Exploring the Dynamics of Self-Organising Systems with Stochastic π-Calculus: Detecting Abnormal Behaviour in MAS

Luca Gardelli*, **Mirko Viroli, Andrea Omicini**
DEIS, Alma Mater Studiorum—Università di Bologna
via Venezia 52, 47023 Cesena, Italy
{luca.gardelli, mirko.viroli, andrea.omicini}@unibo.it

Abstract

The intrinsic complexity of self-organising MASs (multi-agent systems) makes it difficult to predict global system evolutions at early stages of the design process. Simulating high-level models to analyse properties of a MAS design can anticipate detection of incorrect / wrong design choices, and allow tuning of system parameters.

In this paper, we take abnormal-behaviour detection as a case study, and devise an artifact-based MAS architecture inspired by principles of the human immune systems. We use stochastic π-calculus to specify and run quantitative large-scale simulations, which allow us to verify the basic applicability of our IDS (intrusion detection system) and possibly obtain a preliminary set of its main working parameters.

1 Introduction

The trend in today information systems engineering is toward an increasing degree of complexity and openness, leading to rapidly-changing requirements and highly-dynamic environments. Furthermore, the increasing costs of systems engineering [Horn, 2001] call for new engineering methodologies and tools. In that sense social and natural sciences are recognised as rich sources of inspiration: for instance, the Autonomic Computing initiative tries to face complexity by applying self-regulating mechanisms typical of biological processes [Kephart and Chess, 2003].

Self-organisation is a promising theoretical framework to reduce complexity of systems engineering. A system is said to be *self-organising* if it is able to reorganise itself upon environmental changes, by local interaction of its parts without any explicit pressure from the outside [Heylighen, 2003]. A system built according to this principle is usually able to perform complex tasks even though its components are far simpler when compared to a monolithic solution.

*This work is partly supported by the European Science Foundation (ESF) Minema Scientific Programme, Grant No. 805.

In this paper we elaborate along the line developed in [Gardelli *et al.*, 2005b; 2005a] by exploring methodological aspects of the engineering of self-organising MASs. Because of the complexity inherent in such systems, as well as the difficulties in predicting their behaviour and properties, we find it crucial to exploit formal tools for simulating systems dynamics at the early stages of design, in order to (i) nurture evolving ideas and design choices, and (ii) to provide better foundation for the feasibility of the system. In the deployment stage, the models devised can then provide suitable parameters for (iii) tuning the system.

Among the various formal models to specify quantitative aspects of MASs we promote the use of the stochastic π-calculus process algebra [Milner *et al.*, 1992; Priami, 1995]. Process algebras have already been successfully applied to several domains [Baeten, 1990]: now their value is even more recognised in fields like security [Abadi and Gordon, 1997], performance analysis [Gilmore and Hillston, 1994], social insects [Tofts, 1992], molecular biology [Regev *et al.*, 2001], but still almost unexplored in the context of self-organising and autonomic systems engineering.

In this paper we show that the π-calculus process algebra can be fruitfully applied in particular to the engineering of multi-agent systems (MASs): in particular we will show how to track the dynamics of global system properties via stochastic simulations, and determining suitable system parameters.

In this paper, we apply these ideas to the study of an intrusion detection (ID) infrastructure for open MASs: the focus is on detecting anomalies in agents behaviour drawing inspiration from principles of the human immune system [Forrest *et al.*, 1997]. The infrastructure we devise is based on the TuCSoN technology [Omicini and Zambonelli, 1999][1]. This allows us to structure a MAS not only in terms of agents, but also with *tuple centres* [Omicini and Denti, 2001] as *coordination artifacts* [Omicini *et al.*, 2004] and *agent coordination contexts* (ACC) [Omicini, 2002]. Coordination artifacts are used to model resources in the environment on which agents act upon. ACCs specify and enact the access policies which each agent is subject to, and can be used to both *(i)* reify relevant information about the agent/artifacts interaction, and

[1] http://tucson.sourceforge.net

(ii) to deny malicious agents access to the resources available in the environment.

To evaluate the impact of different design choices and parameters of the ID infrastructure—such as inspection/detection rates, number of inspectors, and the like—we simulate the system behaviour in different scenarios using SPiM specifications. SPiM—Stochastic Pi Machine—is a simulator that is able to execute systems specifications written in stochastic π-calculus and track the quantitative system dynamics [Phillips, 2005].

The rest of the article is structured as follows. In Section 2 we briefly highlight the subject of ID and related mechanisms in the human immune system. In Section 3 we describe our reference architecture for a MAS based on TuCSoN, and show how to develop an anomaly detection application based on artifacts. Section 4 motivates the use of π-calculus and its stochastic extension. Then Section 5 discusses simulations related to our case study, along with some meaningful experimental results. Finally, Section 6 concludes with final remarks, and with some hints on the main directions for our future research.

2 Intrusion Detection and the Human Immune System

There are several mechanisms used to protect information systems, but usually only the basic ones are implemented: *(i) authentication*—identity is proved by the knowledge of a secret (e.g. password) or a physical unique property (e.g. fingerprint, retina, voice); *(ii) authorisation*—user actions on the system are constrained by its role and the policy linked to that role.

However, application flaws typically cause these methods alone not to be sufficient [Forrest *et al.*, 1997]. Furthermore, authorisation policies cannot account for all possible sequences of actions, and a specific sequence might exhibit unexpected side-effects: it is in general too expensive and impractical—or even unfeasible—to intercept all emergent harmful paths at design-time.

Hence, automated tools are essential to enable run-time detection of malicious behaviours: in this direction, many efforts have already been spent in developing IDSs. An IDS tries to detect intruders and misuse of a target software system by observing users behaviour and deciding whether actions performed are symptomatic of an attack. *Misuse-based* IDSs try to detect intruders matching the actual user behaviour with known signatures of malicious behaviour. *Anomaly-based* IDSs try to detect behaviours that are different from what it is considered to be the normal activity. Both approaches have been inspired by the inner mechanisms of the human immune system [Forrest *et al.*, 1997; Somayaji *et al.*, 1997], and are widely explored in the literature. In this paper, we draw useful principles from the human immune system and apply them at the architectural level, while relying on previous work by IDS community for the algorithmic aspects.

In our case study we consider three main features: independent layers, distribution and adaptivity. In order to achieve better protection we design our security system as made of *independent layers*:

- static non-specific mechanisms—like the skin of animals—are implemented through authentication;
- static specific mechanisms are typically realised by authorisation policies;
- dynamical issues are the main concern of anomaly and misuse detection, and resembles the human acquired immune system.

Responsibilities of our security system should be *distributed across several entities*—like lymphocytes in acquired immune system—in order to avoid the single-point-of-failure problem. The dynamic layer should raise its defences if under attack while releasing computational resources when not needed, i.e., *adaptively*: this point in particular is the main subject of investigation of this article. We follow the "self-organisation" approach, i.e., designing a strategy based on simple rules exploiting coupled positive/negative feedback loops. We investigate stochastic π-calculus models of several strategies, and run simulations for several parameters values using the Stochastic Pi-Machine (SPiM) [Phillips, 2005].

3 Security in MAS

In this section we describe our reference architecture for MASs based on the TuCSoN coordination infrastructure [Omicini and Zambonelli, 1999], showing how this fits an anomaly detection layer.

We consider a system that provides agents with services encoded in terms of coordination artifacts, i.e., runtime abstractions encapsulating and providing a coordination service, to be exploited by agents in social contexts expressed by coordination rules and norms [Omicini *et al.*, 2004]. Following the general model of artifact [Viroli *et al.*, 2005], a coordination artifact could be characterised by a usage interface, a set of operating instructions, and a coordination behaviour specification, which can be exploited rationally by cognitive agents.

Accesses of agents to these resources is restricted by an authentication procedure. When an agent enters the system an authorisation policy limits its actions allowing the exploitation of a limited set of services and resources—e.g. those it has payed for. An agent acts upon the environment via a run-time structure, the ACC, which enables and rules the interaction between the agent and the environment [Omicini, 2002]— so the ACC is the right place where to embed authorisation policies. The whole architecture is depicted in Figure 1.

Even if the two mechanisms of authentication and authorisation are considered to guarantee a reasonable degree of protection, we promote the idea—as pointed out in the ID community—that a system is better protected by additional dynamic mechanisms. Correspondingly, we introduce a layer aimed at detecting anomalies in agents behaviour inspired by the

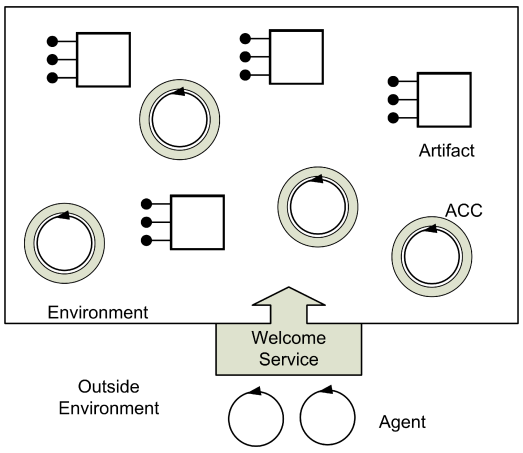

Figure 1: In our reference architecture of MASs, environments are populated by artifacts and agents.

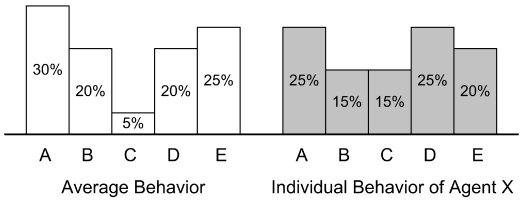

Figure 2: Comparison between the average behaviour of agents and individual one as the basis for anomaly detection: possible actions are grouped into classes.

immune system as well as by previous works on IDSs [Forrest *et al.*, 1997; Somayaji *et al.*, 1997].

We consider agents willing to exploit a specific artifact: an artifact provides a finite set of services, but we can group them into classes. For the sake of simplicity we consider only five classes from A to E. If we trace the agents behaviour "for a while" then we are able to create an average distribution of actions over that resource: that distribution is taken as the "normal way" for agents to interact with a particular artifact (Figure 2 left). Note that this approach is valid under two hypotheses:

1. the number of traces is such that the data is statistically significant;
2. the percentage of agents exhibiting abnormal behaviour is sufficiently low during the initial observation stage.

From now on it is possible to observe an individual agent in order to build its particular distribution of actions: the deviation from the average distribution might be a symptom of intrusion or abnormal activity. For instance, if an action belonging to class C is critical then agent X (Figure 2 right) should be inspected to decide whether it is acting properly or might cause problems. Note that there exist better approaches than the one above described: indeed for us it is just a case to investigate a methodology based on formal languages and simulation.

Referring to the previous section we describe now how the security layer fits into the general architecture. Basically we need three artifacts, (i) for providing the service, (ii) for logging purpose and (iii) to report abnormal activity. Due to space constraints we cannot provide the details on how to program the various artifacts to achieve the various behaviours: interested readers can refer to [Gardelli *et al.*, 2005a] for an overview, or download the whole source code.[2]

We consider two classes of agents: the ones that request services and the ones that perform inspection. The main role of an inspector is to compare the average signature with the individual ones and report any anomaly: the inspection for obvious reasons cannot be performed over all the agents but it will be done through sampling. We can consider two parameters related to an inspector: (i) the rate of inspection and (ii) the number of inspectors active simultaneously. Note that it is functionally equivalent to have one inspector working at rate kr_{insp} or k inspectors working at rate r_{insp}: as previously stated distributed entities let us avoid the single-point-of-failure problem.

In the remainder of this paper we will briefly introduce the π-calculus process algebra and show how it can be fruitfully applied to this case so as to predict the global behaviour of the system and choose a strategy for managing the number of inspectors.

4 The π-Calculus

The π-calculus is a formal language developed to reason about concurrency [Milner *et al.*, 1992]. Initially, it was intended for describing and analysing systems consisting of agents (or processes) which interact with each other. The basic entity is the *name*, which is used as an unstructured reference to a synchronous channel where messages can be sent and received. The semantics of π-calculus can be described by a transition system, where the transition relation $P \longrightarrow P'$ —a process P moving to P' by the occurrence of an inner interaction—is defined by operational rules [Milner *et al.*, 1992].

In general, each formal model whose semantics is given by a transition system can be extended to a stochastic version, resorting to the idea of Markov transition system. There, each transition $P \xrightarrow{r} P'$ is labelled with a *rate* r, a non-negative real value that describes how the transition probability between P and P' increases with time.

Stochastic models allow for quantitative simulations, for rates can be used to express aspects such as probability, speed, delays, and so on. One of the notable stochastic extensions to the π-calculus was introduced by Priami [Priami, 1995]. Each channel name is associated with an activity rate r: the delay of an interaction through that channel—representing the use of a resource—is then a random variable with an exponential distribution defined by r. Given a channel name x, the probability p_i of a transition $P \xrightarrow{r_i} P_i$ representing an interaction through x is the ratio between

[2]The source code for the experiments, π-calculus specifications and charts can be downloaded from `http://www.alice.unibo.it/download/spim/at2ai-5.zip`.

its rate r_i and the sum of rates of the n transitions through x enabled by P:

$$p_i = \frac{r_i}{\sum_{j=1..n} r_j}, \quad 1 \leq i \leq n \ . \tag{1}$$

Because of space constraints we are not going to describe here how to write stochastic π-calculus specifications since it has already been widely covered by the literature—see for instance [Milner *et al.*, 1992; Phillips and Cardelli, 2004]. In the next section we discuss how to exploit stochastic π-calculus in order to simulate the dynamics of the previously-described anomaly detection system: we use the Stochastic pi Machine (SPiM) [Phillips, 2005] in order to run simulations directly from the specifications.

5 Simulating the Dynamics of an Anomaly Detection System

In this section we focus on the description of models for anomaly detection systems, and on the analysis of simulation results: interested readers can download the specifications used to derive the simulations, along with the coloured version of the plots.

The main objective here is to evaluate a number of different strategies letting inspectors adapt to the changing number of malicious agents: in other words, we would like the system to *self-regulate*—raising its defences when needed, otherwise releasing computational resources. For the sake of simplicity we consider here neither *missed detections* nor *false alarms*: while these are key evaluation parameters for detection algorithms, the focus here is on the overall qualitative MAS behaviour rather than on performances.

In the basic scenario, we consider an agent that performs inspections at a fixed rate drawing randomly an agent from the population: the initial population of the system is made of 90 normally-behaving agents and 10 abnormally-behaving ones. While the number of normal agents remains constant, malicious ones enter the system at an arbitrary rate: hence, if the rate of inspection is 10.0 then *statistically* the rate of inspection of malicious agents is at least 1.0 and increases as more malicious agents enter the environment.

Once this model is specified in stochastic π-calculus, the dynamics of the system can be simulated by using SPiM: each simulation adopted a different[3] arrival rate for the malicious agents. From the chart in Figure 3 it is possible to observe—although it was easily predictable—that an agents inspecting once about every 10 time units is able to contain malicious agents only in case the arrival rate is sufficiently low.

We now extend the basic model in order to change the number of inspector agents according to the number of malicious agents, which is an unknown from the viewpoint of inspectors. Drawing inspiration from the human immune system and the well-known predator-prey system, we introduce the ability of inspectors to clone themselves. In particular the stimulus for the cloning process is the anomaly-detection event: hence, every time an inspector detects a malicious agent it clones itself, while it dies if it inspects a normal agent. The infrastructure takes care that there is at least an inspector active. The anomaly-detection event triggers a positive feedback on the number of inspectors, while the death upon inspection of a normal agent provides the negative feedback necessary to stabilise the population of both inspectors and malicious agents—see chart in Figure 4. Chart in Figure 5 shows that whatever the rate of arrivals of malicious agents, the system always reaches a dynamic equilibrium. While the strategy described above is somewhat effective, the equilibrium achieved might not be suitable—i.e., the average number of malicious agents may still be too high.

So, we experiment with a new strategy, which is a refinement of the previous one: we introduce a memory of previous attacks—a mechanism inspired by the human immune system. To this end, we add the life-

[3]For the sake of clarity, the charts included show data only for a couple of the parameters values in order to highlight the trends.

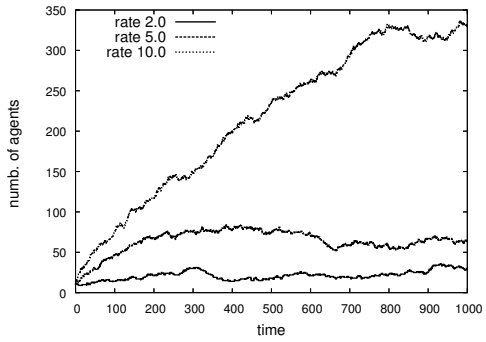

Figure 3: The population of malicious agents rapidly grows by increasing the rate of arrivals. With a single inspector, the system is able to contain malicious agents only in the case that the rate of arrivals is comparable to the rate of inspections.

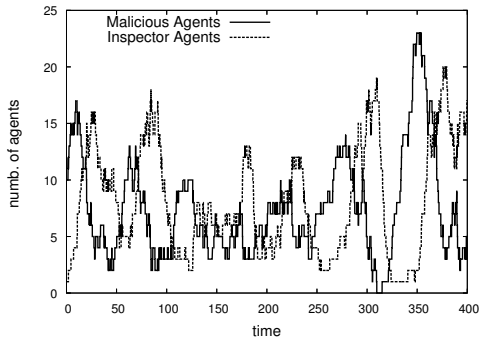

Figure 4: The dynamic equilibrium showed in this system is very similar to the one exhibited by a prey-predator-like system. The size of the malicious population evolves periodically due to the adaptively-changing inspectors population.

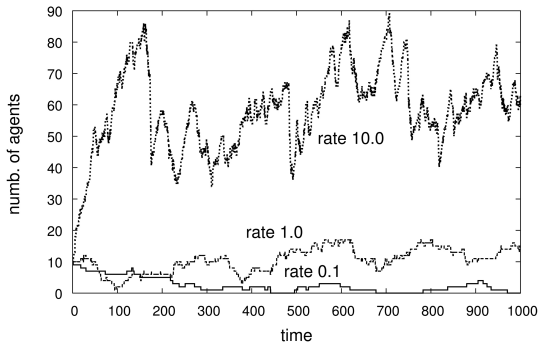

Figure 5: Increasing the rate of arrivals of malicious agents the population reach a dynamic equilibrium which might however be unsuitable: this is achieved using inspectors able to clone themselves.

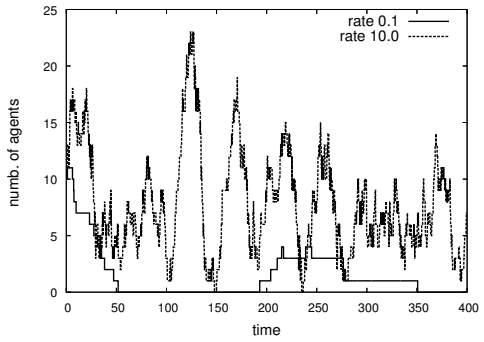

Figure 6: Adding the lifetime feature to inspectors results in a better equilibrium achieved: lifetime of inspectors is constant while the rate of arrivals increases. Compare this chart with the one in Figure 5.

time property to inspectors: each time an agent performs an inspection its lifetime is decreased, but if it detects a malicious agent then its lifetime is extended. In this sense the lifetime is measured in terms of number of inspections allowed. It is possible to notice from the chart in Figure 6 that the equilibrium reached is much more reasonable—i.e., the number of malicious agents is lower—hence the system is able to limit the activity of malicious agents. In particular, the chart in Figure 7 makes the relation between inspectors' lifetime and the system dynamic equilibrium mostly evident: as the lifetime value increases, the equilibrium stabilises at lower values. Note that, given the target quality-level, there is trade-off between lifetime value and use of computational resources.

6 Conclusion and Future Works

One of the general aims of this research line is paving the way toward an agent-oriented framework for engineering self-organising applications. In [Gardelli *et al.*, 2005a] we initially focused on the applicability of stochastic π-calculus and the related modelling pro-

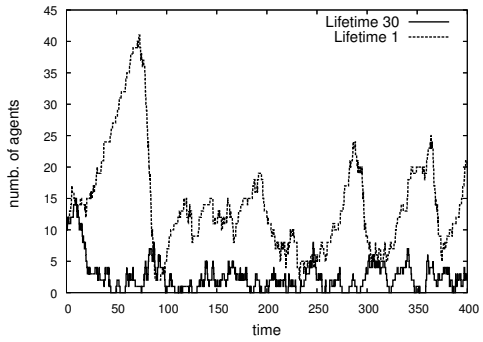

Figure 7: Holding constant the rate of arrivals of malicious agents, it is possible to observe that the longer the inspectors' lifetime, the better the dynamic equilibrium achieved. Lifetime is measured in terms of number of inspections allowed.

cess. Here, instead, we gave more details on domain-specific issues, i.e., to support anomaly detection for MAS and targeting the simulations to the specific case. This paper is largely based on [Gardelli *et al.*, 2005b], where we showed how to exploit π-calculus specifications to quickly investigate the dynamics of self-organising systems. Here, however, a clear experimental evidence has been given to the emergence of global system properties such as the dynamic equilibrium achieved in prey-predator like systems, thus providing more support for our claims.

According to our reference architecture a MAS is populated both by agents and artifacts: the latter embed resources or services to be exploited by agents. The system depicted is based on the TuCSoN coordination model, and has been prototyped upon the TuCSoN infrastructure. For the architecture and general principles we took inspiration from the human immune system and previous works on IDSs. For the methodology, we relied on simulations and modelling via stochastic π-calculus, which—even though is a quite new language in the context of the MAS community—showed its effectiveness as a tool to investigate several design strategies.

While our experiments need to be further detailed, we believe they generally emphasise the ability of the proposed approach to help the MASs developers to anticipate design decisions and strategies at the early stages of design, before actually developing prototypes and testing them.

References

[Abadi and Gordon, 1997] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *4th ACM conference on Computer and Communications Security (CCS'97)*, pages 36–47, New York, NY, USA, 1997. ACM Press.

[Baeten, 1990] Jos C. M. Baeten, editor. *Applications of Process Algebra*, volume 17 of *Cambridge Tracts*

in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, September 1990.

[Forrest *et al.*, 1997] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[Gardelli *et al.*, 2005a] Luca Gardelli, Mirko Viroli, and Andrea Omicini. On the role of simulation in the engineering of self-organising systems: Detecting abnormal behaviour in MAS. In Flavio Corradini, Flavio De Paoli, Emanuela Merelli, and Andrea Omicini, editors, *AI*IA/TABOO Joint Workshop "Dagli oggetti agli agenti: simulazione e analisi formale di sistemi complessi" (WOA 2005)*, pages 85–90, Camerino, MC, Italy, 14–16 November 2005. Pitagora Editrice Bologna.

[Gardelli *et al.*, 2005b] Luca Gardelli, Mirko Viroli, and Andrea Omicini. On the role of simulations in engineering self-organizing MAS: the case of an intrusion detection system in TuCSoN. In Sven Brueckner, Giovanna Di Marzo Serugendo, David Hales, and Franco Zambonelli, editors, *3rd International Workshop "Engineering Self-Organising Applications" (ESOA 2005)*, pages 161–175, AAMAS 2005, Utrecht, The Netherlands, 26 July 2005.

[Gilmore and Hillston, 1994] Stephen Gilmore and Jane Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In Günter Haring and Gabriele Kotsis, editors, *Computer Performance Evaluation*, number 794 in LNCS, pages 353–368. Springer, 1994. Modelling Techniques and Tools. 7th International Conference, Vienna, Austria, 3–6 May 1994. Proceedings.

[Heylighen, 2003] Francis Heylighen. The science of self-organization and adaptivity. In *Knowledge Management, Organizational Intelligence and Learning, and Complexity*, The Encyclopedia of Life Support Systems. EOLSS Publishers, 2003. Available online at http://www.eolss.net.

[Horn, 2001] Paul Horn. Autonomic computing: IBM's perspective on the state of information technology. Technical report, IBM Corporation, 15 October 2001.

[Kephart and Chess, 2003] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

[Milner *et al.*, 1992] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I. *Information and Computation*, 100(1):1–40, September 1992.

[Omicini and Denti, 2001] Andrea Omicini and Enrico Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, November 2001.

[Omicini and Zambonelli, 1999] Andrea Omicini and Franco Zambonelli. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.

[Omicini *et al.*, 2004] Andrea Omicini, Alessandro Ricci, Mirko Viroli, Cristiano Castelfranchi, and Luca Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, editors, *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 19–23 July 2004. ACM.

[Omicini, 2002] Andrea Omicini. Towards a notion of agent coordination context. In Dan C. Marinescu and Craig Lee, editors, *Process Coordination and Ubiquitous Computing*, chapter 12, pages 187–200. CRC Press, October 2002.

[Phillips and Cardelli, 2004] Andrew Phillips and Luca Cardelli. Simulating biological systems in the stochastic pi-calculus. Technical report, Microsoft Research, Cambridge, UK, July 2004.

[Phillips, 2005] Andrew Phillips. The Stochastic Pi Machine (SPiM), 2005. Version 0.041 available online at http://www.doc.ic.ac.uk/~anp/spim/.

[Priami, 1995] Corrado Priami. Stochastic pi-calculus. *Computer Journal*, 38(7):578–589, 1995.

[Regev *et al.*, 2001] Aviv Regev, William Silverman, and Ehud Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *6th Pacific Symposium on Biocomputing (PSB 2001)*, volume 6, pages 459–470, Hawaii, USA, 3-7 January 2001. World Scientific Press.

[Somayaji *et al.*, 1997] Anil Somayaji, Steven Hofmeyr, and Stephanie Forrest. Principles of a computer immune system. In Tom Haigh, Bob Blakley, Mary Ellen Zurbo, and Catherine Meodaws, editors, *1997 Workshop on New Security Paradigms (NSPW'97)*, pages 75–82, New York, NY, USA, 23–26 September 1997. ACM Press.

[Tofts, 1992] Chris Tofts. Describing social insect behaviour using process algebra. *Transactions of the Society for Computer Simulation*, 9:227–283, December 1992.

[Viroli *et al.*, 2005] Mirko Viroli, Andrea Omicini, and Alessandro Ricci. Engineering MAS environment with artifacts. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, pages 62–77, AAMAS 2005, Utrecht, The Netherlands, 26 July 2005.