Engineering Environment-mediated Coordination via Nature-inspired Laws

Franco Zambonellii

Dipartimento di Scienze e Metodi dell'Ingegneria,
University of Modena and Reggio Emilia
e-mail: franco.zambonelli@unimore.it

Abstract SAPERE is a general multiagent framework to support the development of self-organizing pervasive computing services. One of the key aspects of the SAPERE approach is to have all interactions between agents take place in an indirect way, via a shared spatial environment. In such environment, a set of nature-inspired coordination laws have been defined to rule the coordination activities of the application agents and promote the provisioning of adaptive and self-organizing services.

1 Introduction

Progresses in mobile and ubiquitous computing are making possible to conceive a variety of innovative general-purpose pervasive computing services for interacting with the physical and social worlds around us [2]. However, the effective design and development of such services requires the capability of promoting flexible and adaptive interactions among a multiple of distributed devices and software components.

To support the vision, a great deal of research activity in pervasive computing has been devoted to meet the requirements of pervasive service systems, i.e.: supporting self-configuration and context-aware composition; enforcing self-adaptability and self-organization; and ensuring that service frameworks can be highly-flexible and long-lasting [8]. The SAPERE ("Self-aware Pervasive Service Ecosystems") approach [7] tackles the problem at the foundation, conceiving a radically new way of modeling integrated pervasive services and their execution environments, such that the apparently diverse issues of context-awareness, dependability, openness, flexibility, can all be uniformly addressed.

SAPERE models a pervasive service framework as a distributed multiagent system, in which the coordination between the application agents rely on spatially-situated and environment-mediated interactions [5]. In particular, in the SAPERE environment, a set of simple yet very expressive nature-inspired interaction laws dictates how agents will interact with each other, e.g., how they will compose and orchestrate their activities and how they will exchange information. Such an approach supports the provisioning of adaptive self-organizing services, suitable to meet the requirements of pervasive service systems.

In the remainder of this paper, we introduce the SAPERE environment-centered architecture and the key characteristics of its nature-inspired laws.

2 The SAPERE Approach and its Reference Architecture

SAPERE takes its primary inspiration from nature, and starts from the consideration that the dynamics and decentralization of future pervasive networks will make it suitable to model the overall world of pervasive services, data, and devices as a sort of distributed computational *ecosystem*.

As from Figure 1, SAPERE conceptually architects such pervasive service ecosystem as a non-layered *spatial environment*, laid above the actual pervasive network infrastructure. The environment embeds the basic *interaction laws* (which we also call *eco-laws*) that rule the activities of the system. The environment mediates all interactions and represents the ground on which components of different species indirectly interact and combine with each other. Such interactions take place in respect of the eco-laws and typically based on the spatial relationships between components, so as to serve their own individual needs as well as the sustainability of the overall ecology. Users can access the ecology in a decentralized way to use and consume data and services, and they can also act as "prosumers" by injecting new data or service components, possibly also for the sake of controlling the ecology behavior.

For the *components* living in the ecosystem, all of which can be abstracted as autonomous software agents (and whether being sensors, actuators, services, users, data, or resources in general), SAPERE adopts a common modeling and a common treatment. Each of them has an associated semantic representation which we call "LSA" (*Live Semantic Annotations*), to be injected in the spatial environment as it it were a sort of shared spatial memory (or tuple space [3]). This is a basic ingredient for enabling dynamic environment-mediated interactions between components. To account for the high dynamics of the scenario and for its need of continuous adaptation, SAPERE defines LSAs as living, active entities, tightly associated to the agent they describe, and capable of reflecting its current situation and context. This supports semantic and context-aware interactions both for service aggregation/composition and for data/knowledge management. In the case of pure data items, the entity and its LSA coincide.

The *eco-laws* define the basic interaction policies among the LSAs of the various agents of the ecology. In particular the idea is to enforce on a spatial basis, and possibly relying on diffusive mechanisms, dynamic networking and composition of data and services by composing their LSAs and exchanging data via them. Data and services (as represented by their associated LSAs) will be sort of chemical reagents, and interactions and compositions will occur via chemical reactions, relying on semantic pattern-matching between LSAs.

As detailed later on, the set of eco-laws includes: *Bond*, which is the basic mechanism for local interactions between components, and acts as a sort of virtual chemical bond between two LSAs (i.e., their associated agents); *Spread*, which diffuses LSAs on a spatial basis, and is necessary to support propagation of information and interactions among remote agents; *Aggregate*, which enforces a sort of catalysis among LSAs, to support distributed data aggregation; *Decay*, which mimics chemical evaporation and is necessary to garbage collect data.

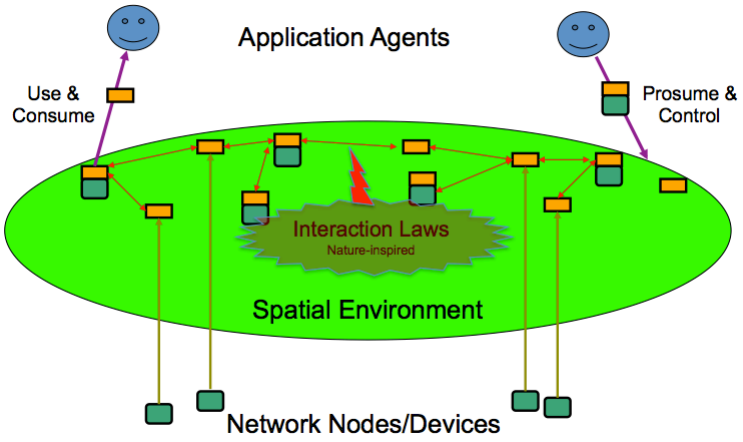

Figure 1. The SAPERE Architecture based on environment-mediated interactions

Adaptivity in SAPERE is not in the capability of individual components, but in the overall self-organizing dynamics of the ecosystem. In particular, adaptivity will be ensured by the fact that any change in the system (as well as any change in its components or in the context of the components, as reflected by dynamic changes in their LSAs) will reflect in the firing of new eco-laws, thus possibly leading to the establishment of new bonds or aggregations, and/or in the breaking of some existing bonds between components.

3 The SAPERE Middleware and its Programming Interface

In this section we shortly overview how SAPERE applications can be programmed, by introducing the API of the SAPERE middleware and exemplifying its usage.

3.1 The Middleware

The execution of SAPERE applications is supported by a middleware infrastructure [6] which reifies the SAPERE architecture in terms of a lightweight software support, enabling a SAPERE node to be installed in tablets and smartphones. Operationally, all SAPERE nodes (whether fixed at the infrastructure level or mobile) are considered at the same level since the middleware code they run could support the same services and provides the same set of functions.

Each SAPERE node hosts a local tuple space [3], that acts as a local repository of LSAs for local agents, and a local eco-laws engine. The LSA-space of each node is connected with a limited set of neighbor nodes based on spatial

proximity relations. Such relations consequently determine the spatial shape of the SAPERE substrate. From the viewpoint of individual agents (that will constitute the basic execution unit) the middleware provides an API to access the local LSA space, to advertise themselves (via the injection of an LSA), and to support the agents' need of continuously updating their LSAs. In addition, such API enables agents to detect local events (as the modifications of some LSAs) or the enactment of some eco-laws on available LSAs.

Eco-laws are realized as a set of rules embedded in SAPERE nodes. For each node, the same set of eco-laws applies to rule the dynamics between local LSAs (in the form of bonding, aggregation, and decay) and those between non-locally-situated LSAs (via the spreading eco-law that can propagate LSAs from a node to another to support distributed interactions).

From the viewpoint of the underlying network infrastructure, the middleware transparently absorbs dynamic changes at the arrival/dismissing of the supporting devices, without affecting the perception of the spatial environment by individuals.

3.2 The SAPERE API

In the SAPERE model, each agent executing on a node takes care of initializing at least one LSA (representing the agent itself), of injecting it on the local LSA space, and of keeping the values of such LSA (and of any additional LSA it decides to inject) updated to reflect its current situation. Each agent can modify only its own LSAs, and eventually read the LSAs to which it has been linked by a proper eco-law. Moreover LSAs can be manipulated by eco-laws, as explained in the following sections.

At the middleware level, a simple API is provided to let agents inject LSA – `injectLSA(LSA myLSA)` – and to let agents atomically update some fields of an LSA to keep it "alive" – `updateLSA(field = new-value)`. In addition, it is possible for an agent to sense and handle whatever events occur on the LSAs of an agent, e.g., some match that triggers some eco-laws. E.g., it is possible to handle the event represented by the LSA being bound with another LSA via the `onBond(LSA mylsa)` method.

The eco-laws assure self-adaptive and self-organizing activities in the ecosystems. Eco-laws operate on a pattern-matching schema: they are triggered by the presence of LSAs matching with each other, and manipulate such LSAs (and the fields within) according to a sort of artificial chemistry [8].

3.3 LSAs

LSAs are realized as descriptive tuples made by a number of fields in the form of "name-value" properties. By building over tuple-based models and extending upon them [3], the values in a LSA can be: *actual*, yet possibly dynamic and changing over time (which makes LSAs live); *formal* not tied to any actual value unless bound to one and representing a dangling connection (typically represented with a "`?`").

Pattern matching between LSAs – which is at the basis of the triggering of eco-laws – happens when all the properties of a description match, i.e, when for each property whose names correspond (i.e., are semantically equivalent) then the associated values match. As in classical tuple-based approaches, a formal value matches with any corresponding actual value.

For instance, the following `LSAa:(sensor-type = temperature; accuracy = 0.1; temp = 45)`, that can express the LSA of a temperature sensor, can match the following `LSAb:(sensor-type = temperature; temp = ?)`, which can express a request for acquiring the current temperature value. LSAa and LSAb match with each other. The properties present in LSAa (e.g., accuracy) are not taken into account by the matching function because it considers only inclusive match.

4 The Eco-laws Set

Let us now detail the SAPERE eco-laws and discuss their role in the SAPERE ecosystem.

4.1 Bond

Bonding is the primary form of interaction among co-located agents in SAPERE (i.e., within the same LSA space). In particular, bonding can be used to locally discover and access information, as well as to get in touch and access local services. All of which with a single and unique adaptive mechanism. Basically, the bond eco-law realizes a sort of a virtual link between LSAs, whenever two LSAs (or some SubDescriptions within) match.

The bond eco-law is triggered by the presence of formal values in at least one of the LSAs involved. Upon a successful pattern matching 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 bonding in the presence of the "`?`" formal fields is bi-directional and symmetric.

Once a bond is established, 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 can read each other 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. For instance, with reference to the `LSAa` and `LSAb` of the previous subsection, the agent having injected LSAb, upon bonding with `LSAa` (which the agent can detect with the `onBond` method) it can access the temperature measure by the sensor represented by `LSAb`.

As bonding is automatically triggered upon match, debonding takes place automatically whenever some changes in the actual "live" values of some LSAs make the matching conditions no longer holding.

We emphasize that bonding can be used to enable two agents to spontaneously get in touch with each other and exchange information, all of which

with a single operation and with both having injected an LSA in the space. That is, unlike in traditional discovery of data and services, without distinguishing between the roles of the involved agents and subsuming the traditionally separated phases of discovery and invocation.

4.2 Aggregate

The ability of aggregating 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 actual information will be available (some information source may disappear, other may appear, etc.) and the availability of ways to extract a summary of all available information (without having to explicitly discover and access the individual information sources) is very important.

The aggregate eco-law is intended to aggregate LSAs together so as to compute summaries of the current system's context. An agent can inject an LSA with the *aggregate* and *type* properties. The aggregate property identifies a function to base the aggregation upon. The type property identifies which LSAs to aggregate. In particular it identifies a numerical property of LSAs to be aggregated. In the current implementation, the aggregate eco-law can perform most common order and duplicate insensitive aggregation functions [1].

The aggregate eco-law supports separation of concern and allows to re-use 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 hand, the LSA resulting from an aggregation can be read (via a proper bond) by any other agent that needs to get the pre-computed result.

4.3 Decay

The decay eco-law enables the vanishing of components from the SAPERE environment. It applies to all LSAs that specify a decay property to update the remaining time to live according to the specific decay function, or actually removing LSAs that, based on their decay property, are expired.

The Decay eco-law therefore is a kind of garbage collector capable of removing LSAs that are no longer needed in the ecosystem or no longer maintained by an agent, for instance because they are the result of a propagation.

4.4 Spread

The above eco-laws act on a local basis, i.e., on a single LSA space. Since the SAPERE model is based on a network of interaction spaces, it is fundamental to enable non-local interactions, by providing a mechanism to send information to remote LSA spaces and making it possible to distribute information and results across a network of LSA spaces.

To this end, in SAPERE we defined the 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 vice versa to enable the remote advertisement of services. For an LSA to be subjected to the spread eco-law, it has to include a `diffusion` field, whose value (along with additional parameters) defines the specific type of propagation.

Two different types of propagation are implemented in the SAPERE framework: *(i)* a direct propagation used to spread an LSA to a specified neighbor node, so as to make it possible to realize gossiping schemes and multicasts; *(ii)* a general diffusion capable of propagating an LSA to all neighboring SAPERE nodes, possibly recursively applying such propagation up to a maximum distance form the source node.

General diffusion of an LSA via the spread eco-law to distances greater than one is a sort of broadcast 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 aggregate eco-law, so as to merge together such multiple replicas.

5 From Eco-laws to Distributed Self-organization

The above presented eco-laws form a necessary and complete set to support self-organizing environment-mediated interactions.

The eco-laws are necessary to support decentralized adaptive behaviors for pervasive service systems. Bonding is necessary to support adaptive local service interactions, subsuming the phases of discovery and invocation of traditional service systems. Spreading is necessary to diffuse information in a distributed environment and 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.

Further, and possibly of more software engineering relevance, the eco-law set is sufficient to express a wide variety of interaction schemes (or "patterns"), there included self-organizing ones. Bonding and spreading can be used to realize local and distributed client-server scheme of interactions as well as asynchronous models of interactions and information propagation. Coupling spreading with aggregation and decay, however, makes it possible to realize also those distributed data structures necessary to support all patterns of nature-inspired adaptive and self-organizing behaviors, i.e., virtual physical fields, digital pheromones, and virtual chemical gradients [1].

In particular, aggregation applied to the multiple copies of diffused LSAs can reduce the number of redundant LSAs so as to form a distributed *gradient* structures, also known as *computational force fields*. As detailed in [4], many different classes of self-organized motion coordination schemes, self-assembly, and distributed navigation can be expressed in terms of gradients.

In addition, spreading and aggregation can be used together to produce distributed self-organized aggregations, i.e., dynamically computing some distributed property of the system and have the results of such computation avail-

able at each and every node of the system. Distributed aggregation is a basic mechanism via which to realize forms of distributed consensus and distributed task allocation and behavior differentiation.

By bringing also the decay eco-law into play, and combining it with spreading and aggregation, one can realize pheromone-based data structures, which makes possible to realize a variety of bio-inspired schemes for distributed self-organization [1]. 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.

6 Conclusions

The innovative nature-inspired approach of SAPERE is effective to enforce, via environment-mediated interactions, a variety of self-organizing schemes for pervasive computing services. As the activities within the SAPERE European Project have finished, we will now challenge the SAPERE findings and tools against innovative services in the area of urban computing and smart cities [2].

Acknowledgments: Work supported by the EU project SAPERE, No. 256873.

References

1. Ozalp Babaoglu and al. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, 2006.
2. Nicola Bicocchi, Alket Cecaj, Damiano Fontana, Marco Mamei, Andrea Sassi, and Franco Zambonelli. Collective awareness for human-ict collaboration in smart cities. In *21st IEEE International WETICE Symposium*, pages 3–8, 2013.
3. David Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, January 1985.
4. Marco Mamei and Franco Zambonelli. Programming pervasive and mobile computing applications: the tota approach. *ACM Trans. Software Engineering and Methodology*, 18(4), 2009.
5. Danny Weyns, Alexander Helleboogh, Tom Holvoet, and Michael Schumacher. The agent environment in multi-agent systems: A middleware perspective. *Multiagent and Grid Systems*, 5(1):93–108, 2009.
6. Franco Zambonelli, Gabriella Castelli, Marco Mamei, and Alberto Rosi. Integrating pervasive middleware with social networks in sapere. In *International Conference on Selected Topics in Mobile and Wireless Networking*, pages 145–150, Shanghai, PRC, 2011.
7. Franco Zambonelli et al. Self-aware pervasive service ecosystems. *Procedia CS*, 7:197–199, 2011.
8. Franco Zambonelli and Mirko Viroli. A survey on nature-inspired metaphors for pervasive service ecosystems. *Journal of Pervasive Computing and Communications*, 7:186–204, 2011.