing information, a user agent finding some interesting information can start spreading a pheromone trail to allow other agents to easily reach the source (display) of that information by following the trail. Pheromones can be realized as LSAs locally deposited by agent as they move across the network: accordingly, such LSAs would be deployed along the agent's path. Pheromone LSAs would also be associated with the decay eco-law to emulate pheromone evaporation.

Finally, in a more general way, these patterns are provided as services [86] that can be combined with each other to provide more powerful services. A

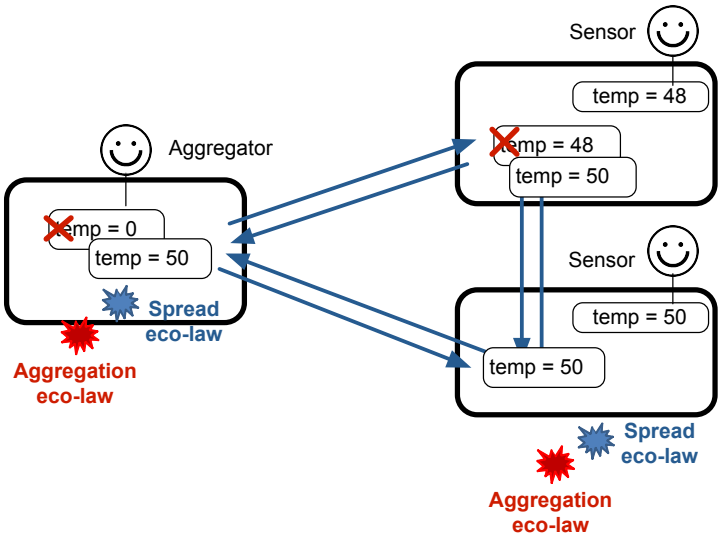



```
Agent Display {
    init() {    injectLSA(name = display, diffusion_op = general,
                hop = 1, aggregation_op = min, previous = local)
}   }

Agent User {
    init() {    injectLSA(name = display, hop = *)   }

    onBond(LSA b) {     float d = computeDistanceFromHop(b.hop)
                        print("display distance = "+d)
                        print("go toward "+b.previous)
}   }
```

Figure 4: Generating and navigating distributed data structures. The agent `Display` uses the spread eco-law combined with aggregation to create field-like data structures, that agent `User` can then detect and follow downhill.




```
Agent X {
    init() {    injectLSA(aggregation_op = max, property = temp, diffusion = general,
                hop = 1, previous = local)
}   }

Sensor 1....N {
    init() {    float t = sample()
                injectLSA(temp = t)
    }

    run() { while(true) {   float t = sample()
                            updateLSA(temp = t)
}   }                   }
```

Figure 5: Distributed aggregation. Many temperature sensors (`1...N`) exist in the ecosystem. An agent `X` can inject an LSA that, by combining spreading and aggregation, can adaptively compute the maximum temperature of sensors.

typical example is provided by a remote query and retrieval information service, which uses a combination of a gradient spreading (static or dynamic) followed by chemotaxis (itself implemented as a pattern using the four core eco-laws, and described in more detail below). This service has been developed and used on several occasions within the SAPERE infrastructure: for progressive self-aggregation of context information [87], for progressive self-aggregation of situation information [80], and for semantic resource discovery [88].

5.2. Situation-aware Coordination

The previous coordination patterns rely on exploiting eco-laws to self-organize adaptive distributed data structures upon which to base the coordination of distributed SAPERE agents. From a different viewpoint, one can view eco-laws (and the possibility of building self-organized data structures) as a means to manipulate and process contextual information so as to enable agents to reach higher levels of self-awareness in their individual and coordinated activities.

At the most basic level, the bonding eco-law provides agents with the means to expand the boundaries of their personal information space, and inspect (but not modify) the static and dynamic information embedded in the LSAs of co-located individuals to which they are bonded. This provides the simplest form of perception and spatial awareness through local interactions. Similarly, the spread eco-law works by replicating an LSA across neighbor devices, bounded by a metric such as hop distance or physical position. From the awareness viewpoint, the spread eco-law allows agents to project their state across a region of the environment by setting spreading and hop-distance properties in their LSAs to trigger the reaction. An agent may bind with LSAs being spread from a distant source (as they would any local LSAs) to perceive their content, including the distance from the original source, thus making the gradient pattern an effective mechanism to express advanced forms of location awareness.

In addition, the contextual information embedded in LSAs – other than made locally available to other agents through bonding and spreading – can also be processed, pruned, and aggregated in distributed and self-organizing ways, by properly combining the different eco-laws. It is also possible to provide forms of distributed reasoning and overall to increase the ability of distributed agents to become aware of the situations of which they are part [89, 90].

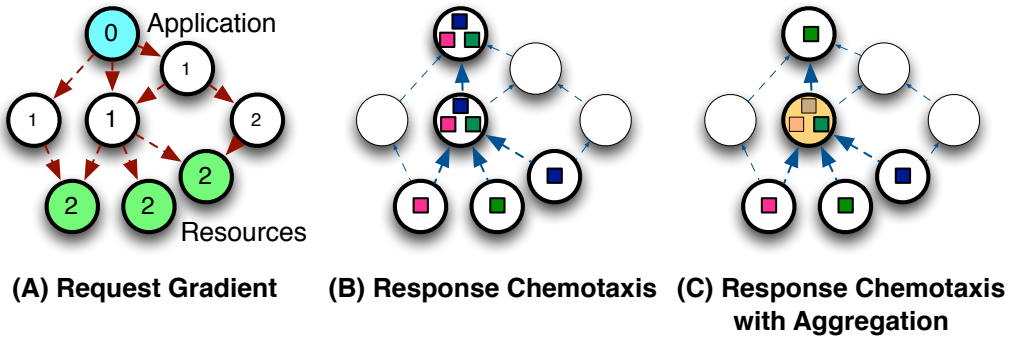


Figure 6: Context acquisition interactions: (A) a request gradient is spread, (B) replies ascend the gradient, (C) optional aggregation – and possibly distributed reasoning – reduces transmission costs and resolves conflict.

An example of particular relevance consists in exploiting the chemotaxis pattern [10] mentioned earlier. This can be thought of as a directed form of spreading, and provides a mechanism to perform routing of information based on established gradients. Information is routed towards the source agent of a gradient along the shortest accessible path from within the gradient's extent. A typical usage scenario (Figure 6) sees a situation classifier agent seeking inputs through a set of requests each embedded in an LSA in which to set the necessary properties to trigger the construction of a gradient: *(A)* spreading and aggregation patterns iteratively fire, establishing a gradient data structure in which each LSA has a pointer to the node in which it was created, which is also the node indicating direction to the source following shortest path; *(B)* where a request gradient LSA reaches a node with a matching LSA, a reply LSA is generated pointing towards the request and routed towards the classifier agent using chemotaxis; *(C)* as the LSAs that contain the information move amongst the SAPERE nodes by climbing the gradient, data can be dynamically aggregated via the aggregation ecolaws [87], and it can also be processed on the fly by applying reasoning and classification techniques in a fully distributed way [88].

5.3. Developing SAPERE Applications

Developing a SAPERE application consists of programming a set of agents (according to templates and guidelines provided in the SAPERE middleware package) encoding specific functionalities and services, and expressing LSAs that makes it possible for them to coordinate with each other and

with the agents (devices, components, resources) that already exist in the ecosystem.

More specifically, accessing pervasive services in SAPERE, e.g., from a smart phone, firstly implies launching a specific SAPERE app. The SAPERE app puts a local instance of the SAPERE middleware in execution that will start looking for other SAPERE nodes to connect to in order to enter the overall pervasive ecosystem. The app presents the users with GUI elements that abstract the concept of an LSA but make available the services offered by the ecosystem. To add new services on a node, the developer has to design an agent that implements the service and a set of LSAs that describe the location and interface to the offered service. Once the local node is part of a neighbourhood of devices, the LSAs are automatically propagated through the ecosystem.

SAPERE naturally suits application scenarios in which users with a smartphone (or, more generally, a smart device) that are moving in an environment (e.g, a city, an exhibition, some event) can access the SAPERE ecosystem and its services to get a richer and more informed experience of interacting and understanding such environments. Within that general context, to test the soundness and applicability of the SAPERE approach, we have focused on "ecosystems of displays" scenarios – i. e., scenarios in which we have an environment densely populated by privately owned display devices (such as smartphones) but also by interactive public displays (such as wall mounted displays enriched with the sensing capabilities to detect users and devices). Such displays can become SAPERE nodes and can, in a coordinated way, effectively provide adaptive information and steering proposals to users.

This was demonstrated in the context of the 2013 Vienna City Marathon (VCM) – a real-world instantiation of a SAPERE ecosystem in the city centre of Vienna, Austria in the spring of 2013 – involving (see Figure 7): a SAPERE app to be downloaded by runners and spectators that instantiates a SAPERE node on the phone, and a number of public displays distributed over VCM hotspots, each implementing a dedicated SAPERE node, that are capable of interacting amongst themselves and with the SAPERE nodes on users' mobile phones. The entire ecosystem that was created for this instantiation consisted of 5192 mobile SAPERE nodes that exchanged LSAs amongst themselves as well as with 7 public SAPERE nodes that were propagating information at key locations along the marathon course.

The challenges of developing a SAPERE ecosystem on this scale mostly

related to the design of agents, services and their respective LSAs. Due to the nature of SAPERE nodes, LSAs were exchanged as a result of the spatial collocation of devices – i. e., if a SAPERE node detected another node in its proximity it propagated its service and data LSAs to this node and vice versa. As a result, the ecosystem topology was fully decentralized and – over the course of the event – 594.702 messages were exchanged and propagated in an opportunistic fashion. This system topology required agents to be able to: *(i)* deal with potentially outdated information, *(ii)* account for circles in the current topology, *(iii)* process LSAs and forward them to their neighbours in the available time frame as well as *(iv)* notify the users about the generated information and displaying it using the developed SAPERE app and the public displays.

During the event, the instantiated ecosystem was used to provided adaptive services to marathon users relating to: exploiting public SAPERE nodes to deliver contextually relevant, personalized information to passing-by users; help people to find like-minded individuals in their immediate surroundings; computing, by means of the the distributed public display nodes, the estimated density of people around the VCM track, and providing this information to security forces and marathon organizers to avert potential crowd disasters (the VCM is usually frequented by around 200.000 spectators). These services were implemented using the introduced SAPERE nodes which host agents that inject and propagate LSAs containing (or requesting) information about *(i)* the local node's context (e.g., location, users' interests, current neighbours) as well as *(ii)* about the information and services available on the public displays (advertising, suggesting locations and directions, information about other people).

The VCM demonstration, along with other applications that we have implemented and tested (see [77] and [78] for details) have helped us verify the effectiveness and versatility of the SAPERE approach and technology.

6. Open Issues and Research Directions

Despite the promises of nature-inspired approaches to coordination for pervasive systems, and despite the sound synthesis promoted by the SAPERE approach, a number of fascinating research challenges are still to be met to make nature-inspired coordination a widespread and easy-to-adopt approach.

Figure 7: The SAPERE mobile phone app (left) and a SAPERE public display in the VIP lounge (right)

6.1. Assessing the Advantages of Nature-inspired Coordination

Any new approach to software development must come along with proper software engineering methodologies and tools to support the activities of designers and developers, as well as to facilitate control over the execution of pervasive MAS.

In the SAPERE project, we have extensively worked towards the elaboration of a specific SAPERE methodology [91], with the aim of defining guidelines for application development, and have also tried to analyze and compare the key differences and the pros and cons of developing applications using a nature-inspired coordination model rather than more traditional message-based interaction protocols [92].

From our analysis, it appears that nature-inspired coordination can be indeed simpler to program in the presence of dynamic environment and a large-number of components. Yet, we think there will be need of more practical experience with SAPERE and, in general, with nature-inspired approaches, to get a real understanding of the implications of programming with such approaches, and of their advantages over more traditional ones.

6.2. Top-Down vs. Bottom-Up Adaptation

Another key challenge in the engineering of nature-inspired coordination, is understanding the trade-offs between the power of top-down adaptation versus bottom-up one.

A relevant thrust of research on self-adaptive and evolvable software systems – rather that looking for inspiration from the area of natural systems, with the aim of reproducing the emergent self-adaptation capabilities emerging in a bottom-up way from local interactions [25] – is focussing on integrating adaptation in software systems according to the most assessed approaches of software engineering, that is, by explicitly encoding adaptation in system designed and coordinated in a top-down way [93].

The key question that arises, in this context, is how it is possible to define methodologies to smooth the tension between the two approaches, i.e., identifying how the two approaches can co-exist and possibly conflict in future systems. The ultimate goal would be to tolerate development methodologies in which the bottom-up endeavor of nature-inspired coordination can become part of a more traditional top-down approach to software engineering.

6.3. Predicting and Controlling Emergent Behaviors

The process of emergent bottom-up self-organization in natural systems, leading to self-adaptive properties, is by definition non-deterministic and irreducible. Although it is possible to design a system that will *probabilistically* behave as desired, it is impossible to exactly predict its final configuration but by executing the system itself.

Probabilistic non-determinism may be satisfactory in some non-critical cases (e.g., in gossip-based diffusion of non-critical information in P2P systems [94] – where the existence of some nodes not reached by the information is not critical). However, in other cases it is not acceptable (e.g., exploration of an environment by a swarm of robots in a rescue operation [95] – where one cannot tolerate the swarm to ignore some portion of the environment). Accordingly, a key issue is to compensate such unpredictability by defining control tools to dynamically tune on-the-fly the coordination laws of self-organizing systems whenever heading toward undesirable states [96].

Some research in software engineering and distributed systems explicitly addresses this general issue, and mostly at the level of simple simulations for multi-agent systems or cellular automata. Yet, a general understanding of how to control emergent behaviors in complex software systems is still to

be reached. In our opinion, the work on regulated norm-based multi-agent systems and electronic institutions [97, 98] can be an effective starting point.

6.4. Coordination for Simulation Frameworks

The role of computational simulation in the engineering of complex artificial systems – and complex computational systems specifically – is nowadays widely acknowledged [99]. What is yet to be clearly understood, is the potential role of coordination models and technologies to work as the core of non-trivial simulation frameworks.

Modeling the interaction among system components, in fact, is a fundamental issue in complex system simulation [100]. Simulation frameworks based on coordination models – that is, handling interaction with first-class coordination abstractions – would then suit well complex system simulation; those based on nature-inspired coordination models would be particularly well-suited for the simulation of complex natural systems.

Accordingly, a challenge for forthcoming agent-based simulation frameworks will be integrating suitable (nature-inspired) coordination abstractions, capable of capturing the full articulation of system interaction, possibly including complex stochastic behaviours: in [100], for instance, biochemical tuple spaces are applied to the simulation of complex interaction patterns of intracellular signalling pathways. Along this line, we can envision future research scenarios where coordination models and technologies – in particular, nature-inspired ones [101] – play a central role in the development of rich agent-based simulation frameworks.

7. Conclusion

In this paper, we have argued that nature-inspired coordination models can be well-suited to meeting the challenging requirements of pervasive environments. We have surveyed a number of nature-inspired coordination approaches that have been proposed so far, and have presented the SAPERE approach to coordination as a general-purpose synthesis that provides a common platform on which to develop nature-inspired pervasive service systems according to a variety of nature-inspired approaches.

Our experience with SAPERE represents a first solid building block towards the widespread exploitation of nature-inspired coordination for pervasive systems. Still, there remain a number of research challenges to tackle,

each of which will help paving the way towards a fascinating nature-inspired pervasive computing future.

References

[1] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, F. Zambonelli, Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence, Pervasive and Mobile Computing 8 (1) (2012) 2–21.

[2] F. Zambonelli, Toward sociotechnical urban superorganisms, IEEE Computer 47 (8).

[3] D. Harnie, T. D'Hondt, E. G. Boix, W. De Meuter, Programming urban-area applications, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, ACM, 2012, pp. 1516–1521.

[4] G. Cabri, L. Leonardi, M. Mamei, F. Zambonelli, Location-dependent services for mobile users, IEEE Transactions on Systems, Man, and Cybernetics, Part A 33 (6) (2003) 667–681.

[5] C. Bisdikian, B. Mitschang, D. Pedreschi, V. S. Tseng, C. Bettini, Challenges for mobile data management in the era of cloud and social computing, in: 12th IEEE International Conference on Mobile Data Management, 2011, p. 6.

[6] P. Lukowicz, S. Pentland, A. Ferscha, From context awareness to socially aware computing, IEEE Pervasive Computing 11 (1) (2012) 32 –41.

[7] D. Schuster, A. Rosi, M. Mamei, T. Springer, M. Endler, F. Zambonelli, Pervasive social context: Taxonomy and survey, ACM Transactions on Intelligent Systems and Technology 4 (3).

[8] C. Bisdikian, I. Boamah, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. L. Yeh, V. Rasin, H. Huang, C. Simonds, Intelligent pervasive middleware for context-based and localized telematics services, in: Workshop Mobile Commerce, 2002, pp. 15–24.

[9] J. Bronsted, K. Hansen, M. Ingstrup, Service composition issues in pervasive computing, IEEE Pervasive Computing 9 (1) (2010) 62–70.

[10] J. L. Fernandez-Marquez, G. D. M. Serugendo, S. Montagna, M. Viroli, J. L. Arcos, Description and composition of bio-inspired design patterns: a complete overview, Natural Computing (2012) 1–25.

[11] F. Zambonelli, M. Viroli, A survey on nature-inspired metaphors for pervasive service ecosystems, Journal of Pervasive Computing and Communications 7 (2011) 186–204.

[12] V. Raychoudhury, J. Cao, M. Kumar, D. Zhang, Middleware for pervasive computing: A survey, Pervasive and Mobile Computing 9 (2) (2013) 177 – 200.

[13] M. Ganzha, M. Paprzycki, A. Omicini, Software agents: Twenty years and counting, Computing Now 6 (11).

[14] M. Mamei, R. Menezes, R. Tolksdorf, F. Zambonelli, Case studies for self-organization in computer science, Journal of Systems Architecture 52 (2006) 443–460.

[15] F. Zambonelli, A. Omicini, Challenges and research directions in agent-oriented software engineering, Autonomous Agents and Multi-Agent Systems 9 (3) (2004) 253–283.

[16] A. Omicini, P. Contucci, Complexity & interaction: Blurring borders between physical, computational, and social systems. Preliminary notes, in: Computational Collective Intelligence. Technologies and Applications, Vol. 8083 of LNCS, Springer Berlin Heidelberg, 2013, pp. 1–10, 5th International Conference (ICCCI 2013). Craiova, Romania, 11–13 Sep. 2013, Proceedings. Invited Paper.

[17] M. Jazayeri, Species evolve, individuals age, in: M. Saeki, G. Canfora, S. Yamamoto (Eds.), 8th International Workshop on Principles of Software Evolution (IWPSE 2005), Lisbon, Portugal, 2005, pp. 3–9.

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

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

[20] P. Wegner, Why interaction is more powerful than algorithms, Commun. ACM 40 (5) (1997) 80–91.

[21] M. Mamei, F. Zambonelli, Programming pervasive and mobile computing applications: the tota approach, ACM Transactions on Software Engineering and Methodology 18 (4).

[22] M. Mamei, A. Roli, F. Zambonelli, Emergence and control of macro-spatial structures in perturbed cellular automata, and implications for pervasive computing systems, IEEE Transactions on Systems, Man, and Cybernetics, Part A 35 (3) (2005) 337–348.

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

[24] P. Ciancarini, Coordination models and languages as software integrators, ACM Computing Surveys 28 (2) (1996) 300–302.

[25] O. Babaoglu, G. Canright, A. Deutsch, G. A. D. Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, T. Urnes, Design patterns from biology for distributed computing, ACM Transactions on Autonomous and Adaptive Systems 1 (1) (2006) 26–66.

[26] L. Kari, G. Rozenberg, The many facets of natural computing, Communications of the ACM 51 (2008) 72–83.

[27] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. D. M. Serugendo, M. Risoldi, A.-E. Tchao, S. Dobson, G. Stevenson, J. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, B. Wally, Self-aware pervasive service ecosystems, Procedia CS 7 (2011) 197–199.

[28] G. A. Papadopoulos, F. Arbab, Coordination models and languages, in: M. V. Zelkowitz (Ed.), The Engineering of Large Systems, Vol. 46 of Advances in Computers, Academic Press, 1998, pp. 329–400.

[29] S. Ossowski, R. Menezes, On coordination and its significance to distributed and multi-agent systems, Concurrency and Computation: Practice and Experience 18 (4) (2006) 359–370, special Issue: Coordination Models and Systems.

[30] P. Ciancarini, A. Omicini, F. Zambonelli, Coordination technologies for Internet agents, Nordic Journal of Computing 6 (3) (1999) 215–240.

[31] D. Q. Goldin, S. A. Smolka, P. Wegner (Eds.), Interactive Computation: The New Paradigm, Springer, 2006.

[32] A. Omicini, A. Ricci, M. Viroli, The multidisciplinary patterns of interaction from sciences to Computer Science, in: Interactive Computation: The New Paradigm, Springer, 2006, pp. 395–414.

[33] P. Ciancarini, A. Omicini, F. Zambonelli, Multiagent system engineering: The coordination viewpoint, in: Intelligent Agents VI. Agent Theories, Architectures, and Languages, Vol. 1757 of LNAI, Springer, 2000, pp. 250–259, 6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 Jul. 1999. Proceedings.

[34] A. Omicini, M. Viroli, Coordination models and languages: From parallel computing to self-organisation, The Knowledge Engineering Review 26 (1) (2011) 53–59, special Issue 01 (25th Anniversary Issue).

[35] D. Rossi, G. Cabri, E. Denti, Tuple-based technologies for coordination, in: A. Omicini, F. Zambonelli, M. Klusch, R. Tolksdorf (Eds.), Coordination of Internet Agents: Models, Technologies, and Applications, Springer-Verlag, 2001, Ch. 4, pp. 83–109.

[36] P. Wyckoff, S. W. McLaughry, T. J. Lehman, D. A. Ford, T Spaces, IBM Systems Journal 37 (3) (1998) 454–474.

[37] E. Freeman, S. Hupfer, K. Arnold, JavaSpaces Principles, Patterns, and Practice: Principles, Patterns and Practices, The Jini Technology Series, Addison-Wesley Longman, 1999.

[38] A. Omicini, F. Zambonelli, Coordination for Internet application development, Autonomous Agents and Multi-Agent Systems 2 (3) (1999) 251–269, special Issue: Coordination Mechanisms for Web Agents.

[39] R. Menezes, R. Tolksdorf, Adaptiveness in Linda-based coordination models, in: G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, F. Zambonelli (Eds.), Engineering Self-Organising Systems. Nature-Inspired Approaches to Software Engineering, Vol. 2977, Springer, 2003, pp. 212–232.

[40] J. Liu, K. C. Tsui, Toward nature-inspired computing, Communications of the ACM 49 (10) (2006) 59–64.

[41] N. Shadbolt, From the Editor in Chief: Nature-Inspired Computing, IEEE Intelligent Systems 19 (1) (2004) 2–3.

[42] S. E. Page, Self organization and coordination, Computational Economics 18 (1) (2001) 25—48.

[43] 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 6 (1) (1959) 41–80.

[44] H. V. D. Parunak, "Go to the ant": Engineering principles from natural agent systems, Annals of Operation Research 75 (0) (1997) 69–101, special Issue on Artificial Intelligence and Management Science.

[45] M. Viroli, M. Casadei, A. Omicini, A framework for modelling and implementing self-organising coordination, in: S. Y. Shin, S. Ossowski, R. Menezes, M. Viroli (Eds.), 24th Annual ACM Symposium on Applied Computing (SAC 2009), Vol. III, ACM, Honolulu, Hawai'i, USA, 2009, pp. 1353–1360.

[46] G. Cabri, L. Leonardi, F. Zambonelli, Engineering mobile agent applications via context-dependent coordination, IEEE Trans. Software Eng. 28 (11) (2002) 1039–1055.

[47] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[48] G. Theraulaz, E. Bonabeau, A brief history of stigmergy, Artificial Life 5 (2) (1999) 97–116.

[49] E. Bonabeau, Editor's introduction: Stigmergy, Artificial Life 5 (2) (1999) 95–96.

[50] H. V. D. Parunak, S. Brueckner, J. Sauter, Digital pheromone mechanisms for coordination of unmanned vehicles, in: C. Castelfranchi, W. L. Johnson (Eds.), 1st International Joint Conference on Autonomous Agents and Multiagent systems, Vol. 1, ACM, New York, NY, USA, 2002, pp. 449–450.

[51] H. V. D. Parunak, A survey of environments and mechanisms for human-human stigmergy, in: D. Weyns, H. V. D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems II, Vol. 3830 of LNCS, Springer, 2006, pp. 163–186.

[52] M. Mamei, F. Zambonelli, Programming stigmergic coordination with the TOTA middleware, in: 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), ACM, New York, NY, USA, 2005, pp. 415–422.

[53] M. Mamei, F. Zambonelli, Pervasive pheromone-based interaction with RFID tags, TAAS 2 (2).

[54] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, 1999.

[55] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, A survey of autonomic communications, ACM Transactions on Autonomous and Adaptive Systems 1 (2) (2006) 223–259.

[56] P. Costa, L. Mottola, A. L. Murphy, G. P. Picco, Tuple space middleware for wireless networks, in: Middleware for Network Eccentric and Mobile Applications, Springer, 2009, pp. 245–264.

[57] A. Bosien, V. Turau, F. Zambonelli, Approaches to fast sequential inventory and path following in rfid-enriched environments, IJRFITA 4 (1) (2012) 28–48.

[58] J.-P. Banâtre, D. Le Métayer, The GAMMA model and its discipline of programming, Science of Computer Programming 15 (1) (1990) 55–77.

[59] J.-P. Banâtre, P. Fradet, D. Le Métayer, Gamma and the chemical reaction model: Fifteen years after, in: C. S. Calude, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View, Vol. 2235 of LNCS, Springer, 2001, pp. 17–44.

[60] G. Berry, The chemical abstract machine, Theoretical Computer Science 96 (1) (1992) 217–248.

[61] R. Frei, G. D. M. Serugendo, T.-F. Serbanuta, Ambient intelligence in self-organising assembly systems using the chemical reaction model, Journal of Ambient Intelligence and Humanized Computing 1 (3) (2010) 163–184.

[62] S. Mariani, A. Omicini, Self-organising news management: The *Molecules of Knowledge* approach, in: J. Pitt (Ed.), Self-Adaptive and Self-Organizing Systems Workshops (SASOW), IEEE CS, 2012, pp. 235–240, 2012 IEEE Sixth International Conference (SASOW 2012), Lyon, France, 10-14 Sep. 2012. Proceedings. doi:10.1109/SASOW.2012.48.

[63] T. Meyer, L. Yamamoto, C. F. Tschudin, An artificial chemistry for networking, in: First Workshop on Bio-Inspired Design of Networks, BIOWIRE 2007, Cambridge, UK, 2007, pp. 45–57.

[64] M. Monti, T. Meyer, C. F. Tschudin, M. Luise, Stability and sensitivity analysis of traffic-shaping algorithms inspired by chemical engineering, IEEE Journal on Selected Areas in Communications 31 (6) (2013) 1105–1114.

[65] J.-P. Banâtre, T. Priol, Chemical programming of future service-oriented architectures, JSW 4 (7) (2009) 738–746.

[66] M. S. Hossain, A. Alamri, A. El Saddik, A biologically inspired framework for multimedia service management in a ubiquitous environment, Concurrency and Computation: Practice and Experience 21 (11) (2009) 1450–1466.

[67] M. Mamei, F. Zambonelli, Field-Based Coordination for Pervasive Multiagent Systems. Models, Technologies, and Applications, Springer Series in Agent Technology, Springer, 2006.

[68] M. Mamei, F. Zambonelli, L. Leonardi, Co-fields: Towards a unifying approach to the engineering of swarm intelligent systems, in: P. Petta, R. Tolksdorf, F. Zambonelli (Eds.), Engineering Societies in the Agents World III, Vol. 2577 of LNCS, Springer, 2003, pp. 68–81, 3rd International Workshop (ESAW 2002), Madrid, Spain, 16–17 Sep. 2002. Revised Papers.

[69] H. Guo, Y. Jin, Y. Meng, A morphogenetic framework for self-organized multirobot pattern formation and boundary coverage, ACM Transactions on Autonomous and Adaptive Systems 7 (1) (2012) 15.

[70] M. Mamei, F. Zambonelli, Programming pervasive and mobile computing applications with the TOTA middleware, in: Pervasive Computing and Communications, 2004, pp. 263–273, 2nd IEEE Annual Conference (PerCom 2004), Orlando, FL, USA, 14–17 Mar. 2004. Proceedings.

[71] J. Beal, J. Bachrach, Infrastructure for engineered emergence on sensor/actuator networks, IEEE Intelligent Systems 21 (2) (2006) 10–19.

[72] M. Viroli, F. Damiani, J. Beal, A calculus of computational fields, in: Advances in Service-Oriented and Cloud Computing, Vol. 393 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2013, pp. 114–128.

[73] M. Viroli, M. Casadei, E. Nardini, A. Omicini, Towards a chemical-inspired infrastructure for self-* pervasive applications, in: D. Weyns, S. Malek, R. de Lemos, J. Andersson (Eds.), Self-Organizing Architectures, Vol. 6090 of LNCS, Springer, 2010, Ch. 8, pp. 152–176, 1st International Workshop on Self-Organizing Architectures (SOAR 2009), Cambridge, UK, 14-17 Sep. 2009, Revised Selected and Invited Papers.

[74] M. Viroli, M. Casadei, S. Montagna, F. Zambonelli, Spatial coordination of pervasive services through chemical-inspired tuple spaces, ACM Transactions on Autonomous and Adaptive Systems.

[75] J. Werfel, Y. Bar-Yam, D. Ingber, Bioinspired environmental coordination in spatial computing systems, in: 1st International SASO Workshop on Spatial Computing, IEEE Computer Society, Washington, DC, USA, 2008, pp. 338–343.

[76] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, P. Lió, K.-W. Lee, Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment, Ad Hoc Networks 7 (2009) 725–741.

[77] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, Jr., R. Nagpal, E. Rauch, G. J. Sussman, R. Weiss, Amorphous computing, Communications of the ACM 43 (2000) 74–82.

[78] F. Zambonelli, G. Castelli, M. Mamei, A. Rosi, Integrating pervasive middleware with social networks in sapere, in: Proceeding of the International Conference on Selected Topics in Mobile and Wireless Networking, Springer, 2011, pp. 145 –150.

[79] J.-P. Banâtre, T. Priol, Chemical programming of future service-oriented architectures, Journal of Software 4 (7) (2009) 738–746.

[80] G. Stevenson, M. Viroli, J. Ye, S. Montagna, S. Dobson, Self-organising semantic resource discovery for pervasive systems, in: 1st International Workshop on Adaptive Service Ecosystems: Natural and Socially Inspired Solutions, Lyon, France, 2012, pp. 47–52.

[81] M. N. Huhns, M. P. Singh, Service-oriented computing: Key concepts and principles, IEEE Internet Computing 9 (1) (2005) 75–81.

[82] S. Nath, P. B. Gibbons, S. Seshan, Z. R. Anderson, Synopsis diffusion for robust aggregation in sensor networks, in: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM, 2004, pp. 250–262.

[83] J. Ye, S. Dobson, S. McKeever, Situation identification techniques in pervasive computing: a review, Pervasive and Mobile Computing 8 (1) (2012) 36–66.

[84] J. Ye, G. Stevenson, S. Dobson, KCAR: a knowledge-driven approach for concurrent activity recognition, Pervasive and Mobile Computing. URL `doi://10.1016/j.pmcj.2014.02.003`

[85] M. Mamei, F. Zambonelli, Field-Based Coordination for Pervasive Multiagent Systems, Springer Series on Agent Technology, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[86] G. Di Marzo Serugendo, J. L. Fernandez-Marquez, Self-organising services, in: Proceedings of the 7th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2013, pp. 257–258.

[87] J. L. Fernandez-Marquez, G. Stevenson, A. E. Tchao, J. Ye, G. Di Marzo Serugendo, S. Dobson, Analysis of new gradient based aggregation algorithms for data-propagation in distributed networks, in: J. Pitt (Ed.), Self-Adaptive and Self-Organizing Systems Workshops (SASOW), IEEE CS, 2012, pp. 217–222, 2012 IEEE Sixth International Conference (SASOW 2012), Lyon, France, 10-14 Sep. 2012. Proceedings. doi:10.1109/SASOW.2012.45.

[88] G. Stevenson, J. Ye, S. Dobson, D. Pianini, S. Montagna, M. Viroli, Combining self-organisation, context-awareness and semantic reasoning: the case of resource discovery in opportunistic networks, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013, 2013, pp. 1369–1376.

[89] J. L. Fernandez-Marquez, G. Di Marzo Serugendo, S. Montagna, Biocore: Bio-inspired self-organising mechanisms core, in: Bio-Inspired Models of Networks, Information, and Computing Systems, Vol. 103, Springer Berlin Heidelberg, 2012, pp. 59–72.

[90] G. Stevenson, G. Castelli, J. Ye, A. Rosi, S. Dobson, F. Zambonelli, A bio-chemically inspired approach to awareness in pervasive systems, in: 1st International Workshop on Sensing and Big Data Mining (SENSEMINE'13), 2013, pp. 1–6. doi:10.1145/2536714.2536721.

[91] A. Molesini, A. Omicini, M. Viroli, F. Zambonelli, Engineering pervasive multiagent systems in SAPERE, in: M. Cossentino, A. El Fallah Seghrouchni, M. Winikoff (Eds.), Engineering Multi-Agent Systems, Vol. 8245 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, Ch. 11, pp. 196–214, 1st International Workshop (EMAS 2013), Saint Paul, Minnesota, USA, 6–7 May 2013, Revised Selected Papers.

[92] I. Ayala, M. Amor, L. Fuentes, M. Mamei, F. Zambonelli, Developing pervasive agent-based applications: A comparison of two coordination approaches, in: International Workshop on Agent-Oriented Software

Engineering, Vol. 7852 of Lecture Notes in Computer Science, 2013, pp. 73–98.

[93] B. H. C. Cheng, al., Software engineering for self-adaptive systems: A research roadmap, in: Software Engineering for Self-Adaptive Systems, 2009, pp. 1–26.

[94] M. Jelasity, A. Montresor, Ö. Babaoglu, Gossip-based aggregation in large dynamic networks, ACM Trans. Comput. Syst. 23 (3) (2005) 219–252.

[95] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41.

[96] J. L. Fernandez-Marquez, G. Di Marzo Serugendo, G. Stevenson, J. Ye, S. Dobson, F. Zambonelli, Self-managing and self-organising mobile computing applications: A separation of concerns approach, in: Proceeding of the 29th Symposium On Applied Computing (SAC 2014), 2014, pp. 458–465.

[97] J. Pitt, J. Schaumeier, A. Artikis, Axiomatization of socio-economic principles for self-organizing institutions: Concepts, experiments and challenges, TAAS 7 (4) (2012) 39.

[98] A. J. I. Jones, A. Artikis, J. Pitt, The design of intelligent socio-technical systems, Artif. Intell. Rev. 39 (1) (2013) 5–20.

[99] A. Molesini, M. Casadei, A. Omicini, M. Viroli, Simulation in Agent-Oriented Software Engineering: The SODA case study, Science of Computer Programming 78 (6) (2013) 705—714, special Section on Agent-oriented Design methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments.

[100] P. P. González Pérez, A. Omicini, M. Sbaraglia, A biochemically-inspired coordination-based model for simulating intracellular signalling pathways, Journal of Simulation 7 (3) (2013) 216–226, special Issue: Agent-based Modeling and Simulation.

[101] A. Omicini, Nature-inspired coordination models: Current status, future trends, ISRN Software Engineering 2013, article ID 384903, Review Article.