Design Patterns for Self-organising Systems

Luca Gardelli, Mirko Viroli, and 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 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-organising systems, we analyse a selection of design patterns devised from the self-organisation literature. Starting from our reference MAS metamodel, we propose a pattern scheme that reflects the peculiarities of self-organising systems. Then, we provide a complete characterisation of each pattern, with particular attention to the problem description, the solution with respect to our metamodel, the natural systems which have inspired the pattern and known applications.

1 Introduction

Self-organisation is a very compelling approach to the engineering of complex systems because of the many interesting properties, including adaptivity and robustness. Although, forward engineering of self-organising systems, i.e. finding the individual behaviour to meet the desired global dynamics, rapidly becomes unfeasible as the complexity of the system increases. Hence, it is becoming common practice to exploit existing models of natural systems, particularly social insects: these models provide a characterisation of global dynamics with respect to individual actions and environmental parameters. To ease this process, we promote the use and development – since few works exist about this topic [1,2] – of design patterns for self-organising systems to establish a mapping between artificial systems problems and natural systems solutions. First introduced in 1977 by Alexander in architecture [3], the concept of *design pattern* later gained wide consensus in computer science with the object-oriented paradigm [4]. 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. Multiagent system (MAS) researchers synthesised patterns for the agent paradigm, providing solutions related to resource access, mobility and basic social skills [5,6,7]. 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 language. Specifically, in the case of self-organising systems, patterns play a key role in driving the designer choices among the chaotic behaviours displayed by complex systems.

To promote the acceptance of new patterns, as well as to reduce ambiguity, it is necessary to frame a pattern with respect to a shared scheme, ensuring a good degree of coherence of the whole pattern catalogue. For example, patterns for the object-oriented paradigm are described according to the scheme provided in [4]: although, as pointed out in [8,7], since the agent paradigm cannot be effectively characterised using only object-oriented abstractions, patterns for MAS should be described using specific schemata. In particular, a candidate scheme should reflect the peculiarities of the target MAS metamodel: to this purpose, several pattern classification and schemata have been proposed [8,9]. However, pattern schemata for MAS, like the one in [8], do not adequately capture the peculiarities of self-organising systems, namely, which are the forces responsible for the feedback loop, and which notion of locality/topology is needed. Furthermore, to our knowledge, no specific pattern scheme has been proposed to deal with the previous aspects, and the few existing works rely on existing MAS schemata.

The contribution of this article is twofold: we extend current pattern schemata to better represent self-organising MAS, and characterise some patterns with respect to our MAS metamodel. The remainder of the article is structured as follows: Section 2 describes our MAS metamodel based on agents and environmental resources, i.e. artifacts. Section 3 describes the reference pattern scheme, then Section 4 analyses each pattern, namely, Collective Sort, Evaporation, Aggregation and Diffusion. Section 5 concludes by providing final remarks and listing future research directions.

2 Our Reference MAS Metamodel

When modelling natural systems and developing artificial ones, the MAS paradigm is the best choice since it provides the suitable abstractions for modelling and relating the entities. Although being recognised as unique, the MAS paradigm is captured from different perspectives in different metamodels emphasising specific features. We adopt the agents & artifacts metamodel (A&A), where a MAS is modelled in terms of two fundamental abstractions: *agents* and *artifacts* [10]. Agents are autonomous pro-active entities encapsulating control, driven by their internal goal/task. When developing a MAS, sometimes entities do not require neither autonomy nor pro-activity to be correctly characterised. 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 [10]. 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 [11,12].

Based on this metamodel, we recognise a recurrent solution when designing self-organising MAS: the solution depicted in Figure 1 is an architectural pattern. In a typical self-organisation scenario, agents perturb the environment, and while the environment evolves to compensate the perturbation, agents perception is affected, creating a feedback loop able to sustain the self-organisation

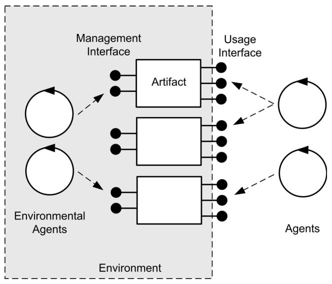

Fig. 1. Architectural pattern featuring environmental agents as artifact administrators

process. Hence, on one side 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 side artifacts provide basic services in terms of simple local elaboration of agents requests. Since 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 built upon 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* – in an Autonomic Computing style [13] – whose access is restricted only to environmental agents: furthermore environmental agents may exploit artifacts inspectability and malleability.

3 A Reference Pattern Scheme

The literature provides several pattern schemata, e.g. [4,9,8]. In a previous work [14], we analysed patterns for self-organising system relying on the scheme for MAS described in [8]: although, we recognise the failure to capture essential self-organisation aspects, namely forces involved in the feedback loop and topology notion. The pattern scheme we propose extends the one described in [8] and is summarised in Table 1, where the novel items are emphasised. Particularly relevant to this work are the *feedback loop* and *locality* elements:

Feedback loop. Describes the processes or actions involved in the establishment of a feedback loop, i.e. the actions providing positive and negative feedback. For example, in a digital pheromone infrastructure [15], the positive feedback consists in the agent depositing pheromones, while the environment provides the negative feedback in the form of pheromone evaporation.

Locality. Requirements in terms of spatial topology or action-perception ranges: if the environment has a notion of continuous space, perception range is specified as a float value; if the environment has a graph topology, ranges are specified as the number of hops.

Table 1. An extension to the pattern scheme described in [8]

Name	The name of the pattern
Aliases	Alternative names
Problem	The problem solved by the pattern
Forces	Trade-offs in system dynamics
Entities	Entities participating to the pattern
Dynamics	Entities interactions
Feedback Loop	Interactions responsible for the feedback loop
Locality	Describe the type of locality required
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

4 Patterns of Self-organising Systems

4.1 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 [16,17]. This process of collectively grouping items is commonly observed in human societies as well, and serves different purposes, e.g. garbage collection. Also 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 exploration of Collective Sort dynamics in a MAS context [18,19] in order to synthesise a pattern. From an arbitrary initial state, see Figure 2, 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 2b. Although, this is not always possible: indeed, if we consider a network having two nodes and three kinds of information, two of them are 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 properties of the system [18], which indeed supports robustness and unpredictable environmental conditions. Table 2 summarises the features of the collective sort pattern.

4.2 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

Table 2. A summary of the features of the collective sort pattern according to the reference scheme

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.
Feedback Loop	Positive feedback is determined by environmental agents moving items to the appropriate cluster, while negative feedback happens when an item is misplaced.
Locality	Either continuous and discrete topology are suitable. Larger perception range improve strategy efficiency, but perception of immediate neighborhood is sufficient, but requires memory of items encountered.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 2 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 [18] for a detailed discussion.
Known Uses	Explorations in robotics for sorting a physical environment [16].
Consequences	Collective Sort may not work when used in combination with other patterns that spread information across the MAS: in particular collective sort opposes to Diffusion (Section 4.4).
See Also	-

specific activities, e.g. food foraging [16,17]. 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 the engineering of artificial systems [20,15,21].

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 discovered food source. In ant food foraging [17] when a food source is exhausted, the pheromone trail is no longer reinforced and slowly evaporates.

Evaporation has a counterpart in artificial systems that is related to information obsolescence [22,23]. 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 3a – evaporation removes old information over time and, in absence of new information insertion, eventually erases everything—see Figure 3b.

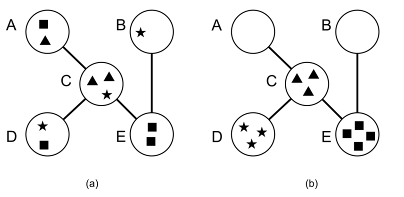

Fig. 2. Collective sort (a) an initial state (b) the final state

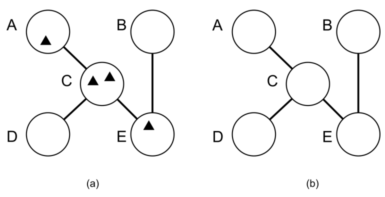

Fig. 3. Evaporation (a) an initial state (b) the final state with no reinforcement

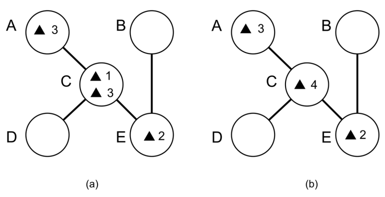

Fig. 4. Aggregation (a) an initial state (b) the final state

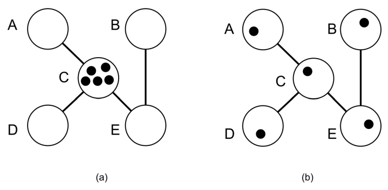

Fig. 5. Diffusion dynamics (a) an initial state (b) the desired final state

4.3 Aggregation Pattern

Pheromone deposited in the environment is spontaneously *aggregated*, i.e. separate quantities of pheromone are perceived as an individual quantity but with greater intensity [16,17], see Figure 4 for a visual example. Aggregation is a mechanism of reinforcement and is also observable in human social tasks [22]. 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 anonymously and 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 for auto-regulated system in self-organisation and Autonomic Computing [13] styles.

4.4 Diffusion Pattern

When pheromone is deposited into the environment it spontaneously tends to diffuse in neighboring locations [17]. 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 5a, diffusion eventually distribute the information equally across all nodes [1], see Figure 5b.

While aggregation and evaporation processes are often used in combination and act locally, diffusion can be used alone and requires a notion of topology. Furthermore, in diffusion the initial quantity of information is *conserved* but

Table 3. A summary of the features of the evaporation pattern according to the reference scheme

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.
Feedback Loop	User agents deposit items in the environment while environmental agent evaporate them.
Locality	Perceptions and actions happens only locally. Either continuous and discrete topology are suitable.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 3 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 [17] and digital pheromone based application [20,15,21].
Consequences	-
See Also	When used in combination with Aggregation (Section 4.3) or Diffusion (Section 4.4), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 4. A summary of the features of the aggregation pattern according to the reference scheme

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 synthesise 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.
Feedback Loop	User agents deposit items in the environment while environmental agent synthesise an aggregated info.
Locality	Perceptions and actions happens only locally. Either continuous and discrete topology are suitable.
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 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 [17] and digital pheromone based application [20,15,21]. 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 4.2) or Diffusion (Section 4.4), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 5. A summary of the features of the diffusion pattern according to the reference scheme

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.
Feedback Loop	User agents deposit items in the environment while environmental agent scatter them to neighboring locations.
Locality	User agents perceptions and actions happens only locally, while environmental agents need to perceive and act at least at one hop of distance. Either continuous and discrete topology are suitable.
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 a specific event: either approaches are suitable and choice depend on performance requirements.
Known Uses	A fundamental element of stigmergy [17] and digital pheromone based application [20,15,21]. 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 opposes to Collective Sort (Section 4.1).
See Also	When used in combination with Evaporation (Section 4.2) or Aggregation (Section 4.3), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

spatially spread: although, other forms of diffusion may be conceived to produce stable gradients [20].

5 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 [8,5,6,9] just to name a few – we currently lack a systematic patterns catalogue which could guide the designer of self-organising systems: few notable exceptions include [24,1,2]. In this article, we provide an extension to the pattern scheme described in [8] to better reflect the peculiarities of self-organising systems. Furthermore, we describe a few patterns devised from the self-organisation literature, namely, Collective Sort, Evaporation, Aggregation and Diffusion: each pattern has been analysed with respect to the proposed pattern scheme. Minor contributions of this article include

– recognising the need of a pattern catalogue and identifying the special role of patterns in the engineering of artificial self-organising systems;

– recognising the qualitative differences between patterns for self-organising systems with respect to MAS and object-oriented patterns;
– show how the composition of patterns is a challenging task: the evaluation of a specific pattern requires the knowledge about the interplay of the dynamics.

Future works include

– identifying and using the appropriate graphical notation for describing patterns;
– extending the pattern catalogue and validating the proposed pattern scheme.

References

1. Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G.A., Ducatelle, F., Gambardella, L.M., Ganguly, N., Roberto Montemanni, M.J., Montresor, A., Urnes, T.: Design patterns from biology for distributed computing. ACM Transactions on Autonomous and Adaptive Systems 1(1), 26–66 (2006)
2. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 28–49. Springer, Heidelberg (2007)
3. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Professional Computing. Addison-Wesley, Reading (1995)
5. Kendall, E.A., Krishna, P.V.M., Pathak, C.V., Suresh, C.B.: Patterns of intelligent and mobile agents. In: Sycara, K.P., Wooldridge, M. (eds.) AGENTS '98. 2nd International Conference on Autonomous Agents, pp. 92–99. ACM Press, New York (1998)
6. Aridor, Y., Lange, D.B.: Agent design patterns: elements of agent application design. In: Sycara, K.P., Wooldridge, M. (eds.) AGENTS '98. 2nd International Conference on Autonomous Agents, pp. 108–115. ACM Press, New York (1998)
7. Deugo, D., Weiss, M., Kendall, E.: Reusable Patterns for Agent Coordination. In: Coordination of Internet Agents: Models, Technologies, and Applications, pp. 347–368. Springer, Heidelberg (2001)
8. Lind, J.: Patterns in agent-oriented software engineering. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 47–58. Springer, Heidelberg (2003)
9. Cossentino, M., Sabatucci, L., Chella, A.: Patterns reuse in the PASSI methodology. In: Omicini, A., Petta, P., Pitt, J. (eds.) ESAW 2003. LNCS (LNAI), vol. 3071, pp. 294–310. Springer, Heidelberg (2004)
10. Ricci, A., Viroli, M., Omicini, A.: Programming MAS with artifacts. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Programming Multi-Agent Systems. LNCS (LNAI), vol. 3862, pp. 206–221. Springer, Heidelberg (2006)
11. Weyns, D., Omicini, A., Odell, J.: Environment as a first-class abstraction in multi-agent systems. Autonomous Agents and Multi-Agent Systems 14(1), 5–30 (2007)

12. Viroli, M., Holvoet, T., Ricci, A., Shelfthout, K., Zambonelli, F.: Infrastructures for the environment of multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1), 49–60 (2007)
13. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
14. Gardelli, L., Viroli, M., Omicini, A.: Design patterns for self-organizing multiagent systems. In: De Wolf, T., Saffre, F., Anthony, R. (eds.) 2nd International Workshop on Engineering Emergence in Decentralised Autonomic Systems (EEDAS 2007), ICAC 2007, University of Greenwich, London, UK, pp. 62–71. CMS Press, Jacksonville, FL, USA (2007)
15. Parunak, H.V.D., Brueckner, S.A., Sauter, J.: Digital pheromones for coordination of unmanned vehicles. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 246–263. Springer, Heidelberg (2005)
16. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems, Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York, US (1999)
17. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, USA (2001)
18. Casadei, M., Gardelli, L., Viroli, M.: Simulating emergent properties of coordination in Maude: the collective sorting case. Electronic Notes in Theoretical Computer Sciences 175(2), 59–80 (2007) (5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA 2006) (2006)
19. Gardelli, L., Viroli, M., Casadei, M., Omicini, A.: Designing self-organising MAS environments: the collective sort case. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) Environments for Multi-Agent Systems III. LNCS (LNAI), vol. 4389, pp. 254–271. Springer, Heidelberg (2007)
20. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the TOTA middleware. In: PerCom 2004. 2nd IEEE Annual Conference on Pervasive Computing and Communications, pp. 263–273. IEEE, Los Alamitos (2004)
21. Weyns, D., Schelfthout, K., Holvoet, T., Lefever, T.: Decentralized control of E'GV transportation systems. In: AAMAS 2005. 4th International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, July 25-29, 2005, pp. 67–74. ACM, New York (2005)
22. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive stigmergy: A framework based on agents and artifacts. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 124–140. Springer, Heidelberg (2007)
23. Parunak, H.V.D.: A survey of environments and mechanisms for human-human stigmergy. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 163–186. Springer, Heidelberg (2006)
24. Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case studies for self-organization in computer science. Journal of Systems Architecture 52(8–9), 443–460 (2006)