MAS Development: Reusing through Agent Oriented Design Patterns

Sylvain SAUVAGE
GREYC — CNRS UMR 6072, université de Caen
Caen, 14032, France

ABSTRACT

Capitalising and diffusing experience about multiagent systems are two key mechanisms the classical approach of methods and tools can't address. Our hypothesis is that, among available techniques that collect and formalise experience, design patterns are the most able technique allowing to express the agent concepts and to adapt itself to the various MAS developing problems.

In this paper, we present several agent oriented patterns in order to demonstrate the feasibility of helping MAS analysis and design through design patterns. Our agent patterns cover all the development stages, from analysis to implementation, including re-engineering (through antipatterns).

Keywords: MultiAgent Systems, Software Engineering, Design Patterns

1. INTRODUCTION

Analysis and design are becoming an important subject of research in multiagent systems (MAS). Now MAS are more and more used, the need for tools and methods allowing the quick and reliable realisation of MAS is clearly appearing. This need appears in the MAS community itself but it's also a necessity to empower a larger community to use the agent paradigm.

Present attempts of the community focus on MAS design methods [1]. Alas, only experts can use those methods, and their associated tools, to develop a multiagent system and to understand the involved concepts. In fact, MAS paradigm development is restricted by the gap between the apparent ease to seize the basic concepts (as agent, role, organisation, or interaction) and the difficulty to create a MAS which solves or helps to solve a real world problem. Then, to help the MAS designer, we seek a new formalisation which will enable reusing and understanding of MAS techniques and concepts.

As agents are often implemented by the way of object oriented programming, the idea to use object-oriented analysis and design methods is appealing. Unfortunately these object methods are not applicable to develop complex MAS, simply because objects and agents do not pertain to the same logical and conceptual levels – agents integrate other concepts, such as autonomy, finality, interaction, and the fact they form a multiagent system. Nevertheless, we can draw our inspiration from object methods and from their development to create methods for MAS.

Since reuse is a main goal of software engineering, and in order to find the way to capitalise and diffuse MAS development experience, we analysed reusing techniques. The results of our analysis are shown on Table 1. In order to compare these reusing techniques, we distinguish three targets of reusing: structures, processes, and concepts. Each of these targets can be viewed at different abstraction levels. The structures target addresses the code, the organisation of the code and the architecture of the concepts. The processes target gathers both code processes (algorithms) and design processes (methods). The concepts target is about reusing the models, principally through classes (as with inheritance).

As shown on Table 1, among the reusing techniques, the design patterns [2] cover all three targets. Moreover, even if the coding part is a little less covered than with components, the method (*i.e.*, the path from the problem to its solution) is much more covered.

Table 1. Reuse in reusing techniques

	Targets of reusing		
Reusing techniques	Structures	Processes	Concepts
Duplication	code	—	—
Functions library	code	algorithms	—
Classes library	code	comportment	classes
Components	code, architecture	comportment	classes
Frameworks	code, architecture	algorithms	(classes)
Design methods	—	method!	models
Design patterns	(code), architecture	algorithms, method	classes, models

These reasons show that design patterns are strongly appropriate to solve the MAS design problems. In fact, design patterns are the perfect means to spread the concepts, the models, and the techniques used in the MAS design and in their implementation.

In the following sections, in order to demonstrate the feasibility of helping MAS analysis and design through design patterns, we will expose shortened versions of four agent oriented design patterns. These patterns are the most representative of the application of design patterns to MAS development.

2. AGENT ORIENTED DESIGN PATTERNS

Definition 1. *A* Pattern *is a partial solution to a contextual recurrent problem.*

Describing design patterns consists in gathering several applications that use similar solutions in similar contexts and then abstracting these solutions in one technique. The main issues are: to detect the similarity of applications, to abstract and unify the technique employed in these applications, to formalise this technique, and to analyse its usage (where, when, what, how, and why).

From this process we can clearly see several main advantages of design patterns: study and analysis of former works, formalisation and enhancement of a technique that was not clearly exposed, and the fact they can be used with other software engineering tools (methods, frameworks, etc.).

A design pattern is described as a tuple. The fields of this tuple differ among authors. The most important point is that they must answer to all the questions one can have about the proposed solution (where, when, what, how, and why). In order to formalise our agent oriented patterns, we use the fields [3] uses. These fields are the following ones: (1) a *name*, (2) a *synopsis* of the described technique, (3) the contextual *forces* of application (where and when), (4) *source examples* of usage (the source applications in which the pattern shows up), (5) the proposed *solution* (what), (6) some *implementation* issues (how), (7) an *examination* of advantages and disadvantages of the solution (why), and (8) *associated patterns*, as design patterns are tied together (by cooperation, usage, delegation, or conflict).

Since the patterns we discovered and designed have different application levels and origins, we class them in four categories: metapatterns, metaphoric patterns, architectural patterns and antipatterns (see Figure 1). In the following sections, we will explain these categories and give shortened versions of selected associated patterns.

3. FIRST CATEGORY OF AGENT PATTERNS: METAPATTERNS

Metapatterns are so called because they are patterns of higher conceptual level than other patterns. Their scope is wider, they address problems at all stages of the design process, from analysis to implementation. Moreover, they also cover children patterns.

Definition 2. *A* child pattern *is a more specific pattern that uses and details the concepts and the solution proposed by the associated metapattern.*

For the formalisation of metapatterns, as these patterns can be applied in several cases and at all the stages of the development process, the *forces (3)* field has less meaning so it is replaced by the *concepts (3)* field, which explains

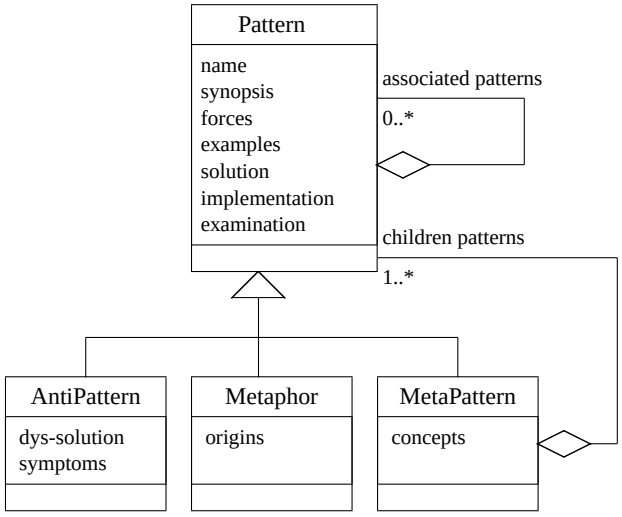

Figure 1. Agent design patterns categories

the concepts the metapattern uses and defines. For the same reasons, the *implementation (6)* field is omitted.

We propose two metapatterns: the *Organisation schemes* pattern and the *Protocols* pattern.

Pattern I. Organisation schemes
allow to integrate real world knowledge into the description of the MAS through high level concepts. They also allow their manipulation by the agents themselves. Here is the description of this pattern:

1. Name: *Organisation schemes.*

2. Synopsis. This pattern explains the concepts associated with organisation and describes their numerous uses.

3. Concepts

- *Agent:* an autonomous entity.
- *Role:* a function an *agent* can take in an *organisation.* This function can have a short life (undertook and shortly after abandoned) or a long life (the function survives the agent as several agents follow each other at its charge). Each role defines associated behaviours, interactions and relations.
- *Organisation:* a structure grouping several agents which undertake roles in the *organisation scheme* by which this *organisation* is modelled.
- *Organisation scheme:* an abstract structure grouping role descriptions. It's a class of organisations.

4. Source examples we used to abstract and describe the usage of *organisation schemes* are various. The concepts are used – but not always well explained – in models, as in Agent-Group-Role [4] or for Zeus [5], in design methods, as in Gaia [6] or Soda [7], and in frameworks, as in MadKit [8] or Durand's framework [9].

5. The Solution this pattern proposes for the analysis of a MAS is to use organisation schemes as a lecture grid: organisations in the MAS are discovered and described by the comparison with a catalogue of organisation schemes ([10] could be a sketch for such a catalogue).

In MAS design, this pattern proposes to integrate and describe roles and organisations in the UML class diagrams of the designed MAS. This integration allows to virtualise the services the roles represent: relations between roles only depend on the roles, not on the agents undertaking the roles.

In MAS implementation, this pattern proposes to embody roles and organisations as objects, so that agents could be able to use and control them.

7. The Examination of this pattern shows us various advantages. Among these advantages is the ability to integrate, through high level concepts, *real world* knowledge into the description of the MAS.

Another advantage is the ability, by dividing the system into various organisations, to divide and conquer the analysis and design of the system.

8. Associated patterns to this pattern are, of course, its children patterns: patterns about specific organisation structures and models that explain specific roles, their associated behaviours and interactions.

Pattern II. Protocols
allow to describe a MAS from the interaction point of view, by the way of messages, roles, and protocols.

1. Name: *Protocols.*

2. Synopsis. This pattern covers the concepts associated to interaction and communication protocols.

3. Concepts

- *Agent:* an autonomous entity.
- *Message:* an information object transmitted by an *agent* to another one. This concept also addresses speech acts, [11].
- *Interaction:* communication, perturbation or influence link between agents. In a communication protocol, it's a *conversation.*
- *Role:* a function an agent undertake in an interaction. Each role defines the messages the agent can send.
- *Protocol:* a set of rules allowing two or more agents to coordinate. A protocol defines the order and the type of messages and actions.

4. Source examples we used to described *Protocols* are numerous in analysis and design: models and methods use them to describe MAS. For instance the methods Soda [12], Mase [13], the AUML notation [14] or [15]'s UML notation.

5. The Solution this pattern proposes is to find messages, conversations and interactions in the system and to abstract them into protocols. Another complementary way of using protocols is to use a catalogue of protocols as a grid to analyse the system (such a catalogue can be written from the list of protocols the FIPA proposes: `http://www.fipa.org`).

7. Examination. Identifying protocols at an early stage allows a better identification of the roles and the interactions of the system, and of the messages agents can send.

In another hand, as design patterns already do, protocols form a common vocabulary that enables a better analysis and a better description of the system.

8. Associated patterns to this one are its children patterns: patterns explaining the application context and the constraints resolved by a particular protocol. *Organisation schemes* is associated to this pattern as they both share the main concept of *role. Influences* and *Marks* are also associated patterns: communications between agents are a form of influences and marks are a form of messages.

4. SECOND CATEGORY OF AGENT PATTERNS: METAPHORIC PATTERNS

Metaphoric patterns are widely used is MAS. They describe the use of a solution that is inspired from a discipline which, at first glance, seems to be totally exotic to multi-agent systems and to their design. The technique or model that inspires the solution of the metaphoric pattern is explained in a new field, *origins (3')*, which is added to the formalisation.

Pattern III. Marks
is a pattern based on the metaphor of pheromones, a model of communication through the environment.

1. Name: *Marks.*

2. Synopsis. *Marks* are deposited in the environment. The environment is used as a situated, shared, supplemental memory.

3. Forces

- Agents have limited memory capabilities.
- Agents have limited communication capacities.
- Agents are situated: there are constraints affecting their positions, their moves.
- There are constraints affecting the resources agents can use: limited speed, energetic autonomy, and time.
- Informations that need to be kept or shared have spatial characteristic (they are only locally relevant).

3'. The Origins of this metaphor are biological. The principle is especially used by insect species. As an example, an ant deposits slight quantities of a chemical matter (called *pheromone*) which enables it to make a track of its path (external memory) and to point out this path to its congeners (communication and recruitment medium). Pheromones directly induce a specific behaviour for the individual perceiving them – it's a *reaction*, as in the *reactive–cognitive* opposition of the agent literature. Pheromones act in the same way as the nervous system chemical transmitters.

4. Source examples. Communication by marks and especially by pheromones has been studied and used in various works, from ant-hill simulations, as Manta [16] or [17], to problem solving, as image recognition [18].

5. Solution. The agent deposits marks in the environment. These marks are low granularity objects and have a minimal size. They have some strength that allows the agent to *sense* their presence at a known distance. Once they are deposited, the environment takes care of their mutation: evaporation, dissemination/propagation, and perception. These marks are perceived in a limited zone and in a limited way (mainly related to associated agent sensors).

The *marks* allow the agents to communicate without message exchange: depositing some quantity of a known *mark* is a modification of the environment every agent perceives.

6. Implementation. If the goal is to have agents living in the real world (*e.g.* robots), therefore marks are slight quantities of a product that agents are able easily and distinctly to perceive. It's unlikely that a chemical matter be used, but one can use marks which will be easy to distinct and which will be persistent to their emission.

If the agents live in a simulated world, therefore the environment has to be pro-active to manage the marks' lifecycle: evaporation, dissemination (due to the wind, other agents' moves), masking (by other marks), disappearance (even removal).

7. Examination. There also are problems due to the use of this pattern. Its implementation in a real world (that is, with robots) is not so easy: there are some troubles to find the good *product* – that implies troubles with the sensors, with their sensitivity and accuracy.

Concerning the difficulties of the simulation, the required memory and computational resources can't be neglected. This pattern also gives a great role to the environment.

However, this pattern has plenty of advantages, in addition to resolving the constraints exposed in the field *Forces*. Marks are simple messages – as much by their form than by their handling – and the fact that they are deposited allows integrating a locality notion to the information they carry. Moreover, this integration is done in an indirect way for the agent: he does not need a coordinate system, marks are deposited where they have a signification.

8. Associated patterns to this one are mainly *Influences* and *Protocols:* the attraction/repulsion effect of marks is mainly an influence and marks are somewhat a type of message.

Pattern IV. Influences

allow avoiding conflicts caused by simultaneous actions. They also allow handling global forces, inertia, or reactions.

1. Name: *Influences.*

2. Synopsis. *Influences* allow separating causes and effects of primitive actions to overcome simultaneous actions conflicts.

3. Forces

- There are global influences and forces.
- There may be simultaneous actions.
- An agent can perturb another one without going through the second agent's cognitive mechanisms (no asynchronous message, physical and social levels are distinct).

3'. The Origins of this metaphor are physics: influences are forged from physical forces. Objects (and though agents) don't modify each other, instead, they create forces that, once combined, have an action.

4. Source examples. The main source example is [19] which was the first to explain the model of influences. [20] incorporate *influences* in their formal logic model. [21] and the Moca framework [22] implement *influences.*

5. The Solution proposed consists in distinguishing the action an agent apply to another agent and the action this other agent undergoes.

6. The Implementation of this solution can use the algorithm defined in [21]. It can also use gradients and a discrete space algorithm to compute them: the wave gradient algorithm, [3].

7. Examination. Concurrent actions conflicts are frequent and difficult to handle. Moreover, it is often convenient to have a mechanism to handle global influences. *Influences* are also suitable to handle inertia, noise or friction.

The main disadvantage of this pattern is that the combination of influences and the calculus of their effects have to be done by a mechanism that is outside the agents. It then leads to a global action controller, though being in contradiction to the principle of MAS: control distribution.

8. Associated patterns. The *Marks* pattern can benefit from this pattern as a mark is a source of attraction/repulsion. The three antipatterns we expose below *(Discretisation, Iniquity and Physical entity)* can be applied to implement this one, as *Command* and *Composite* can be, [2, pp. 86, 233].

5. ARCHITECTURAL PATTERNS

Architectural patterns describe the internal architecture of an agent. As an agent often has many tasks to accomplish and as these tasks require skills of different cognition levels – some are reactive tasks, some are deliberations –, the structure of an agent is often decomposed into several modules. The architectural patterns show different ways to discover and arrange these modules and their interactions.

For now, we can see four main agent architecture types: *BDI architecture* (which embodies the BDI model), *Vertical architecture* (layering agent's internal modules in knowledge levels), *Horizontal architecture* (parallelising the agent's modules), and *Recursive architecture* (recursively seeing an agent as a MAS). For a more complete description, see [3].

6. ANTIPATTERNS

Antipatterns, [23], are somewhat special patterns as they don't explain how to design a system but how to redesign and correct common mistakes. These mistakes are explained in the field we called *Dys-Solution.* We see at least three antipatterns: *Iniquity* (when resources are not equally managed because of inappropriately simulated parallel calculus), *Discretisation* (partial loss of information, especially with numerical data), and *Physical entity* (lack of separation between the agent's physical and rational aspects). For a more complete description and examples of usage, see [3].

7. RELATED WORKS

Different works about agent patterns exist. Some are short or more object-like, [24–27]. Some are interesting, as Aridor's, Kendall's, Deugo's, or Aarsten's [28–32], but it's a consideration not to limit the MAS application of the pattern technique to the simple use of patterns in a particular domain that MAS are. Indeed, some papers submit patterns presenting agents and agent techniques in an object-oriented way. In other words, they present them as object techniques, leaving out the fact that agents are much more than objects. In fact, agents integrate other concepts, as autonomy, finality, interaction, and the fact they form a multiagent system.

8. CONCLUSIONS

Agent design methods are only really usable by experts: they do not explain the involved concepts, they lack details, and they do not really reuse previous works. Fortunately, design patterns are a really great technique to teach and reuse concepts and techniques. In fact, design patterns are a unique way to transmit experience, particularly to transmit development experience.

Finding a pattern seems easy, and it can really be easy to realise that similar solutions occur in similar contexts to solve similar problems in different real world cases. But, describing a pattern demands a meticulous analysis of the solution, of its mechanism, of its advantages and disadvantages, and of its implentations.

In this paper, we have presented shortened versions of four agent oriented patterns among the eleven we already discovered and fully described. One remark to be done is that they are of a higher abstraction level than object oriented design patterns usually are, surely because agent concepts are of a higher abstraction level than objects.

That's a proof the design patterns technique can be employed to present, explain, and spread the agent concepts. We think that patterns can help to develop MAS, as much in analysis, design or implementation as in teaching and spreading of the agent paradigm. Presenting as patterns the models and concepts of the MAS paradigm, as well as the techniques used to implement them, would enable us to structure, spread and constructively extend the knowledge of the agent paradigm we now have. Indeed, more than their *uniform* structure, patterns allow to understand more easily the concepts – for they emerge from experience and widely used examples, but also for they integrate theory to examples and they explain the conceptual reasons behind implementation techniques.

For now, our main goal is to present our agent patterns, as in [33], submit them to discussion and enhancement, and show their use through the development of various multiagent systems in real world cases.

Another goal is to use the concepts of the *Organisation schemes* metapattern as a formalism to describe all the agent patterns we discovered and already described in a formalism closer to natural language.

9. REFERENCES

[1] C. A. Iglesias, M. Garijo, and J. C. González. "A Survey of Agent-Oriented Methodologies." In J. P. Müller, M. P. Singh, and A. S. Rao (eds), *Intelligent Agents V, ATAL'98*, LNAI. Springer-Verlag, 1998.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[3] S. Sauvage. *Conception de systèmes multi-agents : un thésaurus de motifs orientés agent*. Thèse de doctorat (informatique), université de Caen (France), université de Neuchâtel (Switzerland), Oct. 2003.

[4] O. Gutknecht and J. Ferber. "Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents." In J.-P. Barthès, V. Chevrier, and C. Brassac (eds). *JFIADSMA'98*, pp. 267–280. Hermès, Nov. 1998.

[5] E. Kendall. "Agent Roles and Role Models: New Abstractions for Intelligent Agent Systems Analysis and Design." In *AIP'98*, 1998.

[6] M. Wooldridge, N. Jennings, and D. Kinny. "The

Gaia Methodology for Agent-Oriented Analysis and Design." *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.

[7] F. Zambonelli, N. Jennings, A. Omicini, and M. Wooldridge. "Agent-Oriented Software Engineering for Internet Applications." In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf (eds), *Coordination of Internet Agents: Models, Technologies and Applications*, chapter 13. Springer-Verlag, 2000.

[8] O. Gutknecht and J. Ferber. http://www.madkit.org. MadKit official web site.

[9] B. Durand. *Simulation multi-agent et épidémiologie opérationnelle. Étude d'épizooties de fièvre aphteuse*. Thèse de doctorat (informatique), université de Caen (France), Jun. 1996.

[10] J. Collis and D. Ndumu. *The Role Modelling Guide*, Aug. 1999.

[11] J. Searle. *Speech Acts*. Cambridge University Press, 1969.

[12] A. Omicini. "SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems." In AOSE'2000, Limerick (Ireland), Jun. 2000.

[13] M. Wood and S. DeLoach. "An Overview of the Multiagent Systems Engineering Methodology." In AOSE'2000, Limerick (Ireland), Jun. 2000.

[14] J. Odell, H. Van Dyke Parunak, and B. Bauer. "Extending UML for Agents." In AOSE'2000, Limerick (Ireland), Jun. 2000.

[15] C. Oechslein, F. Klügl, R. Herrler, and F. Puppe. "UML for Behaviour-Oriented Multi-Agent Simulations." In B. M. Dunin-Kęplicz and E. Nawarecki (eds). *CEEMAS 2001*, volume 2296 of *LNCS*, pp. 217–226. Springer-Verlag, 2002.

[16] A. Drogoul. *De la simulation multi-agent à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les SMA*. Thèse de doctorat (informatique), université Paris 6 (France), 1993.

[17] F. Klügl, F. Puppe, U. Raub, and J. Tautz. "Simulating Multiple Emergent Phenonema — Exemplified in an Ant Colony." In C. Adami et al. (eds), *Artificial Life VI*, UCLA. A Bradford book, MIT Press, 1998.

[18] P. Ballet. *Intérêts mutuels des SMA et de l'immunologie — Application à l'immunologie, l'hématologie et au traitement d'images*. Thèse de doctorat (informatique), université de Bretagne occidentale (France), Jan. 2000.

[19] J. Ferber and J.-P. Müller. "Influences and Reaction: a Model of Situated Multiagent Systems." In *ICMAS'96*, Kyoto (Japan), Dec. 1996.

[20] P. Gruer, V. Hilaire, and A. Koukam. "Towards Verification of Multi-Agents Systems." In *ICMAS'2000*, pp. 393–394, Boston (USA), Jul. 2000.

[21] R. Canal. "Environnement et réaction en chaîne — Le cas des systèmes multi-agents situés." In J.-P. Barthès, V. Chevrier, and C. Brassac (eds). *JFIADSMA'98*, pp. 235–250. Hermès, Nov. 1998.

[22] M. Amiguet. *MOCA : un modèle componentiel dynamique pour les systèmes multi-agents organisationnels*. Thèse de doctorat (informatique), université de Neuchâtel (Switzerland), 2003.

[23] H. McCormick, R. Malveaux, T. Mowbray, and W. Brown, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons, 1998.

[24] R. Tolksdorf. "Coordination Patterns of Mobile Information Agents." In M. Klusch and G. Weiß (eds), *Cooperative Information Agents II*, volume 1435 of *LNAI*, pp. 246–261. Springer-Verlag, 1998.

[25] E. Hung and J. Pasquale. "Agent Usage Patterns: Bridging the Gap Between Agent-Based Application and Middleware." technical report CS1999-0638, Department of Computer Science and Engineering, UCSD, Nov. 1999.

[26] A. Silva and J. Delgado. "The Agent Pattern for Mobile Agent Systems." In *EuroPLoP'98*, Irsee (Germany), 1998.

[27] M. Occello and J.-L. Koning. "Multiagent Oriented Software Engineering: An Approach Based on Model and Software Reuse." In P. Petta and J. P. Müller (eds), *Second International Symposium From Agent Theory to Agent Implementation*, Vienna (Austria), Apr. 2000.

[28] Y. Aridor and D. Lange. "Agent Design Patterns: Elements of Agent Application Design." In P. Sycara and M. Wooldridge (eds), *Agents'98*. ACM Press, May 1998.

[29] E. Kendall, P. Murali Krishna, C. Pathak, and C. Suresh. "Patterns of Intelligent and Mobile Agents." In P. Sycara and M. Wooldridge (eds), *Agents'98*. ACM Press, May 1998.

[30] D. Deugo, F. Oppacher, J. Kuester, and I. Von Otte. "Patterns as a Means for Intelligent Software Engineering." In *IC-AI'99*, pp. 605–611. CSREA Press, 1999.

[31] D. Deugo, E. Kendall, and M. Weiß. "Agent Patterns." `http://www.scs.carleton.ca/~deugo/Patterns/Agent/Presentations/AgentPatterns`, Nov. 1999.

[32] A. Aarsten, D. Brugali, and G. Menga. "Patterns for Cooperation." In *PLoP'96*, Monticello (USA), Sep. 1996.

[33] S. Sauvage. "MAS Oriented Patterns." In B. Dunin-Kęplicz and E. Nawarecki (eds). *CEEMAS 2001*, volume 2296 of *LNCS*, pp. 283–292. Springer-Verlag, 2002.