Design Patterns for Self-Organizing Multiagent Systems

Luca Gardelli, Mirko Viroli, Andrea Omicini
ALMA MATER STUDIORUM—Università di Bologna
Via Venezia, 52 - 47023 Cesena, Italy
{ luca.gardelli, mirko.viroli, andrea.omicini } @ unibo.it

Abstract

Natural systems are currently being regarded as rich sources of inspiration for engineering artificial systems, particularly when adopting the multiagent system (MAS) paradigm. To promote a systematic reuse of mechanisms featured in self-organizing systems, we analyse a selected list of design patterns for recurrent problems in the literature. Starting from our reference MAS metamodel, we provide a complete characterization of each pattern according to a reference scheme: in particular we describe the problem, the solution with respect to our metamodel, the natural systems which have inspired the pattern and known applications. Furthermore, to contextualize the patterns within an engineering workflow, we briefly describe our methodological approach for designing self-organizing MAS.

1 Introduction

Self-organization is a very compelling approach to the engineering of complex systems because of its many interesting properties, including adaptivity and robustness: unfortunately, they provide difficult-to-face design challenges, too. However, from the analysis of natural systems, it is possible to identify successful strategies and encode them in the shape of patterns [3, 11]. First introduced in 1977 by Alexander in architectural design [1], the concept of *pattern* has become popular in computer science with the object-oriented paradigm [13]. A design pattern provides a reusable solution to a recurrent problem in a specific domain: it is worth noting that a pattern does not describe an actual design, rather it encodes an abstract model of the solution using specific entities of the paradigm in use. The use of design patterns offers several advantages such as reducing design-time by exploiting off-the-shelf solutions and promoting collaboration by providing a shared ontology. Researchers involved in Multiagent System (MAS) domain already synthesized several patterns: notable early contributions in that direction were mostly concerned with the life-cycle of agents, providing solutions related to resource access, mobility and basic social skills [15, 2, 12]. These patterns have been very useful in determining the functionalities offered by agents platforms and environments.

In order to build a systematic catalog of patterns, it is necessary to describe each pattern with respect to a known scheme. The very basic scheme consists in name, problem, solution and consequences: for the object-oriented paradigm a more specific and verbose scheme is provided in [13]. As pointed out in [17, 12], since the agent paradigm cannot be effectively characterised using object-oriented abstractions, patterns for MAS should be described using specific schemes. In particular, a candidate scheme should reflect the peculiarities of the target MAS metamodel: to this purpose, several pattern classification and schemes have been proposed [17, 10].

The main contribution of this article is the characterization of patterns for self-organizing MAS, with respect to our metamodel and a reference pattern scheme. It is worth noting that, since the patterns described encode basic mechanisms, they may seem too general to be useful: nonetheless, we believe that building complex patterns on top of simpler ones—as done in [3]— may allow for a deeper understanding of systems dynamics, hence improving controllability. So, we do not cover here patterns of the granularity of *stigmergy* or field-based coordination—as done in [11]. Instead, we isolate the basic patterns that, when properly combined, produce the complex patterns of self-organizing systems: e.g. in the case of stigmergy the basic patterns are evaporation, aggregation and diffusion. Furthermore, as discussed later, basic patterns may be useful also when used individually.

The remainder of the article is structured as follows: in Section 2 we describe our MAS metamodel based on agents and environmental resources, i.e. artifacts. We continue by analysing each pattern, namely Replication, Collective Sort, Evaporation, Aggregation and Diffusion: as far as the pattern scheme is concerned, we adhere to the one described in [17]. The reuse of patterns is a very crucial practice [10], hence, in Section 3 we contextualise the reuse of patterns in our methodological approach. Then, we conclude in Section 4 by providing final remarks and listing future research directions.

2 Patterns of Self-Organizing Systems

2.1 Our Reference MAS Metamodel

The MAS paradigm is generally acknowledged to be the best paradigm for modelling natural systems and developing artificial self-organizing systems. Briefly, agents are autonomous entities, situated in an environment that agents can perceive and affect. Although being recognised as one, the MAS paradigm is captured from different perspectives in different metamodels emphasising specific features: metamodels are often classified by the relevance of a specific features, e.g. the environment. When dealing with self-organizing systems the most relevant features are a strong notion of environment and support for indirect interactions between agents.

We rely on the agents & artifacts metamodel (A&A) we developed in the last few years [24]: a MAS is modelled by two fundamental abstractions, *agents* and *artifacts*. Agents are autonomous pro-active entities encapsulating control, driven by their internal goal/task. When developing a MAS, sometimes entities require neither autonomy nor pro-activity to be correctly characterised. This is typical of entities that either play the role of targets for agent activities, or serve as tools to achieve a specific goal: we call these entities artifacts. Artifacts are passive, reactive entities providing services and functionalities to be exploited by agents through a *usage interface*. It is worth noting that artifacts typically realise those behaviours that cannot or do not require to be characterised as goal oriented [24]. Artifacts mediate agents interactions and support coordination in social activities, and embody the portion of the environment that can be designed and controlled to support MAS activities [28, 27]. Moving to a scenario involving cognitive agents, artifacts act as fundamentals bricks for the workspace—where interactions happen—as mediators of interaction and encapsulating coordination functions [23].

Based on this metamodel, we focus here on architectural aspects, considering a recurrent solution when designing self-organizing MAS depicted in Figure 1. In a typical self-organization scenario, agents perturb the environment, and while the environment evolves to compensate the perturbation, agents perception is modified, creating a feedback loop able to sustain the self-organization process. Hence, on the one hand we have agents exploiting artifacts services, that we name *user agents* from now on. Although, because of the enormous repertoire of behaviours that can be exhibited by user agents in different scenarios, we cannot further detail their role in the architecture. On the other

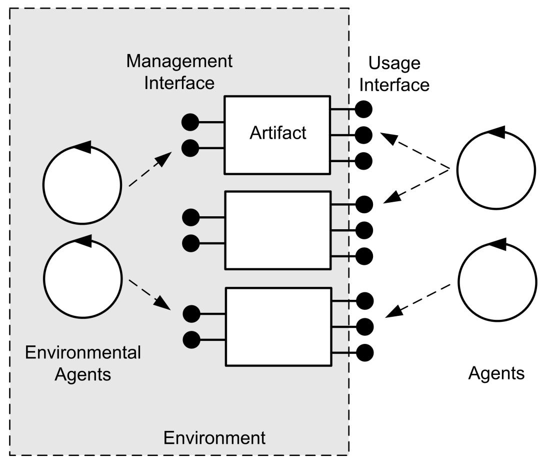

Figure 1. Basic architecture for a MAS featuring environmental agents as artifacts administrators.

hand, artifacts provide basic services in terms of simple local elaboration of agents requests. Because of the passive/reactive attitude of artifacts, they cannot *autonomously* adapt their behaviour to meet the changing requirements of a dynamic and unpredictable environment. Hence, we envision MAS environments as built up of artifacts and *environmental agents*: in particular environmental agents are in charge of those goal-oriented behaviour needed for the management of the artifacts. Specifically, we separate the usage interface from the *management interface*, whose access is restricted only to environmental agents: furthermore environmental agents may exploit artifacts inspectability and malleability. It is worth noting that the characterisation of environmental agent is only in relation with a specific artifact, since an environmental agents for a given artifact may be a normal user agent for another artifact.

The idea of having a managed resource and a resource manager is not original, e.g. see Autonomic Computing initiative [16]. Indeed, adjusting artifact behaviour and status over time is a crucial aspect since it allows for complex behaviour to emerge. In particular, this architecture supports the positive/negative feedback loop together with agents: since self-organization is an active process, it is often the case that artifacts alone cannot close the feedback loop because of the lack of pro-activity, which is instead featured by environmental agents.

In order to provide abstract examples for patterns, we refer to the abstract MAS depicted in Figure 2 featuring user agents, environmental agents and artifacts, and having an arbitrary topology. It is worth noting that in our metamodel there is no explicit notion of topology—although, a topology could be implicitly modelled by granting environmental agents access only to a subset of available artifacts,

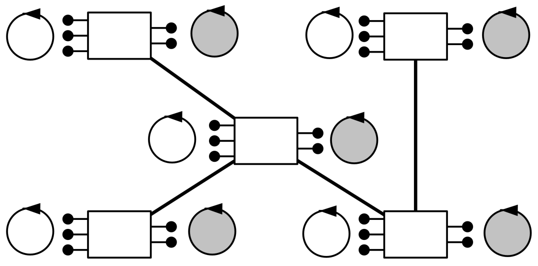

Figure 2. (a) A reference abstract MAS with an arbitrary topology.

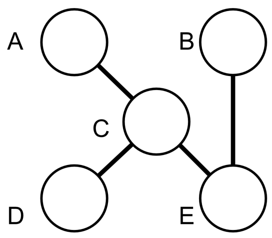

Figure 3. A compact representation equivalent to the MAS depicted in Figure 2.

or explicitly represented by exploiting the A&A notion of *workspace* [25]. For the sake of simplicity, we associated one user and one environmental agent to each artifact: however, a more realistic scenario would involve several user agents and a few of environmental agents for each artifact. To provide readers with an intuitive visual explanation of pattern dynamics, we use the compact representation depicted in Figure 3 where each node represents the coupling of one artifact, one environmental agent and some user agents, hence being equivalent to the MAS in Figure 2. Different shapes are used to represent different kinds of information, physical items and the like, depending on the pattern context and application domain.

2.2 A Reference Pattern Scheme

In this paper we adhere to the scheme for MAS patterns proposed in [17] and briefly summarized in Table 1. It is worth noting that literature proposes several pattern schemes, e.g. see [13, 10]: although, in our opinion, the one described in [17] is a good reference for synthesizing patterns for self-organizing MAS. As a future work, we plan to propose our pattern scheme to better reflect our metamodel and handle specific concerns of self-organizing systems.

Name	The name of the pattern
Aliases	Alternative names
Problem	The problem solved by the pattern
Forces	Trade-offs between metrics dimensions
Entities	Entities participating to the pattern
Dynamics	Entities interactions
Dependencies	Environmental requirements
Example	An abstract example of usage
Implementation	Hints on implementation
Known Uses	Existing applications using the pattern
Consequences	Effects on the overall system design
See Also	References to other patterns

Table 1. The pattern scheme for MAS according to the MASSIVE approach [17].

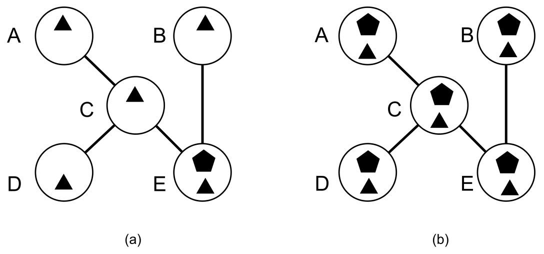

Figure 4. Replication algorithm: (a) an initial state (b) the desired final state.

2.3 Replication Pattern

Natural systems typically feature *replication* mechanisms in order to increase security and robustness. For instance each cell of the human body owns a local copy of the DNA: this allows for recovering minor mutations. Furthermore, replication lowers the time required to access a resource since (i) local copies are *easier* to reach and (ii) copies can be used *concurrently*, hence requiring less queue-time: this aspect is less evident but it actually holds both in biological and social systems. Table 2 summarizes the pattern features, while Figure 4 provides an intuitive example.

2.4 Collective Sort Pattern

Social insects tend to arrange items in their surroundings according to specific criteria, e.g. broods and larvae sorting in ant colonies [6, 7]. This process of collectively grouping items is commonly observed in human societies as well, and serves different purposes, e.g. garbage collection. Also

Name	Replication
Aliases	Duplication, Redundancy
Problem	(1) How can we lower access time to information? (2) In case of attack to artifacts or failure, how can we avoid loosing information?
Forces	Replication ensure better security and faster access at the price of more memory occupation.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts which have to keep a time-stamp for stored information. Environmental agents monitor artifacts for new information, which is eventually sent to neighboring artifacts.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 4 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by an insertion action: either approaches are suitable and choice depend on performance requirements. The forwarding can be implemented in several ways: for example, in the *flooding* approach new information is forwarded to all neighboring nodes but the source.
Known Uses	(1) Cache memories in computer architecture partially replicate RAM content to allow for faster access to data. (2) Redundant Array of Independent Disks (RAID) solutions are used for increasing access speed or recovering purpose, depending on the RAID configuration. (3) In Grid Computing infrastructures [4], local copies of data are often used to reduce network latency.
Consequences	Replication may not work when used in combination with other patterns that spread information across the MAS: in particular replication does not work with Collective Sort (Section 2.4) and Diffusion (Section 2.7).
See Also	-

Table 2. The table summarizes the replication pattern features according to the reference scheme.

in artificial systems collective sort strategies may play an important role: for instance, grouping together related information helps to manage batch processing.

We consider our previous explorations of Collective Sort dynamics in a MAS context [8, 14, 26] in order to synthesize a pattern. From an arbitrary initial state, see Figure 5, the goal of Collective Sort is to group together similar information in the same node, while separating different kinds of information as shown in Figure 5b. Although, this is not always possible: indeed, if we consider a network having two nodes and three kinds of information, two of them are

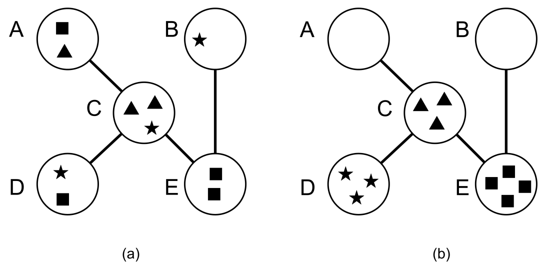

Figure 5. Collective sort (a) an initial state (b) a possible final state.

going to coexist on the same node. Due to random initial situation and asynchronous interactions the whole system can be modelled as stochastic. Hence, it is not generally known a priori where a specific cluster will appear: clusters location is an emergent property of the system [8], which indeed supports robustness and unpredictable environmental conditions. Table 3 summarizes the features of the collective sort pattern.

2.5 Evaporation Pattern

In social insects colonies coordination is often achieved by the use of chemical substances, usually in the form of pheromones: pheromones act as markers for specific activities, e.g. food foraging [6, 7]. Specifically, these substances are regulated by environmental processes called *aggregation*, *diffusion* in space and *evaporation* over time: each process can be captured by a specific pattern, hence, it is analysed separately. This class of mechanisms for indirect coordination mediated by the environment is called *stigmergy* and it has been widely applied in artificial systems engineering [19, 22, 29].

Evaporation is a process observed in everyday life, although with different implications: e.g. from scent intensity it is possible to deduce amount and distance of its source. In the case of insect colonies, marker concentration tracks activities: e.g. absence of pheromone implies no activity or no food source discovered. In ant food foraging [7] when a food source is exhausted, the pheromone tray is no longer reinforced and slowly evaporates.

Evaporation has a counterpart in artificial systems that is related to information obsolescence [23]. Consider a web page listing several news: fresh information is inserted at the top of the page and news *fade* as time passes, which can be translated in visual terms in a movement towards the end of the page. In general, evaporation can be considered a mechanism to reduce information amount, based on a time relevance criterion. As an example, starting from an initial state, see Figure 6a, evaporation removes old information over time and, in absence of new information insertion,

Name	Collective Sort
Aliases	Brood Sorting, Collective Clustering
Problem	MAS environments that does not explicitly impose constraints on information repositories may suffer from the overhead of information discovery.
Forces	Optimal techniques requires more computation while reducing communication costs: on the other hand, heuristics allows for background computation but increase communication costs.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. The artifacts have to provide specific content inspection primitives depending on the implementation. Environmental agents monitor artifacts for new information, and depending on artifacts content may decide to move an information to a neighboring artifacts.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 5 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by an insertion action: either approaches are suitable and choice depend on performance requirements. Moving information requires an aggregated view upon artifacts content, e.g. using counters or spatial entropy measures: in the case this is not feasible or too expensive, content sampling techniques can be used, see [8] for a detailed discussion.
Known Uses	Explorations in robotics for sorting a physical environment [6].
Consequences	Collective Sort may not work when used in combination with other patterns that spread information across the MAS: in particular collective sort does not work with Replication (Section 2.3) and opposes to Diffusion (Section 2.7).
See Also	-

Table 3. The table summarizes the features of the collective sort pattern according to the reference scheme.

eventually erases everything, see Figure 6b.

2.6 Aggregation Pattern

Pheromone deposited in the environment is spontaneously *aggregated*, that is separate quantities of pheromone are perceived as an individual quantity but with greater intensity [6, 7], see Figure 7 for a visual example. Aggregation is a mechanism of reinforcement and is also observable in human social tasks [23, 21]. The ranking mechanism is a typical example: when browsing the Internet someone finds an interesting fact, he/she can leave a (reinforcement) comment that is typically anonymous and

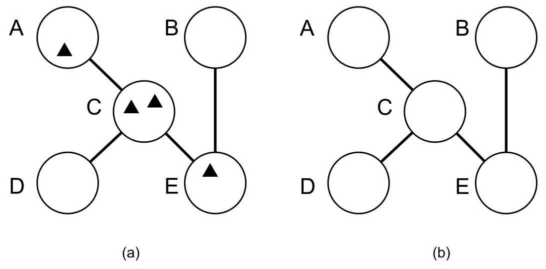

Figure 6. Evaporation (a) an initial state (b) the final state with no reinforcement.

Name	Evaporation
Aliases	None to our knowledge.
Problem	MAS environments can soon become overwhelmed by information deployed by agents.
Forces	Higher evaporation rates release memory, but require more computation: furthermore, evaporated information cannot be recovered!
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. The artifacts assign a timestamp/counter to the received information. Environmental agents erase obsolete information/information whose counter reached zero: eventually, all the information is removed.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 6 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements.
Known Uses	A fundamental element of stigmergy [7] and digital pheromone based application [19, 22, 29].
Consequences	-
See Also	When used in combination with Aggregation (Section 2.6) or Diffusion (Section 2.7), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 4. The table summarizes the features of the evaporation pattern according to the reference scheme.

automatically aggregated with comments of other users. It is then evident that, while evaporation is driven by the environment, aggregation is driven by the user agent. When used in combination with evaporation, aggregation lets the designer close a positive/negative feedback loop, allowing

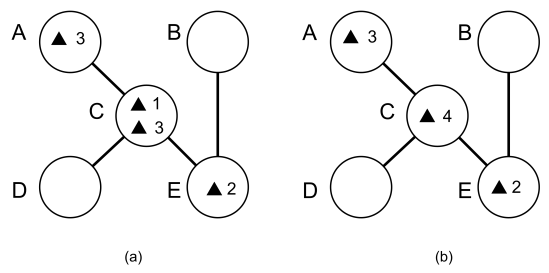

Figure 7. Aggregation (a) an initial state (b) the final state.

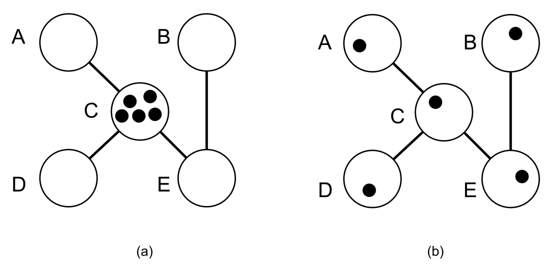

Figure 8. Diffusion dynamics (a) an initial state (b) the desired final state.

for auto-regulated system in self-organization and Autonomic Computing [16] styles.

2.7 Diffusion Pattern

When pheromone is deposited into the environment it spontaneously tends to diffuse to neighboring locations [7]. This process, called *diffusion*, is omnipresent in nature and hence is studied in several fields under different names, e.g. osmosis in chemistry. Starting from an arbitrary state, see Figure 8a, diffusion eventually distribute the information equally across all nodes [3], see Figure 8b.

While aggregation and evaporation processes are often used in combination and act locally, diffusion can be used alone and requires a notion of topology. With respect to replication, in diffusion the initial quantity of information is *preserved* but spatially spread: although, other forms of diffusion may be conceived to produce stable gradients [19].

Name	Aggregation
Aliases	None to our knowledge.
Problem	Large scale MAS suffer from the amount of information deposited by agents, which have to be sifted in order to synthesize macro information.
Forces	Higher aggregation rates provide results closer to the actual environment status, but require more computation.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. Environmental agents look for new information and aggregate it with older information to produce a coherent result.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 7 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements. It is worth noting that aggregation is a very simple task and could be automatically handled by artifacts, when properly programmed: although, it is easier to have separate agents for different functionalities, which can be individually paused or stopped.
Known Uses	A fundamental element of stigmergy [7] and digital pheromone based application [19, 22, 29]. In e-commerce applications customers feedback is usually aggregated, e.g. average ranking, in order to guide other customers.
Consequences	-
See Also	When used in combination with Evaporation (Section 2.5) or Diffusion (Section 2.7), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 5. The table summarizes the features of the aggregation pattern according to the reference scheme.

3 The Role of Patterns in a Methodology for Self-Organizing Systems

Pattern catalogues are an invaluable tool for self-organizing systems designers, since a known pattern is going to display the desired emergent properties when properly used. To support this statement, we briefly describe our approach for designing self-organizing systems, contextualizing the reuse of patterns in the workflow. Due to space constraints, we here deal only with methodological aspects related to patterns: for a more comprehensive discussion readers can refer to [14].

When developing systems featuring self-organizing behaviours and emergent properties, designers have to consider the following issue: *how can we design the individual agent's behaviour in order to let the desired property emerge?* The self-organization community generally accepts that, to deal with the individual dimension, two approaches are available: (i) devising by decomposition an *ad-hoc* strategy that will solve the specific problem; (ii) ob-

Name	Diffusion
Aliases	Plain Diffusion, Osmosis.
Problem	In MAS where agents are only allowed to access local, agents reasoning suffer from the lack of knowledge about neighboring nodes.
Forces	Higher diffusion radius brings information further away from its source, providing a guidance also to distant agents: although, the infrastructure load increases, both in terms of computation and memory occupation. Furthermore, diffused information does not reflect the current status of the environment hence providing false hints.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. A weight is assigned to the information from artifacts or user agents. Environmental agents diffuse information decreasing the weights in local node and correspondingly increasing the weights in neighboring nodes.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 8 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements.
Known Uses	A fundamental element of stigmergy [7] and digital pheromone based application [19, 22, 29]. In e-commerce applications the *see-also* hint is a typical example of information diffusion were the topology is built upon a similarity criterion of products.
Consequences	Diffusion may not work when used in combination with other patterns that spread information across the MAS: in particular diffusion does not work with Replication (Section 2.3) and opposes to Collective Sort (Section 2.4).
See Also	When used in combination with Evaporation (Section 2.5) or Aggregation (Section 2.6), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 6. The table summarizes the features of the diffusion pattern according to the reference scheme.

serving a system that achieves similar results, and trying to reverse-engineer the strategy. While being desirable, the former approach is applicable only to a limited set of simple scenarios: due to the non-linearity in the entities behaviours, the global system dynamics become quite difficult to predict. Instead, the latter approach is commonly regarded as more fruitful: it is possible to identify patterns in natural systems that can be effectively applied to artificial systems [3, 11, 6, 18]. Patterns provide reusable effective solutions to recurrent problems: in particular, patterns not only specify structure but also dynamics, and hence, allow to encode the dynamics responsible for emergent properties. Although, it is quite unlikely to find a pattern that completely fits a given problem: hence, it is a common practice to slightly modify a pattern to fits the designers needs. Due to the complex interactions of self-organizing systems, small modifications may lead to unexpected results: then, can we guarantee that the specific emergent property will actually appear? Providing guarantees about the emergence of the desired global properties is still an open issue: while automatic verification of properties is typically a viable approach to deterministic models, when moving to stochastic models verification becomes more difficult, as existing works in this context are seemingly still immature. Then, it is useful to resort to a different methodology, mixing formal tools and empirical evaluation, so as to support the analysis of the behaviour and qualities of a design.

Our approach consists in evaluating several models with the goal of discovering the one that could provide the quality attributes required for the application at hand: such a task is performed during the early design phases of the software engineering process and allows for evaluating candidate prototypes before committing to the actual design.

In particular, our approach is articulated in three phases:

Modelling to develop an abstract formal specification of the system, based on known patterns;

Simulation to qualitatively and quantitatively investigating the dynamics of the system against the expectation set by the patterns;

Tuning to change model parameters to adjust system behaviour.

Having defined the applications goal and the desired systems dynamics we enter the modelling phase. We start by searching among the known patterns of natural activities a system that exhibits or approximates the target dynamics. This step implies the existence of some sort of natural patterns catalogue, a necessary tool for an engineer of self-organizing systems. Although a full catalogue of this kind does not exist yet, several efforts from different research groups are moving towards that direction: several patterns having strong applications in artificial systems have already been identified and characterised [6, 3, 18]. Hence, as patterns that perfectly match the target system dynamics can hardly be found, it is still feasible to identify some candidate patterns that approximate such dynamics, typically requiring some sort of modifications. As a result, a candidate system design necessarily needs some sort of automatic analysis, such as simulation and automatic verification of properties, which ultimately requires a system description in terms of a formal language. Formal languages promote unambiguity and allow to precisely select the features to be modelled and those to be abstracted away.

So, the deliverable of the modelling phase is a formal specification, which in combination with simulation tools can be used to generate *simulation* traces. To perform stochastic simulations it is necessary to provide a statistical characterisation of the processes described in the specifications. Such characterisation can be easily provided by assigning a *rate of action* defined according to a suitable statistical distribution: typically, the preferred distribution is the exponential one because of the *memoryless property*—to generate new events it is not necessary to know the whole events history, but only the current state.

The main problem in the simulation is to devise valid ranges for the system parameters and action rates: we do not provide any actual strategy for devising those parameters, but they should reflect actual performances in the deployment environment. Then, system specifications along with parameters are used to automatically generate simulations, providing a qualitative and quantitative feedback on the system convergence to the target dynamics.

Unfortunately, since self-organizing systems tend to be very sensitive on initial conditions and working parameters, it is very likely that simulations of initial strategies do not exhibit interesting behaviours. In the *tuning* phase, free parameters are adjusted within the ranges of feasibility and simulations are performed: if none of the parameters produce the desired dynamics the model has to be slightly modified again. If the desired dynamics are previewed then the parameters works as a coarse set for an actual implementation. Although, at the end of the tuning process, we may realise that the devised set of parameters either

- does not satisfy performance expectations;
- has unrealistic values when compared to the execution environment;
- deviates the system from the desired convergence point.

In any of these scenarios we cannot proceed to actual design phase since the system would not behave properly when deployed into the actual environment. Hence, it is required to go back to the modelling phase and evaluate other approaches.

When a model meets the target dynamics, and the parameters lie within the allowed ranges, further elaborations are possible. Indeed, since stochastic simulation traces a partial view on the possible system dynamics, it does not provide actual guarantees about the system behaviour. Instead, providing guarantees about system properties and performance expectations is the goal of formal analysis tools and techniques. For instance, *model checking* uses system specifications and statements about the properties to explore the states space of the target system and test if the statements hold [9]. Short term research goals include evaluating formal analysis techniques for providing actual guarantees about system properties.

4 Conclusion

In the engineering of systems with emergent properties, it is common practice to rely on existing models of natural activities. Despite the existence of many patterns for MAS—see [17, 15, 2, 10] just to name a few—we currently lack a systematic patterns catalogue which could guide the designer of self-organizing systems. A few notable contributions about self-organizing patterns include [3, 11]. In this article, we presented selected patterns synthesized from the self-organization literature, namely, Replication, Collective Sort, Evaporation, Aggregation and Diffusion. Each pattern has been analysed using the scheme proposed in [17] to describe MAS patterns.

In order to clarify the role of patterns within the specific context of self-organizing systems engineering, we provided a brief description our methodological approach based on modelling, simulation and tuning.

Short-term research goals include: using a standard graphical notation for describing patterns; proposing a pattern scheme for self-organizing systems that better reflect our MAS metamodel; evaluating techniques, such as model checking [9], for providing guarantees about the actual emergence of system properties. Medium-term research goals include: integrating our approach in existing agent-oriented software engineering methodologies such as Gaia [30], ADELFE [5], and SODA [20]; applying the full-cycle methodology to case studies, as already done for Collective Sort [8, 26].

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York, 1977.

[2] Y. Aridor and D. B. Lange. Agent design patterns: elements of agent application design. In K. P. Sycara and M. Wooldridge, editors, *2nd International Conference on Autonomous Agents (AGENTS '98)*, pages 108–115, New York, NY, USA, May 1998. ACM Press.

[3] O. Babaoglu, G. Canright, A. Deutsch, G. A. Di Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, , M. J. Roberto Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):26–66, Sept. 2006.

[4] W. Bell, D. Cameron, R. Carvajal-Schiaffino, A. Millar, K. Stockinger, and F. Zini. Evaluation of an economy-based file replication strategy for a data grid. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003, CCGrid 2003*, pages 661–668. IEEE/ACM, May 2003.

[5] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard. ADELFE: A methodology for adaptive multi-agent systems engineering. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNAI*, pages 156–169. Springer, April 2003.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York (NY), US, 1999.

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

[8] M. Casadei, L. Gardelli, and M. Viroli. Simulating emergent properties of coordination in Maude: the collective sorting case. In C. Canal and M. Viroli, editors, *5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06)*, CONCUR 2006, Bonn, Germany, August 2006. University of Málaga, Spain.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, MA, USA, January 2000.

[10] M. Cossentino, L. Sabatucci, and A. Chella. Patterns reuse in the PASSI methodology. In A. Omicini, P. Petta, and J. Pitt, editors, *Engineering Societies in the Agents World IV*, volume 3071 of *LNAI*, pages 294–310. Springer-Verlag, June 2004. 4th International Workshop (ESAW 2003), London, UK, 29–31 October 2003. Revised Selected and Invited Papers.

[11] T. De Wolf and T. Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. In S. Brueckner, S. Hassas, M. Jelasity, and D. Yamins, editors, *Engineering Self-Organising Systems*, volume 4335 of *LNAI*, pages 28–49. Springer, February 2007.

[12] D. Deugo, M. Weiss, and E. Kendall. Coordination of internet agents: Models, technologies, and applications. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Reusable Patterns for Agent Coordination*, chapter 14, pages 347–368. Springer, London, UK, 2001.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Professional Computing. Addison Wesley, December 1995.

[14] L. Gardelli, M. Viroli, M. Casadei, and A. Omicini. Designing self-organising MAS environments: the collective sort case. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for Multi-Agent Systems III*, volume 4389 of *LNAI*, pages 254–271. Springer, February 2007.

[15] E. A. Kendall, P. V. M. Krishna, C. V. Pathak, and C. B. Suresh. Patterns of intelligent and mobile agents. In K. P. Sycara and M. Wooldridge, editors, *2nd International Conference on Autonomous Agents (AGENTS '98)*, pages 92–99, New York, NY, USA, May 1998. ACM Press.

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

[17] J. Lind. Patterns in agent-oriented software engineering. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *Agent-Oriented Software Engineering III*, volume 2585 of *LNCS*, pages 47–58. Springer, 2003. 3rd International Workshop on Agent Oriented Software Engineering (AOSE 2002), Bologna, Italy, July 15, 2002, Revised Papers and Invited Contributions.

[18] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli. Case studies for self-organization in computer science. *Journal of Systems Architecture*, 52(8–9):443–460, August-September 2006.

[19] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *2nd IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2004)*, pages 263–273. IEEE, March 2004.

[20] A. Molesini, A. Omicini, E. Denti, and A. Ricci. SODA: A roadmap to artefacts. In O. Dikenelli, M.-P. Gleizes, and A. Ricci, editors, *Engineering Societies in the Agents World VI*, volume 3963 of *LNAI*, pages 49–62. Springer, June 2006. 6th International Workshop (ESAW 2005), Kuşadası, Aydın, Turkey, 26–28 Oct. 2005. Revised, Selected & Invited Papers.

[21] H. V. D. Parunak. A survey of environments and mechanisms for human-human stigmergy. In D. Weyns, F. Michel, and H. V. D. Parunak, editors, *Environments for Multi-Agent Systems II*, volume 3830 of *LNAI*, pages 163–186. Springer, February 2006.

[22] H. V. D. Parunak, S. A. Brueckner, and J. Sauter. Digital pheromones for coordination of unmanned vehicles. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for Multi-Agent Systems*, volume 3374 of *LNAI*, pages 246–263. Springer, February 2005.

[23] A. Ricci, A. Omicini, M. Viroli, L. Gardelli, and E. Oliva. Cognitive stigmergy: A framework based on agents and artifacts. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for Multi-Agent Systems III*, volume 4389 of *LNAI*. Springer, February 2007.

[24] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In R. P. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems*, volume 3862 of *LNAI*, pages 206–221. Springer, Mar. 2006. 3rd International Workshop (PROMAS 2005), AAMAS 2005, Utrecht, The Netherlands, 26 July 2005. Revised and Invited Papers.

[25] A. Ricci, M. Viroli, and A. Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for MultiAgent Systems*, volume 4389 of *LNAI*, pages 67–86. Springer, Feb. 2007. 3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers.

[26] M. Viroli, M. Casadei, and L. Gardelli. A self-organising solution to the collective sort problem for distributed tuple spaces. In *22th ACM Symposium on Applied Computing (SAC 2007)*, pages 354–359, Seoul, Korea, March 2007.

ACM. Special Track on Coordination Models and Languages.

[27] 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, Feb. 2007. Special Issue on Environments for Multi-agent Systems.

[28] D. Weyns, A. Omicini, and J. Odell. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, Feb. 2007. Special Issue on Environments for Multi-agent Systems.

[29] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 67–74. ACM, July 2005.

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