Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 667–674

ISBN 978-83-60810-22-4

A Chemical Inspired Simulation Framework for Pervasive Services Ecosystems

Danilo Pianini
DEIS–Università di Bologna
via Venezia 52, 47521 Cesena, Italy
Email: danilo.pianini@unibo.it

Sara Montagna
DEIS–Università di Bologna
via Venezia 52, 47521 Cesena, Italy
Email: sara.montagna@unibo.it

Mirko Viroli
DEIS–Università di Bologna
via Venezia 52, 47521 Cesena, Italy
Email: mirko.viroli@unibo.it

***Abstract*—This paper grounds on the SAPERE project (Self-Aware PERvasive Service Ecosystems), which aims at proposing a multi-agent framework for pervasive computing, based on the idea of making each agent (service, device, human) manifest its existence in the ecosystem by a *Live Semantic Annotation* (LSA), and of coordinating agent activities by a small and fixed set of so-called *eco-laws*, which are sort of chemical-like reactions evolving the distributed population of LSAs. System dynamics in SAPERE is complex because of openness and due to the self-* requirements imposed by the pervasive computing setting: a simulation framework is hence needed for what-if analysis prior to deployment. In this paper we present a prototype simulator which – due to the role of chemical-like character of eco-laws – is based on a variation of an existing SSA (Stochastic Simulation Algorithm), tailored to the specific features of SAPERE, including dynamicity of network topology and pattern-based application of eco-laws. The simulator is tested on a crowd steering scenario where groups are guided, through public or private displays, towards the preferential destination and by emergently circumventing crowded regions.**

I. Introduction and Motivation

THE INCREASING evolution of pervasive computing is promoting the emergence of decentralised and complex infrastructures for pervasive services composed by new communication devices (e.g. mobile phones, PDA's, smart sensors, laptops). Such infrastructures 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). Mainstream languages and software infrastructures are often inadequate to face requirements of scalability, openness, adaptivity and self-organisation typical of pervasive systems. In order to better handle these scenarios, a paradigm shift towards agent world is receiving more and more attention in the scientific community. Agents support the implementation of distributed and communicating environments where different kind of autonomous entities (i.e. agents) are located. In this context, one of the hottest research topics regards agent coordination, namely the way an infrastructure can be built to allow agents to produce, consume and exchange information inside the pervasive system [31].

Different approaches were proposed in the area of coordination models and middlewares for pervasive computing scenarios: they try to account for issues related to spatiality [17], [21], spontaneous and opportunistic coordination [2], [10], self-adaptation and self-management [25]; however, they typically propose ad-hoc solutions to specific problems in specific areas, and lack generality.

The SAPERE project ("Self-adaptive Pervasive Service Ecosystems") addresses the issues above in a uniform way by means of a self-adaptive pervasive substrate, namely, a space bringing to life an ecosystem of individuals, which are pervasive services and devices able to interact with humans. The key idea is to coordinate agents in a self-organising way by basic laws (called *eco-laws*) that evolve the population of individuals in the system, enacting mechanisms of coordination, communication, and interaction [32]. Technically, such eco-laws are structured as sort of chemical reactions, working on the "interface annotation" that each agent injects in neighbouring localities, called *LSA* (Live Semantic Annotation).

In this context, simulation can be useful in supporting the design of eco-laws and agent behaviour, and ultimately, of whole pervasive service ecosystems. They give the possibility to experiment the idea of exploiting ecological mechanisms, such as those inspired by biology [9], showing through simulation the overall behaviour of a system designed on top of eco-laws, as well as to elaborate what-if scenarios. Moreover, a well designed framework will enable researchers to formally analyse the properties of such pervasive systems through stochastic model checking [13].

To capture the whole complexity of the SAPERE approach the model has to support in a coherent model the following abstractions: *(i)* highly dynamic environments composed of different, mobile, communicating nodes; *(ii)* reactive behaviours expressed by chemical-like reactions over LSAs; and *(iii)* autonomous behaviour of agents.

On one hand the adoption of standard Agent-Based Models (ABM) [16] seems to be quite natural, since the pervasive system itself is engineered adopting the agent paradigm and relying on a mediated form of interaction—as typical when using, e.g., the A&A metamodel [24]. There are several works which apply this approach in different contexts, from social systems (see, e.g., [3]) to biological systems [19], [4]. An ABM grounds around autonomous and possibly heterogeneous agents that can be situated in an environment. They carry out the most appropriate line of action, possibly interacting with other agents as well as the environment itself. The agent

behaviour is modelled through a set of rules which describe how the agent behaves according to environmental conditions. These rules can be of different types, according to the specific model / architecture: from simple reactive rules specifying how the agent must react to environmental stimuli or perceptions, to pro-active ones specifying how the agent must behave with respect to its goals and tasks [33]. In ABM the environment is also a first class abstraction whose structure, topology and dynamic can be explicitly modelled. To develop and simulate ABMs different simulation frameworks have been developed, such as MASON [15], Repast [22], NetLogo [26] and Swarm [28].

On the other hand, ABMs do not typically provide tools for sophisticated design of behavioural rules in the environment, at least up to the point of supporting an efficient simulation of chemical-like reactions as studied in [12] and its extensions. Examples of works going in this direction escape the field of ABMs, entering the scope of formal models for stochastic and bio-inspired concurrency models, namely, based on Stochastic Simulation Algorithms (SSA) such as BioPEPA [7] and BetaWB [8]. However, in this field few simulators allow to flexibly define network topologies [1] – they mostly deal with a single or few chemical compartments – and to the best of our knowledge no one provides features of network mobility and fine-tuning control over different behaviour in different nodes, for this typically escapes the context of biological systems. Additionally, SAPERE eco-laws do not fit exactly chemical reactions, for they handle structured molecules and advanced matching algorithms in a way that existing chemical simulators can hardly tackle.

To take the best of both approaches we developed a brand new simulation framework, called ALCHEMIST, meant to face natively the above requirements. It implements an optimised version of the Gillespie's SSA, namely the Next Reaction Method [11], extended with the possibility to have dynamic reactions, *i.e.* reactions that can be added or removed once the simulation runs due to network mobility, and adapted to the semantics of the eco-law language.

We exemplify the approach in a case study of crowd steering in pervasive computing, in which groups are guided towards locations based on their preference, along optimal paths and taking into account the presence of crowded regions which should be circumvented. We provide the set of eco-laws solving the problem (which adopts mechanisms proposed in the context of computational fields and spatial computing [17], [20]) and validate it via simulation of the associated stochastich model.

The remainder of this paper is organised as follows: Section II presents details about the computational model we defined and the simulation engine, Section III reports the model application at the crowd steering scenario and the simulation results and Section IV provides concluding remarks and discusses future works.

II. ENGINE ARCHITECTURE

In this section we first introduce how to model a chemical system in both deterministic and stochastic ways, then we show the known algorithms for stochastic simulation and our choices for a full featured high performance engine.

A. Stochastic Simulation Algorithms

A chemical system can be modelled as a single space filled with molecules that may interact through a number of reactions describing how they combine. The instantaneous speed of a reaction is called propensity and depends on the kinetic rate of the reaction and on the concentrations of all the reagents involved. For a reaction i with k reactants, j products, rate r of the form

$$R_0 + R_1 + \ldots + R_k \xrightarrow{r} P_0 + P_1 + \ldots P_j$$

the propensity a_i is defined as:

$$a_i = r \cdot [R_0] \cdot [R_1] \cdot \ldots \cdot [R_k]$$

where $[R_a]$ is the number of molecules of species R_a.

Since classical ODE (Ordinary Differential Equation) models are not accurate when the number of molecules in the system is low, a stochastic model has been proposed in [12]. This kind of description considers the whole system as a CTMC (Continuos-Time Markov Chain), in which the rate of the transition representing the i-th reaction is the propensity function a_i—and represents the average frequency at which the transition should be scheduled, following a negative exponential distribution of probability.

In [12], two algorithms are proposed in order to correctly simulate a stochastic path of a chemical system. Those algorithms were successively improved, but even their optimized versions, they rely on the idea that the system can be simulated by effectively executing the reactions one by one and changing the system status accordingly. Every algorithm follows four main steps:

1) select the next reaction μ to be executed;
2) calculate the time of occurrence of μ according to an exponential time distribution and make it the current simulation time;
3) change the environment status in order to reflect this execution;
4) update the propensities of the reactions.

The known techniques differ in the implementation of first and fourth steps. We will briefly present them and then justify our choice for the engine.

1) Direct Method: The direct method was first proposed in [12]. It chooses the next reaction to be executed by throwing a random number $r \leqslant \sum_i a_i$ and selecting the first reaction μ which verifies the property that $r \geqslant \sum_{i=0}^{\mu} a_i$. After the execution of μ, it updates propensities for each reaction.

2) Optimized Direct Method: The direct method can be optimised as proposed in [11] and [29] by introducing a binary search tree and a dependency graph. The former allows one to choose the next reaction μ to be executed in logarithmic time, the latter to update only the propensities of those reactions in which concentration of reagents is modified by the execution of μ.

3) Composition-Rejection Method: In [27] a constant time method relying on composition-rejection algorithm is proposed. The separation between the number of reactions R and the computational complexity of the algorithm is obtained by splitting the whole set of reactions into G groups, and then arguing that G does not depend (or depends loosely) by R. It may rely on a dependency graph in order to improve the update phase.

4) First Reaction: The First Reaction Method is the dual form of the Direct Method, and was proposed first in [12]. The key idea is to calculate immediately the time of occurrence for each reaction and select the next one to be executed using the lowest time. It is demonstrably the same of the Direct Method both in soundness and in time complexity.

5) Next Reaction: The Next Reaction Method is an optimised form of the First Reaction Method first proposed in [11]. It relies on an Indexed Priority Queue (IPQ) in order to smartly sort the reactions by time and has constant time in the selection phase since the root of the IPQ is always the next reaction to execute. A dependency graph can be used in order to update only the required propensities, and moreover a random-number re-usage is allowed, speeding up consistently the times recalculation.

B. Computational Model

Before discussing our choice of basic SSA algorithm, we describe the computational model we propose in order to fill the gap between the SAPERE world and chemical simulation. In fact, requirements on the model will necessarily influence some aspects of the design choices behind the engine itself.

First of all, we want to motivate the choice of chemical-resembling laws to model self-* behaviours. It has been proved that most ODE equations describing population dynamics can be translated in an equivalent CTMC passing through a set of chemical like reactions that describes the way the population entities interact [6]. This expressive power of chemical reactions is very interesting as soon as it allows us to use them in the design of pervasive systems inspired at ecological systems.

Our model extends the classic model of chemical reactions in three main directions.

First, in the classic chemical formulation [12], the environment is a single compartment that contains the molecules soup. This description is pretty far from the world we want to model, which is a pervasive service ecosystem. The natural extension is to consider many compartments (nodes) placed in a space (environment) which is responsible of linking them based on some rule. Depending on the specific environment, nodes can be dynamically added, moved or removed. A neighbourhood is consequently a structure which contains a node "centre" and a list of all linked compartments.

Second, in classical chemical models, a reaction lists a number of reactant molecules which, combined, produce a set of product molecules. This kind of description is too strict for our purposes. A more generic concept is to consider a reaction as a set of conditions about the environment which, when matched, may allow the execution of a set of actions. A condition is a function which associates a boolean to the current state of the environment, an action is a procedure which modifies it. The propensity function can no longer be simply the product of the reaction rate with the concentrations of the reactants, but needs a more generic definition too: propensity in our model is a function of the reaction rate, the conditions, and the environment state.

Third, we want to deal with events whose occurrence time does not follow an exponential law, like triggers events happening at a specific time regardless the previous evolution of the system—we want to occasionally depart the CTMC model for the sake of flexibile configuration of simulations. For instance, we may want to simulate an alarm event at a specific simulation time, such that one can run multiple simulations in order to understand how the system will react. Another usage of triggers appears when considering the possibility to interact with a running simulation pausing it and, exploiting triggers, interact with the environment in its current status, then resume the simulation. Even if this approach is not useful when the goal is to check the properties of a model, it could be very handy when exploring and testing it.

C. Dynamic engine

Given the model we want to simulate described in Section II-B and the algorithms presented in Section II-A, we can argue that no existing algorithm as-is is appropriate to support our simulations. In particular, no algorithm provides facilities to add and remove reactions dynamically and to inject triggers and other non-exponential time distributed events. Our choice for the engine algorithm to extend was then restricted between the First Reaction and the Next Reaction, because they are the only that choose the next reaction to execute explicitly considering the time of occurrence, which makes a lot simplier to support non-exponential time distributed events. The latter is an optimization of the former, offers a lower computational complexity in every case and consequently can achieve higher performance. Our work had the primary goal to extend Next Reaction providing the possibility to add and remove reactions dynamically, since to the best of our knowledge no work in this sense have been ever made. In order to add this support, it is a mandatory task to provide methods to add and remove reactions from the indexed priority queue and the dependency graph.

1) Dynamic Indexed Priority Queue: A key property of the original Indexed Priority Queue proposed in [11] is that the swap procedure used to update the data structure does not change the balance of the tree, ensuring optimal update times in every situation. This feature was easily achieved because no

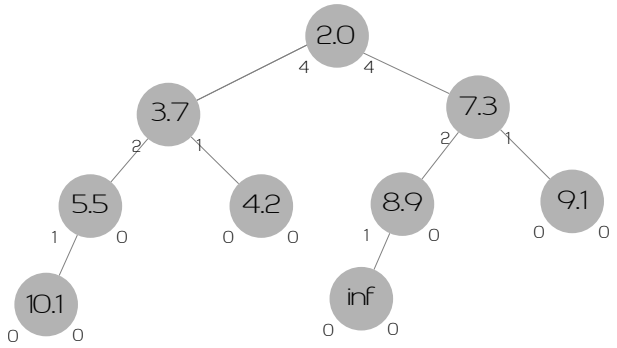

Fig. 1. Indexed Priority Queue extended with children count per branch

nodes were ever added neither removed from the structure. As a consequence, once the tree is balanced at creation time no event can occur to change its topology. This is no longer the case, and we have to provide a small extension to the structure in order to manage the balancing. Our idea is, for each node, to keep track of the number of children per branch, having in such way the possibility to keep the tree balanced when adding nodes. In figure 1 we show how the same IPQ drawn in [11] would appear with our extension. In the following algorithms, the procedure `UPDATE_AUX(n)` is the same described in [11]. Given this data structure, the procedure to add a new node `n` is the following:

```
IF root does not exists
   n is the new root
ELSE
   c <- root
   WHILE c has two children
      IF c.right < c.left
         dir <- right
      ELSE
         dir <- left
      add 1 to count of dir children
      c <- c.dir
   IF c has no left child
      n becomes left child of c
      set count of left nodes of c to 1
   ELSE
      n is right child of c
      set count of right nodes of c to 1
UPDATE_AUX(n)
```

The removal procedure for a node n is the following:

```
c <- root
WHILE c is not a leaf
   IF c.left > c.right
      dir <- left
   ELSE
      dir <- right
   subtract 1 to count of dir children
   c <- c.dir
IF c != n
   swap c and n
   remove n
   UPDATE_AUX(c)
ELSE
   remove n
```

Using the two procedures described above, the topology of the whole tree is constrained to remain balanced despite the dynamic addition and removal of reactions.

2) Dynamic Dependency Graph: Since we want to support natively and efficiently the dependencies among multiple compartments, we defined three contexts (also called scopes): `local`, `neighborhood` and `global`. Each reaction has an input context and an output context, meaning respectively where data influencing the rate calculus is located and and where the modifications are made.

The first issue to address is to evaluate if a reaction `r1` may influence another reaction `r2`, considering their contexts. We introduced a boolean procedure `mayInfluence(r1, r2)` which operates on two reactions and returns a true value if:

- `r1` and `r2` are on the same node OR
- `r1`'s output context is `global` OR
- `r2`'s input context is `global` OR
- `r1`'s output context is `neighborhood` and `r2`'s node is in `r1`'s node neighbourhood OR
- `r2`'s input context is `neighborhood` and `r1`'s node is in `r2`'s node neighbourhood OR
- `r1`'s output context and `r2`'s input context are both `neighborhood` and the neighbourhoods of their nodes have at least one common node.

Given this handy function, we can assert that a dependency exists between the execution of a reaction `r1` and another reaction `r2` if `mayInfluence(r1,r2)` is true and at least a molecule whose concentration is modified by `r1` is among those influencing `r2`.

Adding a new reaction implies to verify its dependencies against every reaction of the system. In case there is a dependency, it must be added to the graph. Removing a reaction `r` requires to delete all dependencies in which `r` is involved both as influencing and influenced. Moreover, in case of change of the system topology, a dependencies check among reactions belonging to nodes with modified neighbourhood is needed. It can be performed by scanning them, calculating the dependencies with the reactions belonging to new neighbours and deleting those with nodes which are no longer in neighbourhood.

D. Engine architecture

The whole framework has been designed to be fully modular and extensible. The whole engine or parts of it can be re-implemented without touching anything in the model, and on the other hand the model can be extended and modified without messing with the engine. This modularity allows to easily make some experiments with other engines, such as Composition-Rejection.

The framework, called ALCHEMIST, was developed from scratch using Java. Being performances a critical issue for a simulator, we compared some common languages in order to evaluate their performance level. Surprisingly, Java performance are at same level of compiled languages such as C/C++ [5], [23]. The Java language was consequently chosen because of the excellent trade off among performances, easy portability and maintainability of the code, plus the support for concurrent programming at language level. The COLT Java library [14] provided us the mathematical functions we needed. In particular, it offers a fast and reliable random number generation algorithm, the so called Mersenne Twister [18].

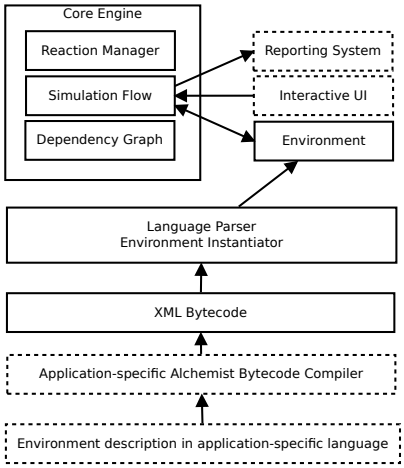

Fig. 2. ALCHEMIST architecture. Elements drawn with continuous lines indicates components common for every scenario and already developed, those with dotted lines are extension-specific components which have to be developed with the specific application in mind.

ALCHEMIST is still actively developed and currently consists of 188 classes for a total of 16527 lines of code.

As shown in Figure 2, at the current status of development the simulations are written in a specific XML language containing a complete description of environment and reactions. This code is interpreted in order to produce an instance of an environment, once it is created, no further interpretation is needed in order to run the simulation. This XML code is not meant to be directly exploited by users, but it represents a way to describe environments in a machine-friendly way and is a formalisation of the generic model of ALCHEMIST. The idea behind this choice is that ALCHEMIST is flexible enough to be used in various contexts, each one requiring a personalised language and a different instantiation of the model. It's up to the extensor to write a translation module from its personalised language to the ALCHEMIST XML.

III. CASE STUDY

We propose a crowd steering scenario as a case study to demonstrate the possibility to exploit eco-laws to lead people in the desired location within a complex environment in short time, avoiding obstacles such as crowded regions and without global supervisioning.

Consider a museum with a set of rooms, whose floor is covered with a network of computational devices (infrastructure nodes). These devices can exchange information with each other based on proximity, sense the presence of visitors, and hold information about expositions currently active in the museum. Each room has four exits and they are connected via external corridors. Visitors wandering the museum are equipped with a hand-held device that holds the visitor's preferences. By interaction with infrastructure nodes, a visitor can be guided towards rooms with a target matching their interest, thanks to signs dynamically appearing on his smartphone. This is done, using techniques suggested in the field of spatial computing [30]—namely, computational gradients injected in a source and diffusing around such that each node holds the minimum distance from source.

A. A SAPERE model

The environment is made of infrastructure nodes. Smartphones are agents dynamically linked with the nearest sensors – the neighbours are the sensors inside a certain radius r, parameter of the model – from which they can retrieve data in order to suggest visitors where to go. Visitors are agents which tend to follow the advices of their hand-held device. They can 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.

In the SAPERE approach, all the information exchanged is in form of LSAs, and the rules are expressed in form of eco-laws. 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. 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, structured types, or function names.
There are three forms of LSAs used in this scenario:

$$\langle \texttt{source}, id, type, N_{max}, \pi, \mu, type' \rangle$$
$$\langle \texttt{field}, id, type, value, \pi, \mu, type', tstamp \rangle$$
$$\langle \texttt{pre\_field}, id, type, value, \pi, \mu, type', tstamp \rangle$$

A **source** LSA is used as a source with the goal of generating a field: id labels the source so as to distinguish sources of the same type; $type$ indicates the type of fields (`target` is used to advertise expositions, and `crowd` to diffuse information about crowding); N_{max} is the field's maximum value; π and μ are two functions used respectively to compute the new field value once it has to be propagated or transformed according to the value of another field of type $type'$—their purpose will be described more in details later, along with eco-laws. A **field** LSA is used for individual values in a gradient: $value$ indicates the individual value; the $tstamp$ reflects the time of creation of the LSA; the other parameters are like in the source LSAs. A **pre_field** LSA is used to diffuse the field before it is influenced by the transformation rule.

An eco-law is a chemical-resembling reaction working over patterns of LSAs. One such pattern P is basically an LSA which may have some variable in place of one or more arguments of a tuple, and an LSA L is matched to 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)*

$$\langle \texttt{source}, id, type, N_{max}, \pi, \mu, type' \rangle \xrightarrow{r_{init}} \langle \texttt{source}, id, type, N_{max}, \pi, \mu, type' \rangle, \langle \texttt{field}, id, type, N_{max}, \pi, \mu, type', \#T \rangle$$

$$\langle \texttt{field}, id, type, N, \pi, \mu, type', t \rangle \xrightarrow{r_{diff}} \langle \texttt{field}, id, type, N, \pi, \mu, type', t \rangle, +\langle \texttt{pre\_field}, id, type, \pi(N, \#D), \pi, \mu, type', t \rangle$$

$$\langle \texttt{pre\_field}, id, type, N, \pi, \mu, type', t \rangle, \langle \texttt{field}, id', type', M, \pi', \mu', Type, t' \rangle \mapsto \langle \texttt{field}, id, type, \mu(N, M), \pi, \mu, type', t \rangle, \langle \texttt{field}, id', type', M, \pi', \mu', Type, t' \rangle$$

$$\langle \texttt{field}, id, type, N, \pi, \mu, type', t \rangle, \langle \texttt{field}, id, type, M, \pi, \mu, type', t' + t \rangle \mapsto \langle \texttt{field}, id, type, M, \pi, \mu, type', t' + t \rangle$$

$$\langle \texttt{field}, id, type, N, \pi, \mu, type', t \rangle, \langle \texttt{field}, id, type, M, \pi, \mu, type', t \rangle \mapsto \langle \texttt{field}, id, type, max(M, N), \pi, \mu, type', t \rangle$$

$$\langle \texttt{field}, id, type, N, \pi, \mu, type', t \rangle, \langle \texttt{field}, id', type, N + M, \pi, \mu, type', t' \rangle \mapsto \langle \texttt{field}, id', type, N + M, \pi, \mu, type', t' \rangle$$

Fig. 3. Eco laws describing the museum application.

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. If no rate is given the reaction is meant to be executed "as soon as possible", which means that the rate that associated with the reaction tends to infinite. 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. In Figure 3, the eco-laws for our case study are given.

As sources LSAs are injected in nodes, gradients are built by the first three rules in Figure 3. The first eco-law, given a source, initiates the field with its possible maximum value. The second eco-law, when a node contains a field LSA, spreads a pre_field LSA to a neighbouring node picked up randomly with a new value computed according to the propagation function, π, which elaborates the distance between sensors – indicated by the variable $\#D$ – and the actual value of the field LSA. The third eco-law, when a node contains a pre_field LSA of type $type$ and a field LSA of type $type'$ by which the pre_field depends, removes the pre_field LSA and creates a new field LSA with a value computed according to the transformation function, μ, which elaborates the values N and M of the two reactants. The purpose of this law is to model the interactions between fields. For instance we may assume that if there is a crowd which jams a region of the museum, that path towards the target should have less probability of being picked, so the value of the target field has to be reduced. As a consequence of these laws, each node will carry a field LSA indicating the topological distance from the source. The closest is the field value to N_{max}, the nearest is the field source. When the spread values reach the minimum value 0, the gradient has to become a plateau.

To address the dynamism of the scenario where people move, targets being possibly shifted, and crowds forming and dissolving, we introduced the following mechanism. We expect that if a gradient source moves the diffused value has to change according to the new position. This is the purpose of the $tstamp$ parameter which is used in the fourth eco-law, continuously updating old values by more recent ones (*youngest* eco-law). In this way we ensure that the system is able to adapt to changes of the source states. Finally, the spreading eco-law above may produce duplicate values in locations, due to multiple sources of the same type (indicated by different ids), multiple paths to a source, or even diffusion of multiple LSAs over time. For this reason we introduced the last two eco-laws. They retain only the maximum value, i.e. the minimum distance, the former when there are two identical LSAs with only a different value, the latter when the id is different (*shortest* eco-laws).

Eco-laws in Figure 3 describe the behaviour of the museum ecosystem in a chemical-oriented fashion, and are accordingly modelled in the simulator as reactions.

People are modelled as agents. They move according to the field value computed for their target probabilistically choosing the neighbour with higher field value. The behaviour of visitors is modelled as a reaction too, featuring a special action in which the behaviour of the visitors is expressed.

The proposed architecture is intrinsically able to dynamically adapt to unexpected events (like node failures, network isolation, crowd formation, and so on) while maintaining its functionality.

B. Simulator configuration and results

The behaviour of each node is programmed according to the eco-laws coordination model explained in Figure 3. Each node in the environment contains by default, for each type in the system – `target` and `crowd` – , an LSA of the form $\langle \texttt{field}, id, type, 0, \pi, \mu, type', 0 \rangle$. The sources of the gradients are injected by sensors when a target or a number of persons is perceived, with the values

$$\langle \texttt{source}, id, \texttt{target}, \texttt{T}_{\texttt{max}}, \pi_t, \mu_t, \texttt{crowd} \rangle$$
$$\langle \texttt{source}, id, \texttt{crowd}, \texttt{C}_{\texttt{max}}, \pi_c, \mu_c, \texttt{crowd} \rangle$$

For the first kind of source we assume that we can have different targets according to different preferences of users. For the crowding source instead, we may assume that sensors are calibrated so as to locally inject an LSA indicating the level

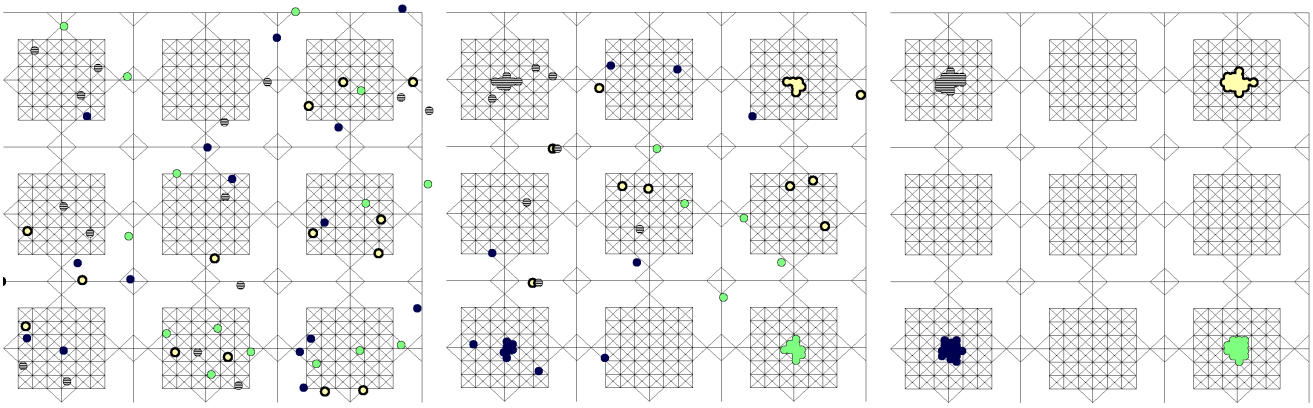

Fig. 4. A simulation run of the reference exposition: three snapshots of the ALCHEMIST graphic reporting module with this simulation

of crowding, *i.e.* if the number of persons, and is periodically updated by the sensors.

The propagation and transformation functions have the following form:

$$\begin{aligned} \pi_t = \pi_c &= N_{t,c} - \#D \\ \mu_t &= N_t - k * N_c \\ \mu_c &= N_c \end{aligned}$$

where $N_{t,c}$ are the actual values of the two types of fields, and k is a model parameter use to modulate the effect that the crowd can have on the target field. Other parameters are: T_{max} = 1000, C_{max} = 1. When $\mu_t, \pi_{t,c} < 0$, whe impose them to be zero.

The reaction rates are identified by hand performing different simulations with different parameters. The results reported below are obtained with $r_{init} = 1$ and $r_{diff} = 50$. The other laws show no rate because it is assumed to be infinite.

We here present simulations conducted over an exposition, where nine rooms are connected via corridors. People can express different preferences represented by their colour.

Four snapshots of a first simulation run are reported in Figure 4. We here consider four different targets that are located in the four rooms near environment angles. People are initially spread randomly in the museum, as shown in the first snapshot, and they eventually reach the room in which the desired target is hosted, as shown in the last snapshot.

Figure 5 shows a simulation experimenting with the effect of crowding in the movement of people. Two groups of people – denoted with empty and full circles – with common interests are initially located in two different rooms, as shown in the first snapshot. The target for the dark visitors is located in the central room of the second row, while the others' is in the right room of the second row. In the simulation, dark visitors reach their target soon before it is nearer, though forming a crowded area intersecting the shortest path towards the target for the other visitors. Due to this jam the latter visitors choose a different path that is longer but less crowded.

IV. CONCLUSION

In the SAPERE metaphor, the ideal level of abstraction to reach in order to easily and correctly model and simulate pervasive systems requires both the rich environment of ABM and the native CTMC model support of biochemistry-oriented stochastic simulators. In this work we shown the ALCHEMIST simulation framework, meant to fully support this way to think pervasive systems. This framework embraces the SAPERE vision and allows to approach the simulation of agent systems in a new flavour, describing the system in terms of reaction-like laws and having consequently the possibility to rely on all the work already made about CTMC. We shown a case study whose complexity overcomes the expressiveness possibility of classical biochemistry-oriented simulation frameworks, and we analysed it exploiting the same CTMC mathematical support. Perspectives for the immediate future include a deeper analysis of performance for the proposed case study, tuning parameters so as to identify the most proper extent to which a crowd should influence movement of people. Then we mean to compare performance and expressiveness with respect to ABM simulation frameworks, such as Repast and NetLogo. Finally, we shall analyse, model and simulate further scenarios, with different types of complexity so as to stress the potentialities of ALCHEMIST.

ACKNOWLEDGMENT

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

REFERENCES

[1] R. Alves, F. Antunes, and A. Salvador. Tools for kinetic modeling of biochemical networks. *Nature Biotechnology*, 24(6):667–672, June 2006.

[2] M. Autili, P. Benedetto, and P. Inverardi. Context-aware adaptive services: The plastic approach. In *FASE '09 Proceedings*, pages 124–139, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] S. Bandini, S. Manzoni, and G. Vizzari. Crowd Behavior Modeling: From Cellular Automata to Multi-Agent Systems. In A. M. Uhrmacher and D. Weyns, editors, *Multi-Agent Systems: Simulation and Applications*, Computational Analysis, Synthesis, and Design of Dynamic Systems, chapter 13, pages 389–418. CRC Press, June 2009.

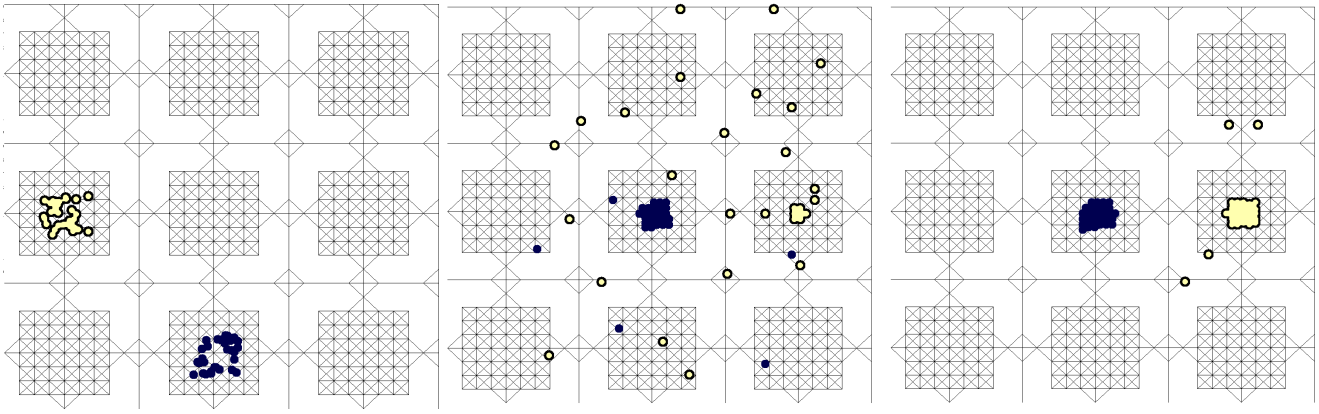

Fig. 5. A run showing the effect of crowding: dark visitors occupy a central room, making other visitors moving left to right by a longer, less crowded path

[4] G. Beurier, F. Michel, and J. Ferber. A morphogenesis model for multiagent embryogeny. In L. M. Rocha, L. S. Yaeger, M. A. Bedau, D. Floreano, R. L. Goldstone, and A. Vespignani, editors, *Artificial Life X*, pages 84–90. MIT Press, Cambridge, MA, 2006.

[5] J. M. Bull, L. A. Smith, C. Ball, L. Pottage, and R. Freeman. Benchmarking java against c and fortran for scientific applications. *Concurrency and Computation: Practice and Experience*, 15(3-5):417–430, 2003.

[6] L. Cardelli. From processes to odes by chemistry. In *IFIP TCS*, pages 261–281, 2008.

[7] F. Ciocchetta and M. L. Guerriero. Modelling biological compartments in Bio-PEPA. *Electronic Notes in Theoretical Computer Science*, 227:77–95, 2009.

[8] L. Dematté, C. Priami, A. Romanel, and O. Soyer. Evolving blenx programs to simulate the evolution of biological networks. *Theoretical Computer Science*, 408(1):83–96, 2008.

[9] J. L. Fernandez-Marquez, J. L. Arcos, G. Di Marzo Serugendo, M. Viroli, and S. Montagna. Description and composition of bio-inspired design patterns: the gradient case. In *Proceedings of the 3rd Workshop on Bio-Inspired and Self-* Algorithms for Distributed Systems*, Karlsruhe, Germany, 14 June 2011. ACM.

[10] C.-L. Fok, G.-C. Roman, and C. Lu. Enhanced coordination in sensor networks through flexible service provisioning. In J. Field and V. T. Vasconcelos, editors, *Proceedings of COORDINATION 2009*, volume 5521 of *LNCS*, pages 66–85. Springer-Verlag, 2009.

[11] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876–1889, 2000.

[12] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, December 1977.

[13] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In B. Steffen and G. Levi, editors, *Proc. 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2004.

[14] W. Hoschek. *The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java*. CERN, Geneva, 2004. Available at http://acs.lbl.gov/software/colt/.

[15] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. C. Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[16] C. M. Macal and M. J. North. Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4:151–162, 2010.

[17] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications: The tota approach. *ACM Trans. Softw. Eng. Methodol.*, 18(4):1–56, 2009.

[18] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[19] S. Montagna, N. Donati, and A. Omicini. An agent-based model for the pattern formation in Drosophila Melanogaster. In H. Fellermann, M. Dörr, M. M. Hanczyc, L. Ladegaard Laursen, S. Maurer, D. Merkle, P.-A. Monnard, K. Stoy, and S. Rasmussen, editors, *Artificial Life XII*, chapter 21, pages 110–117. The MIT Press, Cambridge, MA, USA, 2010.

[20] S. Montagna, M. Viroli, M. Risoldi, D. Pianini, and G. Di Marzo Serugendo. Self-organising pervasive ecosystems: A crowd evacuation example. In *Proceedings of the 3rd International Workshop on Software Engineering for Resilient Systems*, Lecture Notes in Computer Science, Geneva, Switzerland, 2011. Springer. In Press.

[21] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A model and middleware supporting mobility of hosts and agents. *ACM Trans. on Software Engineering and Methodology*, 15(3):279–328, 2006.

[22] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos. A declarative model assembly infrastructure for verification and validation. In S. Takahashi, D. Sallach, and J. Rouchier, editors, *Advancing Social Simulation: The First World Congress*, pages 129–140. Springer Japan, 2007.

[23] B. Oancea, I. G. Rosca, T. Andrei, and A. I. Iacob. Evaluating java performance for linear algebra numerical computations. *Procedia CS*, 3:474–478, 2011.

[24] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3), June 2008.

[25] P. V. Roy, S. Haridi, A. Reinefeld, J.-B. Stefany, R. Yap, and T. Coupaye. Self-management for large-scale distributed systems: an overview of the selfman project. In *Formal Methods for Components and Objects, LNCS No. 5382*, pages 153–178. Springer Verlag, 2008.

[26] E. Sklar. Netlogo, a multi-agent simulation environment. *Artificial Life*, 13(3):303–311, 2007.

[27] A. Slepoy, A. P. Thompson, and S. J. Plimpton. A constant-time kinetic monte carlo algorithm for simulation of large biochemical reaction networks. *The Journal of Chemical Physics*, 128(20):205101, 2008.

[28] S. D. Team. http://www.swarm.org/index.php/Main_Page. Swarm home page.

[29] C. Versari and N. Busi. Efficient stochastic simulation of biological systems with multiple variable volumes. *Electr. Notes Theor. Comput. Sci.*, 194(3):165–180, 2008.

[30] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli. Spatial coordination of pervasive services through chemical-inspired tuple spaces. *ACM Transactions on Autonomous and Adaptive Systems*, 6(2):14:1 – 14:24, June 2011.

[31] M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, July 2007.

[32] M. Viroli and F. Zambonelli. A biochemical approach to adaptive service ecosystems. *Information Sciences*, 180(10):1876–1892, 2010.

[33] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 1st edition, June 2002.