archive ouverte UNIGE
http://archive-ouverte.unige.ch

Proceedings Chapter

Self-organising Pervasive Ecosystems: A Crowd Evacuation Example

MONTAGNA, Sara, *et al*.

Abstract

The dynamics of pervasive ecosystems are typically highly unpredictable, and therefore self-organising approaches are often exploited to make their applications resilient to changes and failures. The SAPERE approach we illustrate in this paper aims at addressing this issue by taking inspiration from natural ecosystems, which are regulated by a limited set of "laws" evolving the population of individuals in a self-organising way. Analogously, in our approach, a set of so-called eco-laws coordinate the individuals of the pervasive computing system (humans, devices, signals), in a way that is shown to be expressive enough to model and implement interesting real-life scenarios. We exemplify the proposed framework discussing a crowd evacuation application, tuning and validating it by simulation.

Reference

MONTAGNA, Sara, *et al*. Self-organising Pervasive Ecosystems: A Crowd Evacuation Example. In: Troubitsyna, Elena. *Software Engineering for Resilient Systems*. Springer Berlin / Heidelberg, 2011. p. 115-129

DOI : 10.1007/978-3-642-24124-6_12

Available at:
http://archive-ouverte.unige.ch/unige:17066

Disclaimer: layout of this document may differ from the published version.

UNIVERSITÉ DE GENÈVE

Self-organising Pervasive Ecosystems: A Crowd Evacuation Example*

Sara Montagna[1], Mirko Viroli[1], Matteo Risoldi[2]
Danilo Pianini[1], and Giovanna Di Marzo Serugendo[2]

[1] Università di Bologna – Via Venezia 52, IT-47521 Cesena
{sara.montagna, mirko.viroli}@unibo.it, danilo.pianini@studio.unibo.it
[2] Université de Genève – Rte. de Drize 7, CH-1227 Carouge
{matteo.risoldi, giovanna.dimarzo}@unige.ch

Abstract. The dynamics of pervasive ecosystems are typically highly unpredictable, and therefore self-organising approaches are often exploited to make their applications resilient to changes and failures. The SAPERE approach we illustrate in this paper aims at addressing this issue by taking inspiration from natural ecosystems, which are regulated by a limited set of "laws" evolving the population of individuals in a self-organising way. Analogously, in our approach, a set of so-called *eco-laws* coordinate the individuals of the pervasive computing system (humans, devices, signals), in a way that is shown to be expressive enough to model and implement interesting real-life scenarios. We exemplify the proposed framework discussing a crowd evacuation application, tuning and validating it by simulation.

Keywords: pervasive computing, software ecosystems, self-adaptation, self-organisation

1 Introduction

The increasing evolution of pervasive computing is promoting the emergence of decentralised infrastructures for pervasive services. These include traditional services with dynamic and autonomous context adaptation (e.g., public displays showing information tailored to bystanders), as well as innovative services for better interacting with the physical world (e.g., people coordinating through their PDAs). Such scenarios feature a number of diverse sensing devices, personal and public displays, personal mobile devices, and humans, all of which are dynamically engaged in flexible coordinated activities and have to account for resilience when conditions change. Recent proposals in the area of coordination models and middlewares for pervasive computing scenarios try to account for issues related to spatiality [11, 13], spontaneous and opportunistic coordination [1, 7], self-adaptation and self-management [18]; however, most works propose

* This work has been supported by the EU-FP7-FET Proactive project SAPERE—Self-aware Pervasive Service Ecosystems, under contract no.256873

ad-hoc solutions to specific problems in specific areas, and lack generality. The SAPERE project ("Self-adaptive Pervasive Service Ecosystems") addresses the above issues in a uniform way by means of a truly self-adaptive pervasive substrate; this is a space bringing to life an ecosystem of individuals, namely, of pervasive services, devices, and humans. These are coordinated in a self-organising way by basic laws (called *eco-laws*), which evolve the population of individuals in the system, thus modelling diverse mechanisms of coordination, communication, and interaction. Technically, such eco-laws are structured as sort of chemical reactions, working on the "interface annotation" of components residing in neighbouring localities—called *LSA* (Live Semantic Annotation).

A notable application of the proposed approach is in resilient crowd steering applications, in which a crowd is guided in a pervasive computing scenario depending on unforeseen events, such as the occurrence of critical events (i.e. alarms) and the dynamic formation of jams. We exemplify the approach in a crowd evacuation scenario, providing its set of eco-laws and validating it via simulation of the associated Continuos-Time Markov Chain (CTMC) model.

The remainder of the paper is organised as follows. In Section 2 we give a brief overview of the SAPERE approach and general architecture. Section 3 examines in detail the language for eco-laws. Section 4 describes the concrete example of crowd evacuation application, which is then validated in Section 5 by simulation. Related work and conclusions wrap up the article.

2 Architecture

The SAPERE approach is inspired by the mechanism of chemical reactions [22]. The basic idea of the framework is to model all the components in the ecosystem in a uniform way, whether they are humans perceiving/acting over the system directly or through their PDAs, pervasive devices (e.g., displays or sensors), or software services. They are all seen as external components (i.e., agents), reifying their relevant interface/behavioural/configuration information in terms of an associated semantic representation called *Live Semantic Annotation* (LSA). To account for dynamic scenarios and for continuous holistic adaptation and resilience, we make LSAs capable of reflecting the current situation and context of the component they describe. As soon as a component enters the ecosystem, its LSA is automatically created and injected in the SAPERE substrate, which is a shared space where all LSAs live and interact. Topologically, this shared space is structured as a network of LSA-spaces spread in the pervasive computing system and containing the LSAs of the associated components, each hosted by a *node* of the SAPERE infrastructure. Proximity of two LSA-spaces implies direct communication abilities.

Each LSA-space embeds the basic laws of the ecosystem, called *eco-laws*, which rule the activities of the system by evolving the population of LSAs. They define the policies to rule *reactions* among LSAs, enforcing coordination of data and services. LSAs (and thus their associated data and services) are like chemical reagents in an ecology in which interactions and composition occur via

chemical-like reactions featuring pattern-matching between LSAs. Such reactions *(i)* change the status of entities depending on the context (e.g., a display showing information on the nearest exit only when a bystander needs it), *(ii)* produce new components (e.g., a composite service coordinating the execution of atomic service components), or *(iii)* diffuse LSAs to nearby nodes (e.g., propagating an alarm, or creating a gradient data structure).

Coordination, adaptivity, and resilience in the SAPERE framework are not bound by the capability of individual components, but rather emerge in the overall dynamics of the ecosystem. Changes in the system (and changes in its components, as reflected by dynamic changes in their LSAs) result in the firing of eco-laws, possibly leading to the creation/removal/modification of other LSAs. Thus, the SAPERE architecture promotes adaptivity and coordination not by enacting resilience at the level of components, but rather promoting a sort of "systemic resilience".

3 Eco-law Language

For the sake of readability, we here present the eco-law language informally[3]. Although in the SAPERE framework LSAs are *semantic* annotations, expressing information with same expressiveness of standard frameworks like RDF, we here consider a simplified notation without affecting the expressiveness of the self-organisation patterns we describe. Namely, an LSA is simply modelled as a tuple $\langle v_1, \ldots, v_n \rangle$ (ordered sequence) of typed values, which could be for example numbers, strings or structured types. For instance, $\langle \texttt{10001}, \texttt{sensor}, \texttt{temperature}, 28 \rangle$ could represent the LSA with identifier 10001, injected by a sensor which currently measures temperature 28. In writing LSAs and eco-laws, we shall use `typetext` font for concrete values, and *italics* for variables.

An eco-law is a chemical-resembling reaction working over patterns of LSAs. An LSA pattern P is basically an LSA which may have some variable in place of one or more arguments of a tuple, and as usual an LSA L is said to match the pattern P if there exists a substitution of variables which applied to P gives L. An eco-law is hence of the kind $P_1, \ldots, P_n \overset{r}{\mapsto} P'_1, \ldots, P'_m$, where: *(i)* the left-hand side (reagents) specifies patterns that should match LSAs $L_1, \ldots, L_n$ to be extracted from the LSA-space; *(ii)* the right-hand side (products) specifies patterns of LSAs which are accordingly to be inserted back in the LSA-space (after applying substitutions found when extracting reagents, as in standard logic-based rule approaches); and *(iii)* rate r is a numerical positive value indicating the average frequency at which the eco-law is to be fired—namely, we model execution of the eco-law as a CTMC transition with Markovian rate (average frequency) r. As a simple example, the eco-law $\langle \texttt{10001}, \texttt{a}, \texttt{10} \rangle \overset{1.0}{\longmapsto} \langle \texttt{10001}, \texttt{a}, \texttt{11} \rangle$ fires when $\langle \texttt{10001}, \texttt{a}, \texttt{10} \rangle$ is found in a space, its effect is to increment value 10

[3] A formal description, associating an operational semantics to the eco-law language based on Continuos-Time Markov Chains, is presented in the companion paper available at `http://www.ingce.unibo.it/~mviroli/SERENE-full.pdf`.

to 11, and the rate of its application is 1.0—an average of once per time unit. A more general eco-law $\langle id, \mathtt{a}, 10\rangle \overset{1.0}{\longmapsto} \langle id, \mathtt{a}, 11\rangle$ would work on LSAs with any identifier.

This simple language is extended with some key ingredients to fit the goals of our framework. First of all, to allow interaction between different LSA-spaces, we introduce the concept of *remote pattern*, written $+P$, which is a pattern that will be matched with an LSA occurring in a neighbouring LSA-space (called a *remote LSA*): for example, $\langle id, \mathtt{a}, \mathtt{10}\rangle \overset{1.0}{\longmapsto} +\langle id, \mathtt{a}, \mathtt{11}\rangle$ removes the reagent LSA in a space (called the *local space*), and injects the product LSA into a neighbouring space, called the *remote space* (chosen probabilistically among matching neighbours). Note that when more remote patterns occur into an eco-law, they are all assumed to work on the same remote space, e.g., $P_1, +P_2, +P_3 \overset{r}{\mapsto} P_4, +P_5$ works on a local space where P_1 occurs and a remote space where P_2, P_3 occur, and its effect is to replace P_1 with P_4 locally, and P_2, P_3 with P_5 remotely.

In order to allow eco-laws to apply to a wide range of LSAs, the argument of a pattern can also be a mathematical expression—including infix/prefix operators, e.g., $+,-,*,/,min$. For instance, $\langle id, \mathtt{a}, x\rangle \overset{1.0}{\longmapsto} \langle id, \mathtt{a}, x + \mathtt{1}\rangle$ makes third argument of a matching LSA be increased by 1. Finally, among variables, some system variables can be used both in reagents and products to refer to contextual information provided by the infrastructure; they are prefixed with # and include $\#T$ which is bound to the time at which the eco-law fires, $\#D$ which is the topological distance between the local space and the remote space, and $\#O$ which is the orientation of the remote space with respect to the local space—e.g., an angle, a `north`/`south`/`west`/`east` indication, or any useful term like `in-front-of`, `on-rigth`, `in-the-same-room`.

4 A crowd evacuation application

A public exposition is taking place in a museum composed of rooms, corridors, and exit doors. The surface of the exposition is covered by sensors, arranged in a grid, able to sense fire, detect the presence of people, interact with other sensors in their proximity as well as with PDAs that visitors carry with them.

When a fire breaks out, PDAs (by interaction with sensors) must show the direction towards an exit, along a safe path. The system has to be resilient to changes or unpredicted situations, in particular the safe path should: *(i - distance)*: lead to the nearest exit; *(ii - fire)*: stay away from fire; and *(iii - crowd)*: avoid overcrowded paths. These factors influence PDAs by means of the following LSAs:

- The exit gradient: each node contains an LSA with a numeric value indicating the distance from the nearest exit. These LSAs form a gradient field covering the whole expo [11]. Over an exit, the gradient value is 0, and it increments with the distance from the exit. When there are several exits or several paths towards an exit, the lowest distance value is kept for a node.

- The fire gradient: each node also contains an LSA indicating its distance from nearest fire. The LSA's value is 0 at the fire location and increases with the distance from it. It reaches a maximum value at a safe distance from a fire.
- The crowding gradient: sensors in the expo surface detect the presence of people and adjust the value in a local LSA indicating the crowding level of the location. As with exit and fire, this LSA is diffused around, and its value increases with the distance from the crowded region.
- The attractiveness information: finally, each node contains an LSA indicating how desirable it is as a position in an escape route. Its value is based on the values of the previous three, and is used to choose which direction displays should point to.

4.1 Types of LSAs in the system

There are three forms of LSAs used in this scenario:

$$\langle \texttt{source}, type, max, ann \rangle,\ \langle \texttt{grad}, type, value, max, ann \rangle,\ \langle \texttt{info}, type, value, tstamp \rangle$$

A **source** LSA is used for gradient sources: *type* indicates the type of gradient (`fire`, `exit`, and so on); *max* is the maximum value the gradient can assume; and *ann* is the annealing factor [4]—its purpose will be described later, along with eco-laws. A **gradient** LSA is used for individual values in a gradient: *value* indicates the individual value; and the other parameters are like in the source LSAs. Finally, an **info** LSA is used for local values (e.g., not part of a gradient)—parameters are like in the source and gradient LSAs. The *tstamp* reflects the time of creation of the LSA.

4.2 Building the fire, exit and crowding gradients

The sources of the gradients are injected by sensors at appropriate locations, with the values $\langle \texttt{source}, \texttt{exit}, \texttt{Me}, \texttt{Ae} \rangle$ and $\langle \texttt{source}, \texttt{fire}, \texttt{Mf}, \texttt{Af} \rangle$. For the crowding information, we may assume that sensors are calibrated so as to locally inject an LSA indicating the level of crowding. The actual threshold in number of people will mainly depend on the sensor arrangement, and should be seen as a configuration issue. The crowding LSA looks like $\langle \texttt{source}, \texttt{crowd}, \texttt{Mc}, \texttt{Ac} \rangle$ and is periodically updated by the sensor.

As sources are established, gradients are built by the following set of eco-laws, applying to exit, fire and crowding gradients by parameterising argument *type* in the LSAs. First, we define Eco-law 1 that, given a source, initiates its gradient:

$$\langle \texttt{source}, T, M, A \rangle \xmapsto{R_{init}} \langle \texttt{source}, T, M, A \rangle, \langle \texttt{grad}, T, \texttt{0}, M, A \rangle \tag{1}$$

When a node contains a gradient LSA, it spreads it to a neighbouring node with an increased value, according to Eco-law 2:

$$\langle \texttt{grad}, T, V, M, A\rangle \overset{R_s}{\longmapsto} \langle \texttt{grad}, T, V, M, A\rangle, +\langle \texttt{grad}, T, \texttt{min}(V{+}\#D, M), M, A\rangle \quad (2)$$

Due to use of system variable $\#D$, each node will carry a `grad` LSA indicating the topological distance from the source. When the spread values reach the maximum value M, the gradient becomes a plateau. Also note that the iterative application of this eco-law causes continuous diffusion of the LSA to all neighbouring nodes.

The spreading eco-law above may produce duplicate values in locations (due to multiple sources, multiple paths to a source, or even diffusion of multiple LSAs over time). Thus, Eco-law 3 retains only the minimum distance:

$$\langle \texttt{grad}, T, V, M, A\rangle, \langle \texttt{grad}, T, W, M, A\rangle \rightarrow \langle \texttt{grad}, T, \texttt{min}(V, W), M, A\rangle \quad (3)$$

Finally, we have to address the dynamism of the scenario where people move, fires extinguish, exits may be blocked, crowds form and dissolve. For instance, if a gradient source vanishes, the diffused values should increase (the distance to exit increases if the nearest exit is no longer available). However, with the above eco-laws this does not happen because Eco-law 3 always retain the minimum value. This is the purpose of the annealing parameter in the gradient LSAs: it defines the rate of Eco-law 4, which continuously tends to level up gradient values, encouraging the replacement of old values by more recent ones:

$$\langle \texttt{grad}, T, V, M, A\rangle \overset{R_{ann}(A)}{\longmapsto} \langle \texttt{grad}, T, V{+}1, M, A\rangle \quad (4)$$

The R_{ann} rate is directly proportional to A. When a fire is put out, for example, this eco-law will gradually raise the fire gradient to the point where it reaches the maximum, indicating no fire. Annealing may introduce a burden on the system, therefore high annealing values should only be used for gradients that have to change often or quickly.

4.3 Ranking escape paths: the attractiveness field

Based on exit distance, fire distance and crowding, a location can be ranked as more or less "attractive" to be part of an escape path. This is done via an *attractiveness* value automatically attached to each node by eco-law 5:

$$\begin{array}{l}\langle \texttt{grad}, \texttt{exit}, E, Me, Ae\rangle, \langle \texttt{grad}, \texttt{fire}, F, Mf, Af\rangle, \langle \texttt{info}, \texttt{crowd}, CR, TS\rangle \overset{R_{att}}{\longmapsto} \\ \langle \texttt{grad}, \texttt{exit}, E, Me, Ae\rangle, \langle \texttt{grad}, \texttt{fire}, F, Mf, Af\rangle, \langle \texttt{info}, \texttt{crowd}, CR, TS\rangle, \\ \langle \texttt{info}, \texttt{attr}, (Me - E)/(1 + (Mf - F) + k \times (Mc - C)), \#T\rangle\end{array} \quad (5)$$

Coefficient k (tuned by simulation) is used to weight the effect of crowding on attractiveness. As gradients evolve, older attractiveness LSAs are replaced with newer ones (T is assumed positive):

$$\langle \texttt{info}, \texttt{attr}, A, TS\rangle, \langle \texttt{info}, \texttt{attr}, A2, TS{+}T\rangle \rightarrow \langle \texttt{info}, \texttt{attr}, A2, TS{+}T\rangle \quad (6)$$

4.4 Choosing a direction

Each location contains by default an LSA of the form $\langle \texttt{info}, \texttt{escape}, L, TS \rangle$, where L is the direction to be suggested by the PDA. In principle, the neighbour with the highest attractiveness should be chosen, but a more resilient solution is to tie the markovian rate of eco-laws to the attractiveness of neighbours, so that the highest probability is to point the best neighbour, with a possibility to point a less-than-optimal (but still attractive) neighbour:

$$\begin{array}{l} \langle \texttt{info}, \texttt{escape}, L \rangle, \langle \texttt{info}, \texttt{attr}, A, TS \rangle, +\langle \texttt{info}, \texttt{attr}, A{+}\Delta, TS2 \rangle \xmapsto{R_{disp}(\Delta)} \\ \langle \texttt{info}, \texttt{escape}, \#O \rangle, \langle \texttt{info}, \texttt{attr}, A, TS \rangle, +\langle \texttt{info}, \texttt{attr}, A + \Delta, TS2 \rangle \end{array} \tag{7}$$

The rate is proportional to the difference in attractiveness between the node and its neighbour (Δ). The higher is Δ, the higher is the rate. Note that Δ is a positive value, hence $A{+}\Delta$ implies that eco-law 7 only considers neighbours with a higher attractiveness, i.e., the PDA will not point *away* from the exit. Finally, once a direction is chosen, people move of discrete steps inside the environment. It is also defined a minimum possible distance between them, so to model the physical limit and the fact that two visitors can't be in the same place at the same time.

4.5 Resilient behaviour

We adopt the definition of "resilient system" as one that is *"tolerant to the harmful faults, which are most likely to affect the overall system service. [...] Fault tolerance is an acute issue in designing resilient pervasive systems"* [9]. The proposed architecture allows to express models that adapt dynamically to unexpected events (e.g., here we may have node failures, network isolation, exits suddenly unavailable, crowd formation, and so on) while maintaining their functionality. The nature of the eco-law language, akin to a rewriting system, is particularly suitable to express models by composing simple patterns that are known to exhibit resilient behaviour, thus not having to explicitly express aspects like error recovery or exceptional behaviour. One such pattern, for example, is the one building gradients. For a discussion more focused on patterns, see [19]. We will now discuss a few examples of possible problems and show how our example system reacts to them.

Single-node failure Most behaviour in this architecture is based on nodes being able to apply eco-laws and host LSAs. Failure of a node clearly impacts this behaviour in some measure. A failing node (i.e., disappearing from the system) results in a physical location where no information for the displays is available, and where gradient information is not received or transmitted.

If the failing node is a generic one (i.e, no fire or exits at that node) and its disappearance does not cause network isolation, the impact is somewhat limited. Gradients will still be transmitted *around* the broken node, although values will change. Displays traversing a failing node will not update their direction; however, they will also never guide people towards a failing node, because it does

not have an attractiveness value; they will rather steer people around the failing node. This means that functionality is preserved with decreased efficiency (i.e., longer paths).

If the failing node is a node containing a fire, the consequences are more severe: the fire will not be detected. PDAs will still never point the failing node on fire directly, but they will let people pass near it. One might argue that people will still see the fire; however, from the point of view of system behaviour, safety is reduced. On the other hand, it can be assumed that a serious fire would be detected by more sensors, and that all of them failing at the same time is unlikely.

If the failing node is an exit node, functionality gets a major hit, because the whole system will lack a fundamental information for display functioning (the exit gradient). If other exits are available, the system will still guide people towards them (again preserving functionality with reduced efficiency). However, if the failing node "hides" the only available exit, the system completely fails to provide useful information. This situation can be tackled by redundant source nodes near each exit.

Network isolation When a group of failing nodes is the only connection between two parts of the network, this causes network isolation. There are a few sub-scenarios to consider. If the isolated section does not include exits, we find another limitation: the evacuation mechanism cannot work inside that section, as it is lacking fundamental information. If the section does however contain exits, it will act as an autonomous system in itself. Efficiency may be reduced but functionality will still be preserved, with an important caveat: this section of the network will ignore fires and alarms occurring in the rest of the network. This may or may not be an acceptable limitation depending on the scenario. For fires, it could probably be unacceptable. In a general way, however, we would consider that an isolated part of the network behaving as an autonomous system is an acceptable fallback. Again, redundancy is desirable for those nodes of the network that can cause isolation.

Exit unavailability If an exit is suddenly unavailable in the expo, for instance it is broken and hence it does not open, a sensor could detect this and drop the source LSA. Because of the annealing mechanism, this causes the gradient value in each node to converge to the situation corresponding to the unavailability of that door; namely, a new path toward another exit emerges and affects the behaviour of all PDAs—people would simply go back to another exit as expected.

Crowd formation While people follow their PDAs towards the nearest exit, it is possible that a jam forms in front of certain doors or corridors: this is one of the most perilous, unexpectable situations to experience in crowd evacuation. The application we set up is meant to emergently tackle these situations by means of the crowding gradient.

Assume two available corridors exist towards some exit, and from a given room people can choose to walk through any of them. If one becomes unavailable because a jam is forming the crowding gradient would emerge, reducing attractiveness along that corridor. Accordingly, it is expected that people behind the crowded area start walking back and head for the other corridor, and people in the room start choosing systematically the path free from crowd, even if they do not "see" the problem from the room.

5 Simulation

The proposed system is specified by a limited set of eco-laws, equipped with CTMC semantics. This specification can be used to feed the SAPERE infrastructure and be simulated in order to check the model dynamics in advance.

In this section we describe the prototype simulator we used, along with some simulations to validate the general correctness of the proposed system and to tune some parameter of the model before implementation and deployment.

5.1 The simulator

In order to simulate the scenario described in Section 4, the main requirement for a simulation engine is to support a computational model built around a set of interacting and mobile nodes, which autonomously perform internal actions to aggregate/transform the information they hold. Such actions have the form of eco-laws that change system state following the CTMC model. Moreover, the number and position of nodes can change over time to model for instance new PDAs entering the system and moving around, or nodes breaking down.

Although many works manage to capture complex scenarios like the above one, we found existing approaches not completely fitting our framework. For instance, the agent-based model (ABM) [10] is a computational approach which provides the useful abstractions for modelling some of the mentioned concepts, *i.e.*, each node can be modelled as an agent that owns an internal behaviour and interacting capabilities. An ABM is simulated on top of platforms such as Repast or MASON [16] which provide the user with model specification and execution services. Unfortunately they do not support the CTMC model, which is a key element of the eco-laws model.

This is instead typically supported by simulators of chemical systems. However, the state-of-the-art in this context does not focus on highly mobile networks of chemical compartments [12]. A good deal of work has actually been moving towards multi-compartimentalisation: from the single, global solution idea of e.g. stochastic π-calculus [15], to mechanisms and constructs tackling the multi-compartment scenario of *Membrane computing* [14], the $S\pi@$ process calculus [20] and Bio-PEPA [5]. The mentioned languages and frameworks are not however conceived to address systems composed by a huge number of interacting compartments, and do not support dynamic networks, for the topology of the system is static.

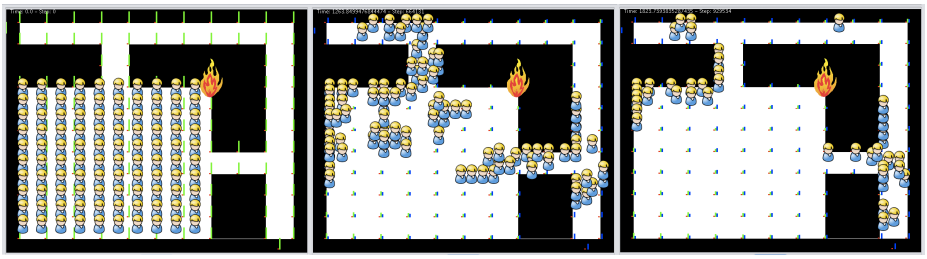

Fig. 1. A simulation run of the reference exposition: three snapshots

For the above reasons, in the context of the SAPERE project we are developing an ad-hoc simulator with the purpose of executing chemical-like, multi-compartment scenarios resembling pervasive ecosystems, in which the position and number of nodes can dynamically vary with time as a results of adding, removing and moving nodes in the system. For the sake of brevity, we only sketch here its basic features.

The behaviour of each node is programmed according to the eco-laws coordination model explained in Section 3. Moreover the concept of *reaction* in the classical form of $A+B \xrightarrow{r} C+D$ is extended, still maintaining the CTMC semantics, into the form $c_1, \ldots, c_i \xmapsto{f(k,c_1,\ldots,c_i,a_1,\ldots,a_j)} a_1, \ldots, a_j$ where reactants are a series of conditions over LSAs in local and remote LSA-spaces, and products are a series of changes to such LSA-spaces. The markovian rate is computed according to a kinetic function f. This model allows us to represent simple chemical reactions up to complex state transitions such as those modelled by eco-laws.

The simulator engine is written in Java[4] and features: *(i)* an XML-based, low-level specification language to structure reactions, which can be used as sort of byte-code that higher-level specifications can be compiled to; *(ii)* an internal engine based on the "Next Reaction Method" (an optimised version of the well-known Gillespie's algorithm [8]); *(iii)* a set of configurable interfaces to show simulation results visually.

5.2 Simulation setting

We here present simulations conducted over an exposition structured as shown in Figure 1, where three snapshots of a simulation run are reported: all the people in the room start moving towards one of the two exists (located at the ends of the corridor) because of the fire in the top-right corner of the room. Note in third snapshot that a person is walking in the middle of the corridor, for she was suggested to go to a farther exit because of the corridor jam at the bottom-right.

Rooms and corridors are covered by a grid of locations hosting sensors, one per meter in the room, one per two meters in the corridor: such locations are

[4] The choice of this language, compared to others more standard for simulators like C++, revealed no significant penalisation in performance.

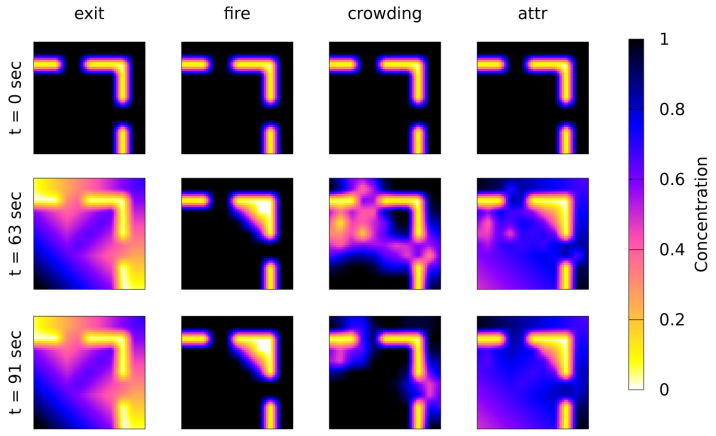

Fig. 2. Gradients for the snaphots in Figure 1. Concentration is the gradient value normalised to its maximum value.

the infrastructure nodes where LSAs are reified. The maximum values for the gradients are set to: `Me` = 30, `Mf` = 3, `Mc` = 20. The PDA of each person is modelled as a mobile node, able to perceive the attractiveness gradient in the nearest sensor location: accordingly, the person moves in the suggested direction.

Figure 2 shows the gradients of exit, fire, crowding and attractiveness (one per column) corresponding to the simulation steps of Figure 1 (one step per row). At $t = 0$, gradients are level; with time, the gradient self-modify—it is easy to see the exits, fire and crowds in the respective gradients. The crowding gradient in the third column changes dynamically during simulation according to the movement of people. The last column shows the attractiveness gradient, computed from the other three gradients. Note how the second snapshot shows an attractiveness "hole" in the middle of the room and in the corridor due to crowding.

5.3 Tuning parameters

We chose to tune by simulation two parameters of the model: k (used in the eco-law computing attractiveness), and a multiplication factor p we applied to the crowding gradient to control its slope—namely, how far crowding should be perceived. Different simulations have been performed in order to find the set of parameters that minimises the time of exit of all the people in the room: the range of variation of k is [0, 1] with a step of 0.1, while the range for p is [0, 20] with step of 0.5. The smaller is k, the lower is the impact of the crowding in the computation of attractiveness, so that we expect the crowds not be avoided and the exit time to grow. The bigger is p, the higher is the crowding gradient slope, until it becomes a local information that PDAs can perceive only very close to it.

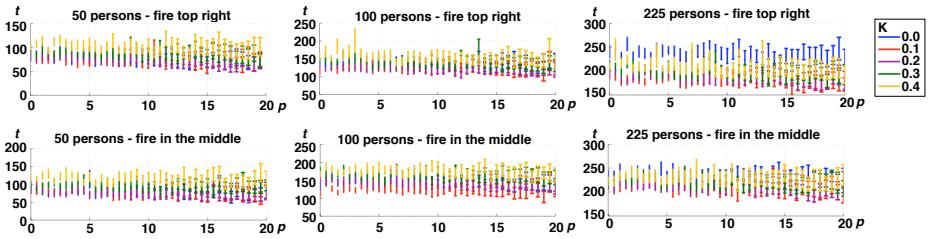

Fig. 3. Results of parameter analysis (t: time of exit [sec])

Experiments have been done with 50, 100 and 225 persons in the room and with a fire in the corner of the room or in the middle, so as to experimentally observe the impact of parameters in the evacuation outcome of different scenarios. For each couple of parameters, 10 simulations have been run, considering average value and standard deviation of the resulting time.

In Figure 3 we show the results we obtained from the analysis of the parameters. Only dynamics for $k < 0.5$ are shown, because the time of exit is much higher elsewhere, and also because standard deviation becomes much higher since the system is much more chaotic.

The graphs show that with $k = 0$ (i.e., if the crowding gradient does *not* influence attractiveness) the performance of the system slows down as the number of persons increases, and the likelihood of crowd formation grows. For 50 and 100 persons not considering the crowding gradient does not have a big impact in the system dynamic as if very minimal congestion is formed, whereas it significantly increases the exit time with 225 persons. This result highlights the impact of considering crowds in resilience with respect to jams. For $k = 0.1$ we obtained the best performance for the system for both simulated fire positions. Results finally showed that the parameter p does not seem to have relevant impact on the average exit time in this particular expo.

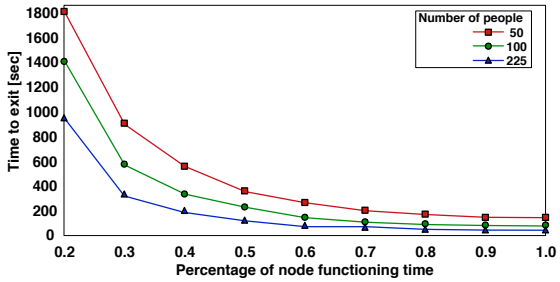

Fig. 4. Resilience to temporary node failures

5.4 Resilience to node failures

Among the many simulations analysing resilience aspects, we focused on the case in which all nodes experience temporary failures, i.e., they work correctly only for a percentage of time. We expect the system to still be safe, i.e., be able to drive people out of the room in reasonable time, in spite of the changes in gradient values around the broken nodes. The results of simulation with 50, 100 and 225 persons are shown in Figure 4: the time to exit does not significantly increase even in the case of nodes being failing for 30%-40% of the time, and people are still able to eventually reach the exit even with 80% of node downtime.

6 Related Works

Chemical-oriented coordination The issue we face in this article can be framed as the problem of finding the proper coordination model for enabling and ruling interactions of pervasive services. With LSAs and LSA-spaces we took as basis the archetypal LINDA model, which simply provides for a blackboard or space with associative matching for mediating component interactions through insertion/retrieval of tuples. Then, we followed the idea of engineering the coordination space of a distributed system by some policy "inside" tuple spaces. Our proposal extends these models to include bio-inspired ecological mechanisms, via fine-grained and well structured chemical-like reactions, the eco-laws. This idea originated from the chemical tuple space model in [21], though with some notable differences: *(i)* here we provide a detail notational framework to flexibly express eco-laws that work on patterns of LSAs and affect their properties; *(ii)* the chemical concentration mechanisms proposed in [21] to exactly mimic chemistry is not mandatory here—though it can be achieved by a suitable design of rate expressions; *(iii)* the local-remote space mechanism described here is new; *(iv)* the way we conceive the overall infrastructure and its applications goes beyond the mere definition of the tuple space model in [21].

Beside tuple spaces, chemistry has been a source of inspiration for several works in (distributed) computing and coordination like in the Gamma language and its extensions such as HOCL [2] and CHAM [3]. The main features we inherit from this research thread include: *(i)* conferring a high-level, abstract, and nature-inspired character to the language used to program the distributed system behaviour; *(ii)* providing a reactive computational model very useful in autonomic contexts. While Gamma and it extensions (such as HOCL) were exploited in different application contexts [2], they originated with the goal of writing concurrent, general-purpose programming languages. Our approach instead aims at specifically tackling coordination infrastructures for pervasive systems, which calls for dictating specific mechanisms of diffusion, context- and spatial-awareness.

Situatedness and Context-Awareness Considering the issues of situatedness and context-awareness, extensions or modifications to traditional ap-

proaches have been recently proposed to address adaptivity in pervasive environments. Similarly to our approach, in PLASTIC [1] service descriptions are coupled with dynamic annotations related to the current context and state of a service, to be used for enforcing adaptable forms of service discovery. However, our approach gets rid of traditional discovery services and enforces dynamic and adaptive service interaction via simple chemical reactions and a minimal middleware. In many proposals for pervasive computing environments and middleware infrastructures, the idea of "situatedness" has been promoted by the adoption of shared virtual spaces for services and components interactions. The pioneering system Gaia [17] introduces the concept of active spaces, that is active blackboard spaces acting as the means for service interactions. Later on, a number of Gaia extensions where proposed to enforce dynamic semantic pattern-matching for service composition and discovery [7] or access to contextual information [6]. Other related approaches include LIME [13] and TOTA [11]. Our model shares the idea of conceiving components as "living" and interacting in a shared spatial substrate (of tuple spaces) where they can automatically discover and interact with one another. Yet, our aim is broader, namely, to dynamically and systemically enforce situatedness, service interaction and data management with a simple language of chemical reactions.

7 Conclusions

This article discussed the SAPERE approach to modeling self-organising and self-adaptive ecosystems. We briefly discussed the approach and overviewed a fragment of the SAPERE model and eco-laws language. We gave an example of application of the approach to model crowd evacuation, and validated the approach using simulation.

The example we made shows that the SAPERE approach is well-suited to support generalised descriptions of self-organising behavioural patterns. Through parameterisation of eco-laws and LSAs, the same set of rules could be used for different gradients. On the one hand this simplifies modeling, on the other hand it supports scalability and expandability. This is especially important if one does not want to impose closed ecosystems. A further level of openness can be achieved by semantic matching (not discussed in this paper).

SAPERE models can achieve resilience without the need to explicitly model error recovery or exceptional behaviour. This is thanks to their self-adaptive nature and to the possibility to modeling by simple patterns known to be resilient.

This project has started recently, and these are the first analyses of concrete scenarios. Perspectives for the immediate future include performing a larger set of simulations, including larger expo environments, effects of crowd formation in steering, and so on; analysis, modeling and simulation of further scenarios, with different types of complexity; a further refinement of the LSA and eco-law syntax (more readable and user-friendly); and finally, more advanced validation techniques, like probabilistic symbolic model checking.

References

1. Autili, M., Benedetto, P., Inverardi, P.: Context-aware adaptive services: The plastic approach. In: FASE '09 Proceedings. pp. 124–139. Springer-Verlag (2009)
2. Banâtre, J.P., Priol, T.: Chemical programming of future service-oriented architectures. Journal of Software 4, 738–746 (September 2009)
3. Berry, G., Boudol, G.: The chemical abstract machine. In: POPL. pp. 81–94 (1990)
4. Casadei, M., Gardelli, L., Viroli, M.: Simulating emergent properties of coordination in Maude: the collective sort case. In: Proceedings of the 5th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2006). Electronic Notes in Theoretical Computer Science, vol. 175(2), pp. 59–80. Elsevier Science B.V. (2007)
5. Ciocchetta, F., Duguid, A., Guerriero, M.L.: A compartmental model of the cAMP/PKA/MAPK pathway in Bio-PEPA. CoRR abs/0911.4984 (2009)
6. Costa, P.D., Guizzardi, G., Almeida, J.P.A., Pires, L.F., van Sinderen, M.: Situations in conceptual modeling of context. In: EDOC 2006. p. 6. IEEE-CS (2006)
7. Fok, C.L., Roman, G.C., Lu, C.: Enhanced coordination in sensor networks through flexible service provisioning. In: Field, J., Vasconcelos, V.T. (eds.) Proceedings of COORDINATION 2009. LNCS, vol. 5521, pp. 66–85. Springer-Verlag (2009)
8. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. The Journal of Physical Chemistry A 104(9), 1876–1889 (March 2000)
9. Iliasov, A., Laibinis, L., Romanovsky, A., Sere, K., Troubitsyna, E.: Towards rigorous engineering of resilient pervasive systems. In: Proceedings of Seventh European Dependable Computing Conference. IEEE Computer Society (May 2008)
10. Macal, C.M., North, M.J.: Tutorial on agent-based modelling and simulation. Journal of Simulation 4, 151–162 (2010)
11. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications: The tota approach. ACM Trans. Softw. Eng. Methodol. 18(4), 1–56 (2009)
12. Montagna, S., Viroli, M.: A framework for modelling and simulating networks of cells. In: Proceedings of the CS2Bio 2010 Workshop. ENTCS, vol. 268, pp. 115–129. Elsevier Science B.V. (Dec 2010)
13. Murphy, A.L., Picco, G.P., Roman, G.C.: Lime: A model and middleware supporting mobility of hosts and agents. ACM Trans. on Software Engineering and Methodology 15(3), 279–328 (2006)
14. Paun, G.: Membrane Computing: An Introduction. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2002)
15. Priami, C.: Stochastic pi-calculus. The Computer Journal 38(7), 578–589 (1995)
16. Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: Review and development recommendations. Simulation 82(9), 609–623 (2006)
17. Román, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: a middleware platform for active spaces. Mobile Computing and Communications Review 6(4), 65–67 (2002)
18. Roy, P.V., Haridi, S., Reinefeld, A., Stefany, J.B., Yap, R., Coupaye, T.: Self-management for large-scale distributed systems: an overview of the selfman project. In: Formal Methods for Components and Objects, LNCS No. 5382. pp. 153–178. Springer Verlag (2008)
19. Tchao, A., Risoldi, M., Di Marzo Serugendo, G.: Modeling self-* systems using chemically-inspired composable patterns. In: Proceedings of the 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems. IEEE-CS (2011)

20. Versari, C., Busi, N.: Efficient stochastic simulation of biological systems with multiple variable volumes. ENTCS 194(3), 165–180 (2008)
21. Viroli, M., Casadei, M.: Biochemical tuple spaces for self-organising coordination. In: Field, J., Vasconcelos, V.T. (eds.) Proceedings of COORDINATION 2009, LNCS, vol. 5521, pp. 143–162. Springer-Verlag (2009)
22. Viroli, M., Zambonelli, F.: A biochemical approach to adaptive service ecosystems. Information Sciences 180(10), 1876–1892 (2010)