ENGINEERING THE ENVIRONMENT OF SELF-ORGANISING MULTI-AGENT SYSTEMS EXPLOITING FORMAL ANALYSIS TOOLS

Luca Gardelli, Mirko Viroli, Matteo Casadei
Dipartimento di Elettronica, Informatica e Sistemistica
Alma Mater Studiorum-Università di Bologna
[luca.gardelli, mirko.viroli, m.casadei]@unibo.it

We briefly discuss about the role of the environment in a self-organising system and how to apply self-organisation principles to build a multi-agent system (MAS) environment. We describe a design methodology for complex systems featuring emergent properties: our approach mainly relies on the use of formal analysis tools and languages in order to provide better guarantees of effective global system behaviour. In order to explain the details of the methodology, we apply it to a case of dynamic resource allocation strategy for a MAS environment.

1. Self-Organisation and MAS Environment

The typical MAS scenario involves a set of autonomous situated entities able to interact with each other and the environment in order to achieve a common goal: as pointed out in [1], since agents are limited in capabilities, they cannot control, neither completely perceive, the global dynamics of the system. On the other hand, self-organising systems (SOSs) are systems composed of entities with such a partial control, and which locally interact in order to create or maintain an ordered structure under environment perturbations: in these systems—due to the intricacy of interactions—it may happen that global phenomena emerge [2].

The MAS community started considering self-organisation theory as an inspiration for systems development only in the last few years, specifically since the first applications inspired by ants behaviour. Since then, several initiatives driven by the same vision flourished such as Amorphous Computing [3] and Autonomic Computing [4].

In this article we focus on the idea that the environment is the key element in the engineering of a MAS featuring self-organising properties. In the definition of self-organising system there is a crucial aspect which is sometime underestimated: “under environment perturbations” means that the environment plays an active role, which may be interpreted as hostile due to the fact that it perturbs agent situation. Typically, agents have to learn—by evolution or other mechanisms—how to reduce the possibly disrupting effect of the environment: for example flocking reduces wind friction, while schooling let fishes avoid predators [2]. Indeed, it is the agent-environment coupling that creates the feedback loop, balancing between positive and negative feedback [2].

Furthermore, the environment can also be seen as a collective memory used for self-organisation purposes, as it can be designed to retain the history of the

actions of the agents [5]. For example consider pheromone trails in ant colonies, where the environment collects the pheromone laid by the ants, performing aggregation, diffusion and evaporation processes. Hence, self-organisation is considered as a mean for “reducing agent complexity”, because part of that complexity is delegated to—i.e. is hidden into—the environment.
A dual aspect is that environment design may benefit from using self-organisation principles. In general, it is desirable that a software system self-configures and automatically recovers from errors, i.e. that it adopts self-healing—a basic principle of Autonomic Computing [4]. The environment might exploit self-organising techniques for the management of services to be provided to agents: for example, in this article we briefly describe a case of dynamic resource allocation, though other applications such as security [6, 7] could be used.
As far as engineering the environment is relevant in the development of self-organising MAS, in this paper we focus on methodological aspects. We note that currently none of the AOSE methodologies—see e.g. Gaia [8] and SODA—take into account self-organisation issues. The few exceptions, such as ADELFE [9], still do not promote any kind of practice to guarantee effectiveness of global system properties. For example, how can we design the environment features of a MAS exploiting stigmergic-coordination so as to guarantee that the agents will eventually find a path between the source and the target location? Based on initial explorations we developed in [10, 7], our goal here is to outline a methodological approach to address this kind of questions.

2. Towards a Methodology for Designing Self-Organising Systems

Our approach plugs into the design stage of system development, and can be possibly used in combination with existing methodologies—future works in this context will pursue this research line. The first observation is that it is not very clear—both for agent-based and traditional methodologies—how the deliverable of the analysis process maps onto a specific design. There is often an early design stage where several approaches have to be evaluated, for example making preliminary feasibility tests: in this stage we do not want to deal with all the details of a complete design.
The first step of our approach occurs in this stage: we use formal languages to specify abstract models of possible architectural solutions, in order to nurture evolving ideas. Specifically, in the case of self-organising systems where complex patterns arise from low level interaction, selective models can make us focus on the most crucial properties of interest. The use of formal languages for this scope has several advantages like unambiguity and precise selection of the subsystem to model, as well as opening to the possibility of using general purpose tools to perform automatic analysis.
These formal models are then used in the second stage to generate simulation runs, where expected behaviours of the system are previewed and analysed. Simulation is a very useful tool to provide a first feedback on the suitability of a solution: if the simulation results are not satisfying another approach can be tested or some parameter can be adjusted—i.e. we go back to the first phase until we get useful results. Though it is widely believed that simulations may be the only tool to investigate self-organising systems and emergent phenomena,

we observe that from an engineering point of view they do not provide actual guarantees: the final system might in some specific cases exhibit completely different behaviours.
In spite simulation is a valuable tool both to observe qualitative behaviours and provide a coarse set of system parameters, we also see potentialities in formal analysis tools. Indeed, in the third stage we thus envision the use of model checking tools to verify system properties [11], and specifically emergent ones.
The model checking process involves both system specifications and statements about the properties to verify: these elements are used by the model checker to explore the states space of the target system in order to test if the statements hold [11]. These statements might be expressed in several formalisms and affect the model checking algorithm: in common languages we can express statements such as "*will deadlock eventually occur in the system?*". Traditional model checking algorithms deal only with finite state deterministic systems [11], which are apparently not good approximations for a self-organising system. However, it is often possible to constrain the state space, or to apply abstraction techniques to see a syntactically infinite system as a semantically finite one [12]. Moreover, model checking algorithms exist that are able to handle probabilistic systems [13]: in this model it is possible to express statements such as "*is there a probability greater than 50% that the property X will eventually occur ?*". Other examples also consider time, and can deal with statements like "*will the system reach the state S within 5 seconds with a probability greater than 90% ?*".
As far as self-organising systems are concerned, good models of system behaviour should be able to describe stochastic phenomena, i.e. phenomena which duration and/or execution time is aleatory. Sometimes, stochasticity is inherent to the problem subject to investigation and cannot be abstracted away: for example the time to execute an agent's action could be an aleatory variable. In general, emergence of properties appears to be intrinsically related to stochasticity, hence, we would like to investigate how it may affect the way to model check emergent system properties. At the end of this investigation we would like to be able to answer questions such as "*will the emergent property X occur in the system within 5 minutes with a probability greater than 95% ?*". Answering that kind of statements will help us to better understand the dynamics of self-organising systems and improve the design and engineering process of modern complex systems. Applying forthcoming research results in the context of model checking for stochastic systems is hence an important part of our research.

3. Designing a MAS Environment

In this section we describe an example application of this approach, whose simplicity should help focussing on the most relevant details. We want to deploy a MAS, where agents can exploit resources provided by the environment[1]. Due to scalability and quality of service issues, we would like the environment to adaptively allocate resources depending on the number of requests.

[1] For the sake of clarity, in this example we consider only a single kind of resource and abstract away the inner working details.

Hence, the objective is to design a dynamic resource allocation (DRA) strategy. From Section 2, we recall that the first stage is about formalising candidate solutions without providing all the details of a real design. Since the number of agents inside the system is an aleatory variable, any scheduling algorithm which makes assumptions about the number of agents cannot be applied. A simple strategy that promotes a self-organising style is such that: (i) each resource that receives a request clones itself, (ii) after an arbitrary idle time the resource is deallocated. We can now model the above strategy using a formal language like pi-calculus—as we already did in [7]—or any other formal language that suits the requirements listed in Section 2: in particular we use the Maude tool [14] because it eventually supports our approach at each stage[2].
A formal model enables investigation by simulating system dynamics. Having analysed the simulation results for several parameters values, we found three qualitative behaviours: (i) insufficiency of resources, (ii) dynamic equilibrium and (iii) redundancy of resources. In particular, the second one adapts the resource number to the actual number of agents: if the arrival rate of agents is constant, we observe a periodical evolution of the number of agents to be served and the number of free resources, see Figure 1. Notice that the dynamic equilibrium condition is an emergent property of the system!

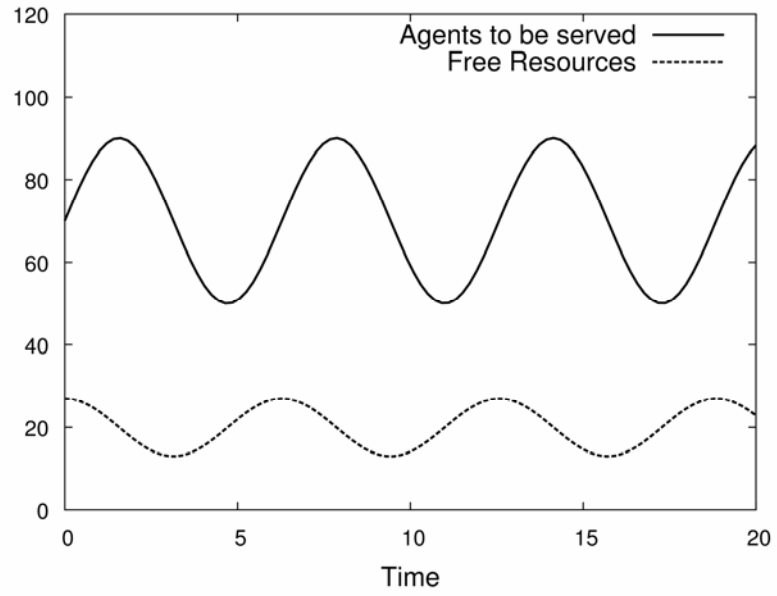

Figure 1 The chart shows the behaviour of the system in the dynamic equilibrium condition: the number of free resources follows the number of agents to be served with an offset of $\pi/2$.

From the first two stages we have a formal model of the strategy, we have analysed simulation results providing evidence of a qualitative behaviour—see Figure 2—and we have devised a coarse set of parameters to tune the system.

[2] Details about the use of Maude and the model specifications can be downloaded from www.alice.unibo.it/download/spim/aica-2006.zip

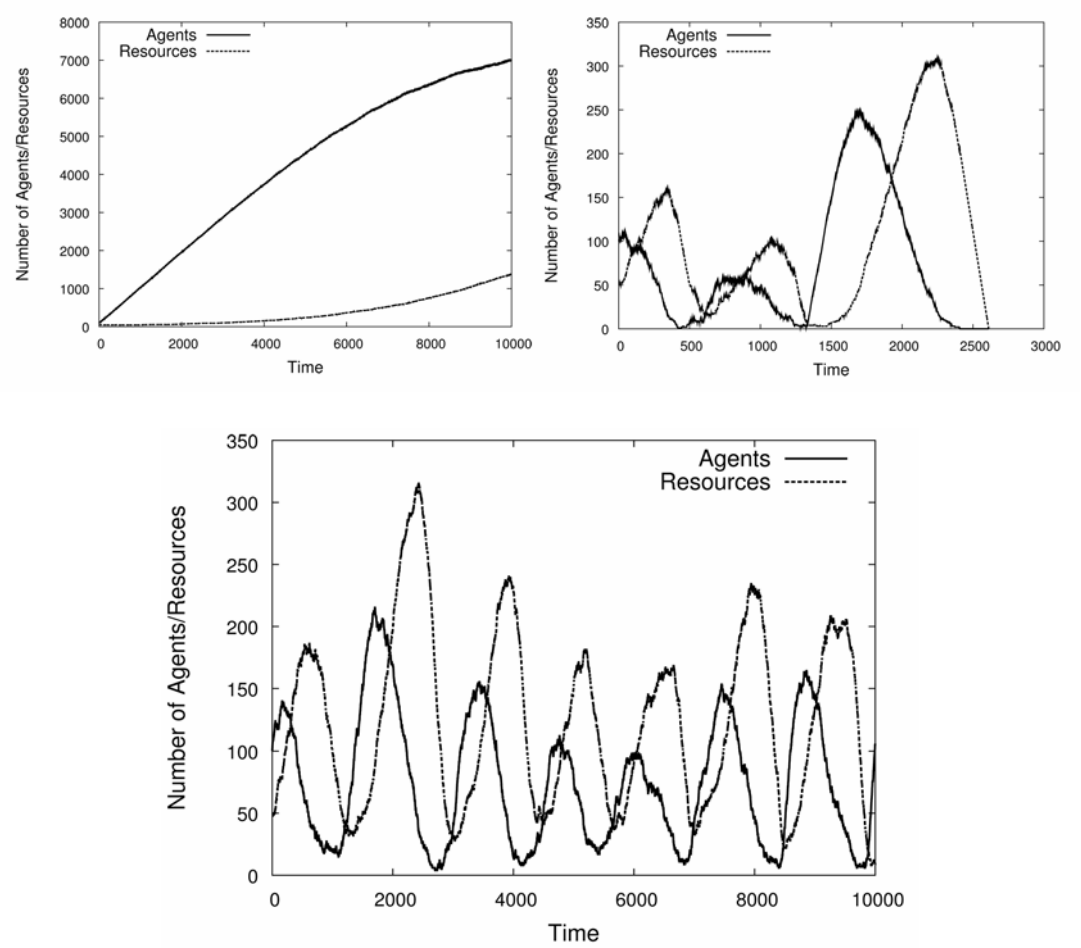

Figure 2 The system modelled exhibits three qualitative behaviours: insufficiency of agents (left), redundancy of resources (right), dynamic balance between agents and resources (bottom).

The third stage would be to about giving guarantees of the observed behaviour, i.e. verifying properties of the system via model checking: this is very important, since the system includes stochastic phenomena and each simulation run is different. Referring to the DRA strategy, an interesting property to verify might be "*will the number of free resources lie in the range (A,B) with a probability greater than 95% ?*": this statement assumes that the parameters are tuned in order to have a dynamic equilibrium condition and the range is adjusted according to the parameters values. If this condition is verified, we have stronger guarantees—with respect to simulations results—that the system will operate in the desired working condition: in other words, there will not be neither insufficiency nor redundancy of resources.
We are currently working on the implications of extending model-checking to stochastic processes, since Maude implements only traditional model checking algorithms [14].

4. Ongoing and Future Works

In this article we elaborated on the relevance of self-organisation theory to the engineering of MAS environments, highlighting the essentiality of providing

guarantees about emergent properties. We outlined a basic design methodology for the engineering of system with emergent properties, focussing on the role of formal languages and model checking algorithms. Due to space constraint we have not described issues related to tools or implementation.
We made some experiences with the Maude tool, a meta-programming language in the declarative style, suitable for modelling systems and their dynamics. More precisely, Maude is a high-performance reflective language supporting both equational and rewriting logic specifications for programming a wide range of applications [14]. Among the other features, Maude provides a Linear Temporal Logic (LTL) model checker which can be used to verify whether a specification satisfy some safety and liveness properties.
We developed a framework to quickly simulate system dynamics based on the Gillespie's algorithm [15], and modelled the DRA case. We are currently evaluating model checking algorithms for verifying properties of stochastic concurrent systems.
We plan to further investigate these topics as we believe that self-organising systems really depends on the notion of environment, and that the environment will surely benefit from the development of self-organising techniques.

5. References

[1] Muller, J.P., 2004, *Emergence of collective behaviour and problem solving*, In: Fourth International Workshop in Engineering Societies in the Agents World (ESAW 2003), Volume 3071 of Lecture Notes in Computer Science., Springer-Verlag (2004) 1–21

[2] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E., 2001, *Self-Organization in Biological Systems*, Princeton Studies in Complexity, Princeton University Press

[3] Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Thomas F. Knight, J., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R., 2000, *Amorphous computing,* Communications of the ACM 43(5) 74–82

[4] Horn, P., 2001, *Autonomic computing: IBM's perspective on the state of information technology*, Online at http://www.research.ibm.com/autonomic/manifesto

[5] Parunak, H.V.D., 1997, *Go to the ant: Engineering principles from natural multi-agent systems*, Annals of Operations Research 75 69–101 (Special Issue on Artificial Intelligence and Management Science)

[6] Gardelli, L., Viroli, M., Omicini, A., 2005*, On the role of simulation in the engineering of self-organising systems: Detecting abnormal behaviour in MAS*, AI*IA/TABOO Joint Workshop "Dagli oggetti agli agenti: simulazione e analisi formale di sistemi complessi" (WOA 2005), Camerino, MC, Italy, Pitagora Editrice Bologna 85–90

[7] Gardelli, L., Viroli, M., Omicini, A., 2006, *Exploring the dynamics of self-organising systems with stochastic pi-calculus: Detecting abnormal behaviour in MAS*, Fifth International Symposium From Agent Theory to Agent Implementation (AT2AI-5), Vienna, Austria

[8] Zambonelli, F., Jennings, N.R., Wooldridge, M., 2003, *Developing multiagent systems: The Gaia methodology*, ACM Transactions on Software Engineering and Methodology (TOSEM) 12(3) 317–370

[9] Bernon, C., Gleizes, M.P., Peyruqueou, S., Picard, G.: Adelfe, 2003, *A methodology for adaptive multi-agent systems engineering*, Engineering Societies in the Agents World III. Volume 2577 of LNAI, Springer 156–169

[10] Gardelli, L., Viroli, M., Omicini, A., 2006, *On the role of simulations in engineering self-organizing MAS: the case of an intrusion detection system in TuCSoN*, Engineering Self-Organising Applications III, Volume 3910 of LNAI, Springer 153–166

[11] Edmund M. Clarke, Orna Grumberg, D.A.P., 1999, *Model checking*, MIT Press, Cambridge, Massachusetts, USA

[12] Finkel, A., Ph. Schnoebelen, 2001, *Well-structured transition systems everywhere!,* Theoretical Computer Science 256(1-2) 63–92

[13] Kwiatkowska, M., Norman, G., Parker, D., 2002, *Prism: Probabilistic symbolic model Checker*, Computer Performance Evaluation : Modelling Techniques and Tools 12th International Conference, Volume 2324 of Lecture Notes in Computer Science 200–204

[14] Clavel, M., Duran, F., Eker, S., Lincoln, P., Martì-Oliet, N., Meseguer, J., Talcott, C., 2005, *Maude Manual*, Department of Computer Science University of Illinois at Urbana-Champaign. Version 2.2 edn, Version 2.2 is available online at http://maude.cs.uiuc.edu.

[15] Gillespie, D.T., 1997, *Exact stochastic simulation of coupled chemical reactions,* The Journal of Physical Chemistry 81(25) 2340–2361