Design Patterns for Multi-Agent Systems: A Systematic Literature Review

Joanna Juziuk, Danny Weyns, and Tom Holvoet

Abstract Design patterns document a field's systematic knowledge derived from experiences. Despite the vast body of work in the field of multi-agent systems (MAS), design patterns for MAS are not popular among software practitioners. As MAS have features that are widely considered as key to engineering complex distributed applications, it is important to provide a clear overview of existing patterns to make this knowledge accessible. To that end, we performed a systematic literature review covering the main publication venues of the field since 1998, resulting in 206 patterns. The study shows that (1) there is a lack of a standard template for documenting design patterns for MAS, which hampers the use of patterns by practitioners, (2) associations between patterns are poorly described, which results in a lack of overview of the pattern space, (3) patterns for MAS have been used for a variety of application domains, which underpins their high potential for practitioners, and (4) classifications of design patterns for MAS are bounded to specific pattern catalogs, a more holistic view on the pattern space is missing. From our study, we outline a number of guidelines that are important for future work on design patterns for MAS and their adoption in practice.

Key words: Design patterns, multi-agent systems, systematic literature review

Joanna Juziuk and Danny Weyns
Department of Computer Science, Linnaeus University, 35195 Växjö, Sweden, e-mail: jjuziuk@gmail.com and e-mail: danny.weyns@lnu.se

Tom Holvoet
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven, Belgium, e-mail: tom.holvoet@cs.kuleuven.be

1 Introduction

Capturing design knowledge in the form of patterns is a common practice in mainstream software engineering [2, 18, 19, 51]. Design patterns allow reuse of best practices and avoiding worst. The usefulness of patterns has been proven empirically [42, 47]. Design patterns improve software's quality properties, like maintainability and re-usability, and speed up the development time. These factors are crucial in practice, especially for project managers, since improving them reduces costs.

During the last decade, the multi-agent system (MAS) community has put significant efforts in documenting design patterns. Despite the substantial body of work, design patterns for MAS have not received the attention they deserve, neither in the agent-oriented software community, nor among software practitioners [59, 61]. In [11], the authors state that:

> *One of the main reasons why mainstream software developers do not benefit from MAS patterns is that they simply do not know them.*

As MAS have features that are widely considered as key to engineering complex distributed applications [35, 62], it is important to provide a clear overview of existing patterns to make this knowledge accessible to practitioners. To that end, we performed a systematic literature review covering the main publication venues of the field since 1998. From the 815 studies considered, 39 were included in the study, resulting in 206 patterns. In this paper we report the results of this literature review, and from our findings, we outline a number of guidelines that, in our opinion, are important for future work on design patterns for MAS and their adoption in practice.

2 Background

> *To understand is to perceive patterns* - Issaiah Berlin

The concept of *design pattern* was introduced in building architecture by Christopher Alexander in the 1970s [1]. According to Alexander, building structures and town planning should be supported by design patterns. These patterns consist of three layers as shown in Figure 1.

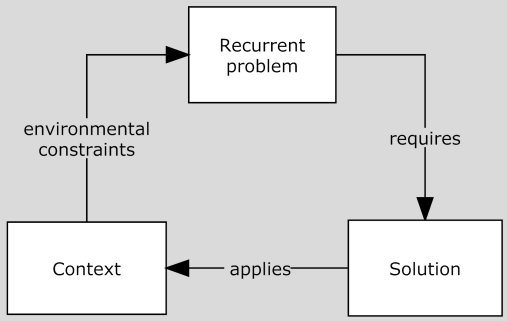


Fig. 1 Pattern anatomy [1]. The first layer embodies a recurring problem. A problem arises in a situation known as a context - i.e., the second layer. The third layer is the solution, that is, a well-known and proven solution to a problem in a context.

Design patterns have been adopted in many disciplines, for example psychology and social sciences. In the context of software engineering, patterns support better design decisions, improve communication among stakeholders, and save time by reusing proven solutions. Gabriel, as cited in [2], defines a design pattern as:

> [...] *a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves.*

Design patterns are meant to be generic, so their application and implementation may vary. Furthermore, design patterns are pragmatic, yet tested solutions, as they are derived from experience with building real, concrete systems.

Design patterns are often classified and grouped in a form of a catalog. A catalog serves as a library of expertise of successful solutions, hence it is an effective tool for learning and teaching. The Gang of Four (GoF) proposed the first classification of software design patterns [19]. This pattern catalog uses a two dimensional classification based on scope and purpose. The catalog contains 23 design patterns that were previously undocumented. Buschman's classification [5] is another well-known organization of software design patterns. This catalog promotes functionality and structural principles, and uses also a classification along two dimensions: granularity and purpose. Other popular catalogs are Fowler's pattern catalog [18], J2EE blueprints [2] for large scale enterprise applications, and Schmidt's catalog [50] that documents design patterns for concurrency.

In conclusion, design patterns are considered particularly useful assets in engineering complex systems, and various pattern catalogs have been documented.

3 Research method

The research method used in this study is a systematic literature review (SLR). An SLR is a well-defined approach to identify, evaluate and interpret all relevant studies regarding a particular research question, topic area or phenomenon of interest [31].

The study aims to provide an overview of documented design patterns for MAS. In particular, we aim to identify how the patterns are documented, whether and how the patterns are related, and for what applications the patterns have been applied. From our study, we aim to outline guidelines for future work on design patterns for MAS and in particular, their popularization in practice. The main benefit of applying an SLR is that it decreases the likelihood that the results of our study will be biased. The material of the review is available online via http://homepage.lnu.se/staff/daweaa/SLR-MASpatterns.htm

3.1 SLR Process

Figure 2 shows an overview of the SLR process we have followed. The study started with defining an initial review protocol, followed by retrieving and selecting publications, data analysis, and report writing. We organized the harvesting of the publications in four iterations. This approach was inspired by methodologies for Agile system development as their core principles are: adaptive planning, time-boxed iterations, and rapid/flexible response to change [37]. We deviated from a single harvesting step of a regular SLR [31] as we wanted to learn from each iteration and adapt the search strategy accordingly.

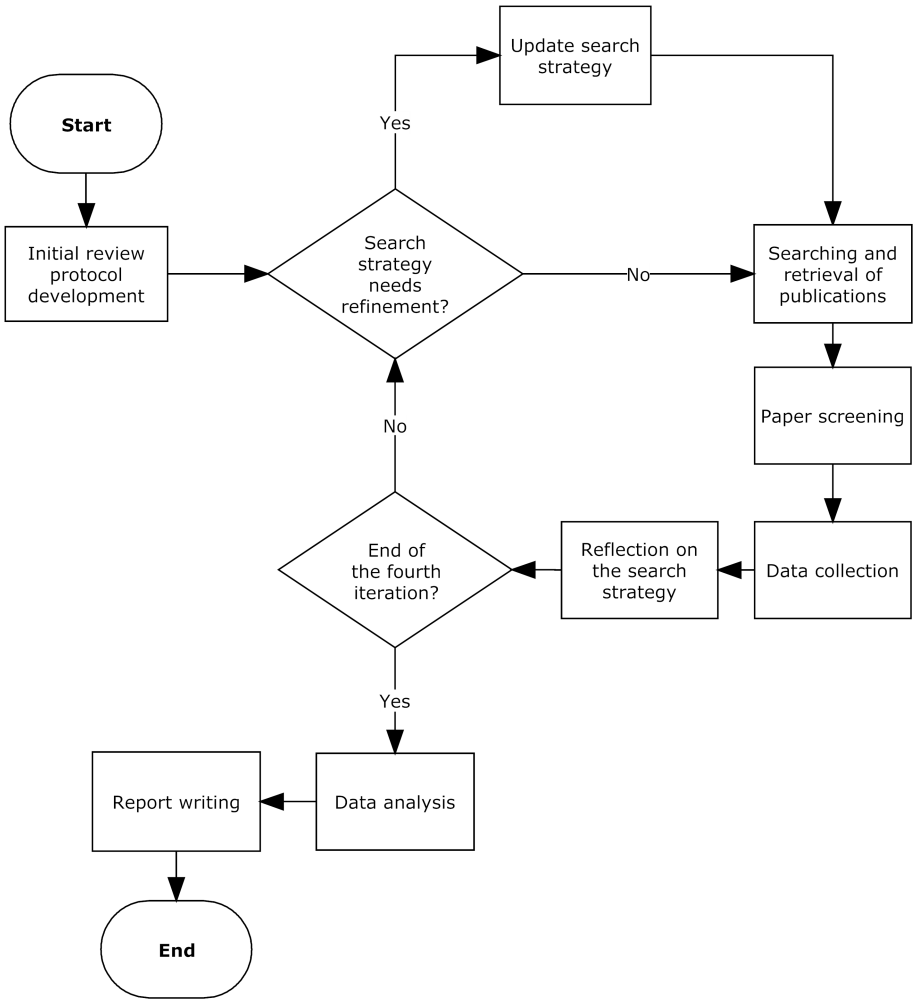


Fig. 2 SLR process used in our study

The review was performed by three researchers. Two reviewers defined the initial protocol. The actual harvesting process was performed by one reviewer, while the three reviewers evaluated the results of the subsequent harvesting steps and adapted the search strategy in consultation. One reviewer extracted the data from the selected studies. Finally, two reviewers synthesized and analyzed the data and wrote the review report. These final steps were crosschecked by the third reviewer.

3.2 Research questions

The following research questions are defined:

RQ1: How are the patterns documented and what pattern templates are used?
RQ2: How are the design patterns interconnected?
RQ3: For what types of systems have the design patterns been applied?
RQ4: How can the design patterns be classified?

As simplicity and overview of a template are crucial factors for the usability of patterns for practitioners, the motivation behind RQ1 is to study how pattern templates are used, identify whether there is any (need for) standardization to describe design patterns for MAS, and what may constitute as a common vocabulary for a future pattern language. We formulated RQ2 to understand the relationships between patterns. Visualizing associations between patterns will benefit users' learnability and help orientation in the space of design patterns for MAS. RQ3 aims to identify for which type of systems or application domains design patterns have been applied in practice. The answer may reveal potential domains for future application of design patterns for MAS. Finally, with RQ4 we aim to identify dimensions to classify design patterns. Such classification can serve as a roadmap to search for particular patterns. The coverage for the various dimensions can help in identifying areas that deserve attention in future work on patterns for MAS.

3.3 Review protocol

A review protocol is essential to any systematic literature review [31]. Driven by the research questions, the protocol defines inclusion/exclusion criteria to select primary studies, a search strategy, the data items that will be collected to answer the research questions, and finally the approach that will be used for data analysis. In the following sections, we explain in more detail how we have applied the different steps of the protocol.

3.3.1 Inclusion and exclusion criteria

A study was included if it fulfilled all the inclusion criteria, i.e.:

IC1: The study concerns design patterns for multi-agent systems,
IC2: is published between 1998 and 2012,
IC3: and the abstract and content are written in English.

A study was excluded if it fulfilled one of the exclusion criteria, i.e.:

EC1: The patterns are not described in detail, or a structured template is lacking,
EC2: a newer study exists that documents the same patterns,
EC3: or the paper concerns a review or evaluation of existing patterns for MAS.

3.3.2 Search strategy

As explained above, we followed an iterative approach to search studies. In the subsequent iterations we adopted a mixed search strategy to incorporate new search results. The first iteration included a manual search of the *International Journal of Agent-Oriented Software Engineering* (IJAOSE) and an automatic search based on a list of keywords in the electronic databases: ACM Digital Library, Science Direct and Lib Hub. The following Boolean search strings were used:

- (multi-agent OR multiagent OR MAS OR “multi-agent system” OR “multi-agent systems” OR “multiagent system” OR “multiagent systems”) AND ("design pattern" OR pattern OR patterns OR “design patterns”)
- (agent-based OR agent-oriented OR agent) AND ("design pattern" OR pattern OR patterns OR “design patterns”)

In the second iteration, the automatic search was extended to other databases: IEEE Xplore, SpringerLink and GoogleScholar. In the third iteration, after initial selection and creation of a preliminary list with patterns, other search techniques were incorporated. In particular, we searched the reference lists of the selected studies to find related missing articles and other publications of the same authors to exclude doubles of the same patterns, mainly using CiteSeerX. In the fourth iteration, we performed additional searches in three primary journals of the field: *The Knowledge Engineering Review* (KER) was searched automatically, *Transactions on Autonomous and Adaptive Systems* (TAAS) and *Journal of Autonomous Agents and Multi-Agent Systems* (JAAMAS) were searched manually.

3.3.3 Data collection

Table 1 shows data items that were collected for each paper. Data items F1-F7 were used for documentation purposes, and include authors, year, title, venue, keywords, and design pattern name/alias. Catalog pattern categories (F8) refers to different categories of patterns. We have built up a list of categories during data collection. A number of authors explicitly define categories of their patterns, such as Architectural, Mobility, Organizational, Mediation etc. Other categories were inferred from the publications’ titles, descriptions of the patterns, such as Self-organizing, Large-scale, Bio-inspired, Social etc. F8 helps determining a pattern classification, answering RQ4. Short pattern description (F9) provides a brief overview of a pattern, which helps identifying pattern relations (RQ2) and supports pattern classification (RQ4). Pattern application domains (F10) refers to specific type of systems or domains where the patterns are initially applied, and is used to answer RQ3. This item includes the options domain-independent, industrial applications (process control and manufacturing, air traffic control, traffic and transportation), robotics, entertainment (games, etc.), and simulation. These options were derived from a number of papers and books that comment on practical MAS applications, including [28, 63, 15, **?**, 22]. Pattern associations (F11) refers to explicitly docu-

mented relations with other patterns. This information can be found in template's paragraphs such as "See also" or "Related patterns." This data supports answering RQ2. Finally, pattern template details (F12) refers to the template paragraphs used to document the patterns (containing both textual and graphical data), which helps to answer RQ1.

Table 1 Data collection form

Item id	Field	Concern
F1	Author(s)	Documentation
F2	Year	Documentation
F3	Title	Documentation
F4	Venue	Documentation
F5	Keywords	Documentation
F6	Design pattern name	RQ2
F7	Design pattern alias	RQ2
F8	Catalog pattern categories	RQ4
F9	Short pattern description	RQ3, RQ4
F10	Pattern application domains	RQ3
F11	Pattern associations	RQ2
F12	Pattern template details	RQ1

3.3.4 Data analysis

The process of synthesizing and analyzing the collected data included the following steps:

1. Listing of design patterns and articles
2. Analysis of the data
3. Answering research questions
4. Interpretation of the results

We used three different methods to perform data analysis: meta analysis to answer RQ1 and RQ3, cluster analysis based on a graph model to answer RQ2, and data classification to answer RQ4. During meta analysis we defined qualitative coding schema for different topics of interest with binary parameters. For RQ1, we marked a pattern with 1 if it was documented with a structured template, and 0 otherwise. Furthermore, we systematically listed the various template paragraphs. If a pattern template contained a paragraph we marked it 1, or 0 otherwise. We applied the same approach for the documented application domains of the patterns. The collected data was further analyzed using descriptive statistics. We used a graph-based model for cluster analysis aiming to identify groups of related patterns. The objective of topical classification of the patterns was to get a clear view on the pattern space. However, the results of this method may be biased as it relies on subjective categorization.

4 Data Collection and Results

In total, 815 papers were considered for the study: 526 journal articles and 289 papers derived from digital databases. From this set, 39 were included after applying exclusion criteria. Figure 3 shows the distribution of included studies from academic databases and search engines.

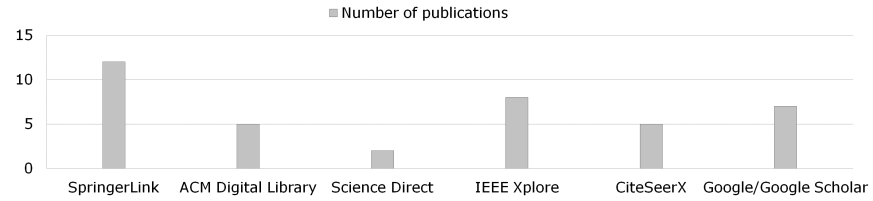
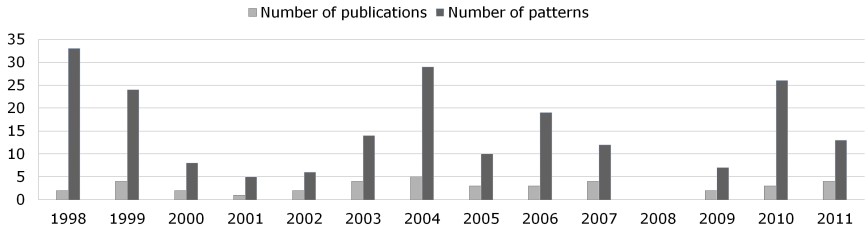


Fig. 3 Included studies from digital databases/search engines

From the 39 articles, 206 patterns were identified written by 95 researchers. Table 2 lists the patterns in chronological order. 93% of the patterns have unique names which indicates that some existing patterns where rewritten.

Figure 4 shows the distribution of documented design patterns over the years. We notice three peaks around the years: 1998, 2004, and 2010. We could not identify clear arguments for these waves of publications over time. The Appendix gives an overview of the publication venues.



	Publications	Patterns
Mean	2,79	14,71
Median	3	12,50
Variance	1,57	98,37
Maximum	5	32
Total	39	206

Fig. 4 Publications and patterns per year

4.1 How are the patterns documented and what pattern templates are used? (RQ1)

The huge number of existing patterns indicates the need for a standard approach to document patterns (F12). Overall, 69% of the patterns are described using a structured templates, which contains on average 7 paragraphs. From the pattern with a structured template, 53% contains some graphical descriptions. 73% of these graphical descriptions are modeled using UML.

Table 2 Design Patterns for MAS (chronological order)

Patterns	Ref.	Year
Itinerary, Master-Slave, Meeting, Plan, Ticket, Facilitator, Forwarding, Organized group, Locker, Messenger	[3]	1998
The Layered Agent, Reactive Agent, Deliberative Agent, Opportunistic Agent, Interface Agent, Intention, Prioritizer, Adaptable Active Object, Message Forwarder, Plan as Command, Plan and Intention Factory, Conversation, Facilitator, Agent Proxy, Protocol, Emergent Society, Clone, Remote Configurator, Broker, Migration Thread Factory, Agent Builder, Layer Linker	[29]	
Basic Mobility, Itinerary, Star-shaped Movement, Branching, Contract net protocol, Cooperation Protocol Pattern, Direct Interaction, Mediation, Dispatching	[55]	1999
Broker, Embassy, Mediator, Monitor, Wrapper	[25]	
Metamorphic Architecture Pattern, Mediator, Task Decomposition Pattern, Virtual Clustering Pattern, The Partial Cloning Pattern, The Prototyping Pattern	[52]	
Direct Coupling Pattern, Proxy Agent Pattern, Communication Sessions Pattern, Badges, Event Dispatcher	[13]	
Sentinel Agent Behavior Pattern	[7]	2000
Receptionist, Session Pattern, Secretary, Mobile Session, Antenna, Private Session, Meeting with Moderator	[53]	
Blackboard, Market Maker, Meeting, Master-Slave, Negotiating Agents	[14]	2001
Synchronizer, Environment Mediated Communication, Updating shared state, Behavior-based Decision	[49]	2002
InteRRaP, Contract net protocol	[36]	
Agency Guard, Agent Authenticator, Sandbox, Access Controller	[39]	2003
Role Agent Pattern	[6]	
Monitor, Broker, Matchmaker, Mediator, Embassy, Booking, Call-For-Proposal, Bidding	[33]	
Reflective Blackboard	[54]	
Organisation schemes, Protocols, Marks, Influences, BDI architecture, Vertical architecture, Horizontal architecture, Recursive architecture, Iniquity, Discretisation, Physical entity	[48]	2004
Hybrid Recogniser, Sense-and-Infer, Assisted Sense and Infer, Ecological Recogniser, Assisted Ecological Recogniser, Clairvoyant	[26]	
Market-organiser, Agent-side comparison-shopping, Server-side comparison-shopping pattern, Itinerary-balancing pattern	[43]	
Pipeline	[23]	
Agent Society, Agent as Delegate, Agent as Mediator, Common Vocabulary, User Agent, Task Agent, Resource Agent	[58]	

Patterns	Ref.	Year
Multi-agent layer pattern, Web usage mining pattern for adaptive multi-agent systems, WUMA-Interface, WUMA-UserM, WUMA-Acquirer, WUMA-Miner	[21]	2005
Rule Composer, Decision Tree	[57]	
Service Client Pattern, Service Representative	[41]	
Initiator, Observer	[38]	2006
Structure-in-5, Pyramid, Chain of values, The matrix, Bidding, Joint venture, Arm's-length, Hierarchical contracting, Co-optation	[34]	
Define Actors, Refine Actor Goals, Means-End Analysis, Contribution Analysis, AND/OR Decomposition, Ask for Help, Refine Models, System-to-Be	[40]	
Replication, Collective Sort, Evaporation, Aggregation, Diffusion	[20]	2007
Behaviour Helper Pattern	[8]	
Gradient Fields, Market-Based Control	[12]	
Agent Interface, SCADA control sequence agent, Agent optimization, Agent NET-MAP pattern	[16]	
Virtual Environment, Situated Agent, Selective Perception, Roles & Situated Commitments, Protocol-based Communication	[60]	2009
Perception Memory Pattern, Exponential Growth Pattern	[32]	
Disciplined flood, Propertinerary, Smart message, Delegate ant MAS, Delegate MAS	[27]	2010
Planner, VFHPlanner Pattern, A*Planner, Persistence agent, Memento agent, Social agent, Explorer, Sequential-resource share , Parallel resource share, Query, Inform, Request, Secure Query, Secure Request, Secure Inform, Contract net protocol, Publish-subscribe, Information agent, Holonic society, Supply chain	[10]	
Selection of Relevant Source Material	[9]	
Policy-based	[24]	2011
Scheduler Scramble, Context and Projection Hierarchy, Diffuser, Strategy, Logo World, Learning, Model-View-Controller, Double Buffer	[44]	
Aggregation, Spreading, Gossip	[17]	
Composite DelegateMAS	[11]	

In total, we identified 102 unique paragraphs. This huge number clearly indicates that there is no consensus about a pattern template. However, some of the paragraphs are semantically more or less equivalent and can be grouped. For example paragraphs such as Related patterns, Associated patterns, See also, References, all refer to related patterns. For many other paragraphs it is difficult to identify semantic relations, and as such grouping them is not obvious, for example, Context/Applicability, Problem/Intent/Purpose, Examples/Known Uses or Participants/Entities.

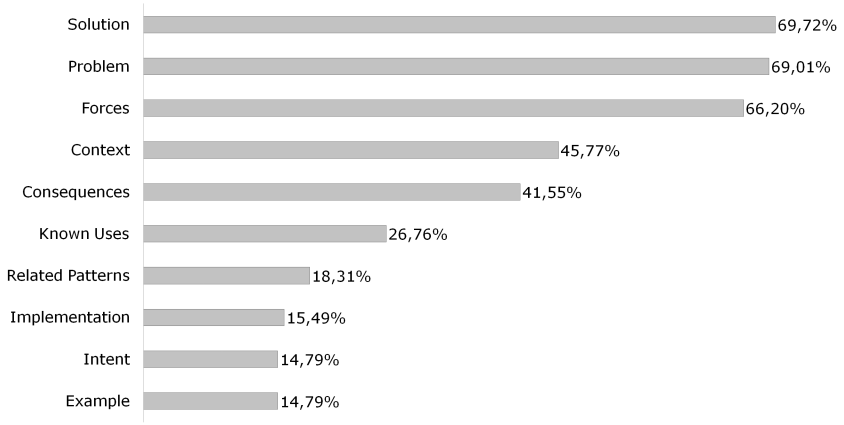


Fig. 5 Popular paragraphs in pattern templates

The most popular paragraphs are illustrated in Figure 5. Several of these paragraphs are generally considered as essential for documenting design patterns, especially Problem, Context and Solution. Other paragraphs are rare and rather unusual, for example Technical Issues, Reasoning capabilities, Temporal context, and Configuration.

4.2 How are the design patterns interconnected? (RQ2)

To answer this question, we created a graph based on data extracted from data items: Pattern associations (F11), Design pattern alias (F7) and Design pattern name (F6). Figure 6 shows the directed graph (114 nodes, 130 edges, modularity: 0,762) with associations between the patterns. Three main clusters can be distinguished.

A first cluster contains patterns that are inspired by object-oriented pattern catalogs, such as the GoF catalog [19] (Proxy, Mediator), and by the concurrency patterns of [50] (Broker, Active Object). Those patterns are mainly concerned with low level design and implementation issues and are found in early papers: [25, 29, 49, 55].

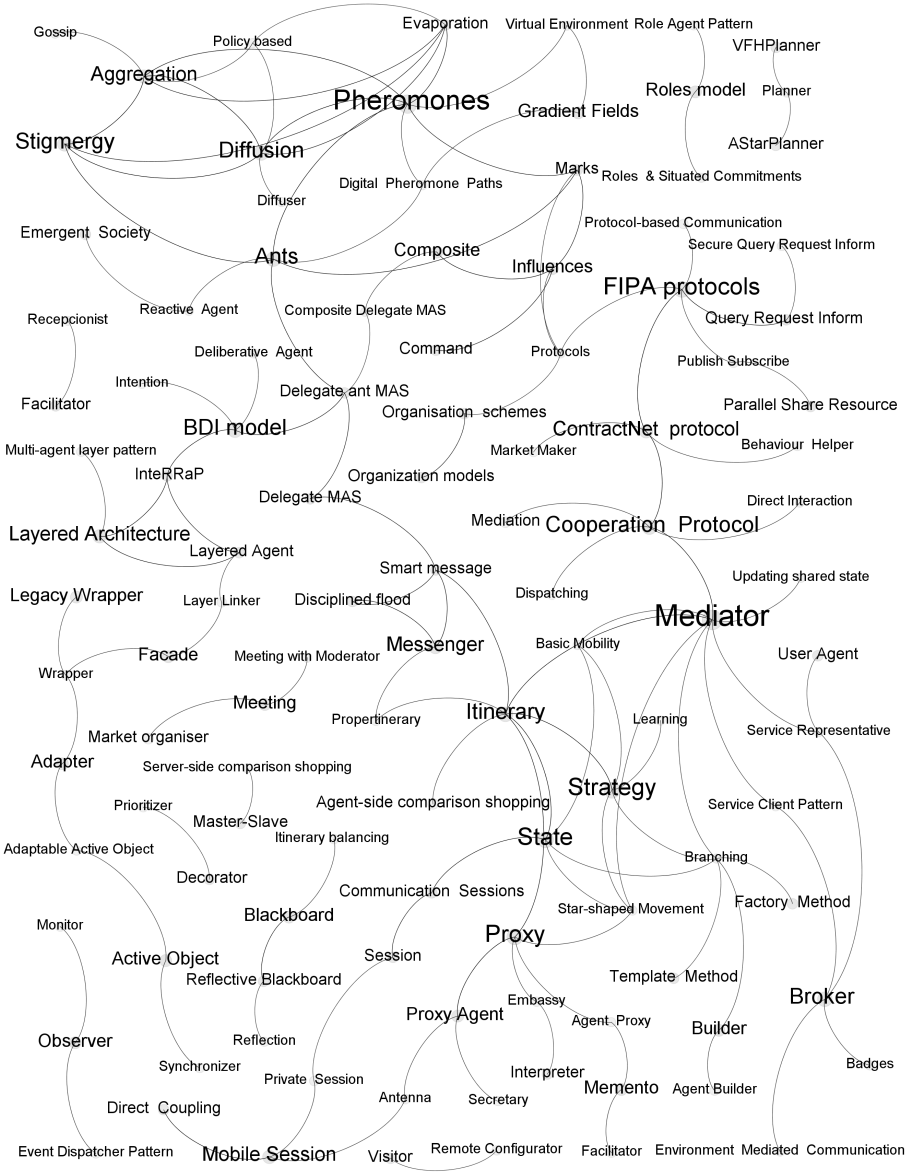


Fig. 6 Pattern space

A second cluster is built around bio-inspired concepts such as pheromones, ants, and stigmergy. This cluster includes recent patterns related to self-organization and adaptive behavior: [11, 12, 17, 20, 24, 27, 48].

A third cluster groups patterns that are related to the mobile agents and Aglets patterns [3]. This group contains patterns from: [14, 27, 43, 53].

Another cluster groups protocols related to FIPA protocols [10, 36, 55]. Several closed clusters can be identified that typically include patterns from a single catalog and as such only contains relationships within the catalog. Several of these closed clusters contain domain specific patterns, for example patterns for military simulations, e-commerce, intelligent manufacturing, and security [16, 26, 39, 43, 52].

4.3 For what types of systems have the design patterns been applied? (RQ3)

To answer this research question, we derived data from Short pattern description (F9) and Pattern application domain (F10). Overall, there is not a dominant type of system for which MAS design patterns have been used. Figure 7 shows that 59% of the patterns are domain-independent, while 41% of the patterns focus on more specific uses.

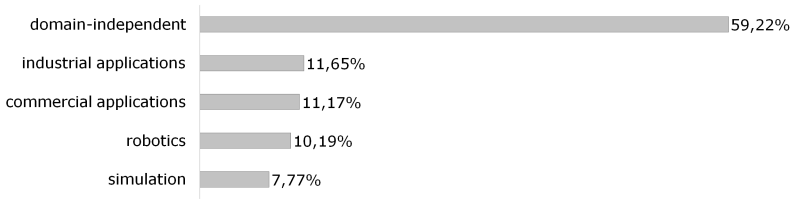


Fig. 7 Pattern application domains

For industrial applications, the main reported subdomains are process control and manufacturing (6%) and traffic and transportation (5%). For commercial applications we identified information management (4%) and electronic commerce (3%). We found no patterns applied in the entertainment domain. Nevertheless, we observe that the design patterns for MAS have been used for a diversity of application domains which reveals their high potential for software practitioners.

4.4 How can the design patterns be classified? (RQ4)

The objective of classifying design patterns for multi-agent systems is to provide an intellectual graspable overview of the huge space of existing patterns. The classification offers engineers a general picture of the pattern space of multi-agent systems, and helps those who are not familiar with the domain to get an easy jump-start to understand the pattern space. Several researchers have proposed classifications of design patterns for MAS, but these classifications are either bound to a specific

catalog of patterns, or to an development methodology [3, 10, 29, 45, 49, 55]. The classification presented in this paper covers the full space of patterns for multi-agent systems as document at the time of writing.

We derived the data for the classification resulted from Catalog pattern categories (F8) and Short pattern description (F9). Based on the analysis of the data, we identified four dimensions of patterns for multi-agent systems: inspiration, abstraction, focus, and granularity. Fig. 8 shows a graphical overview of the dimensions, illustrated with example patterns.

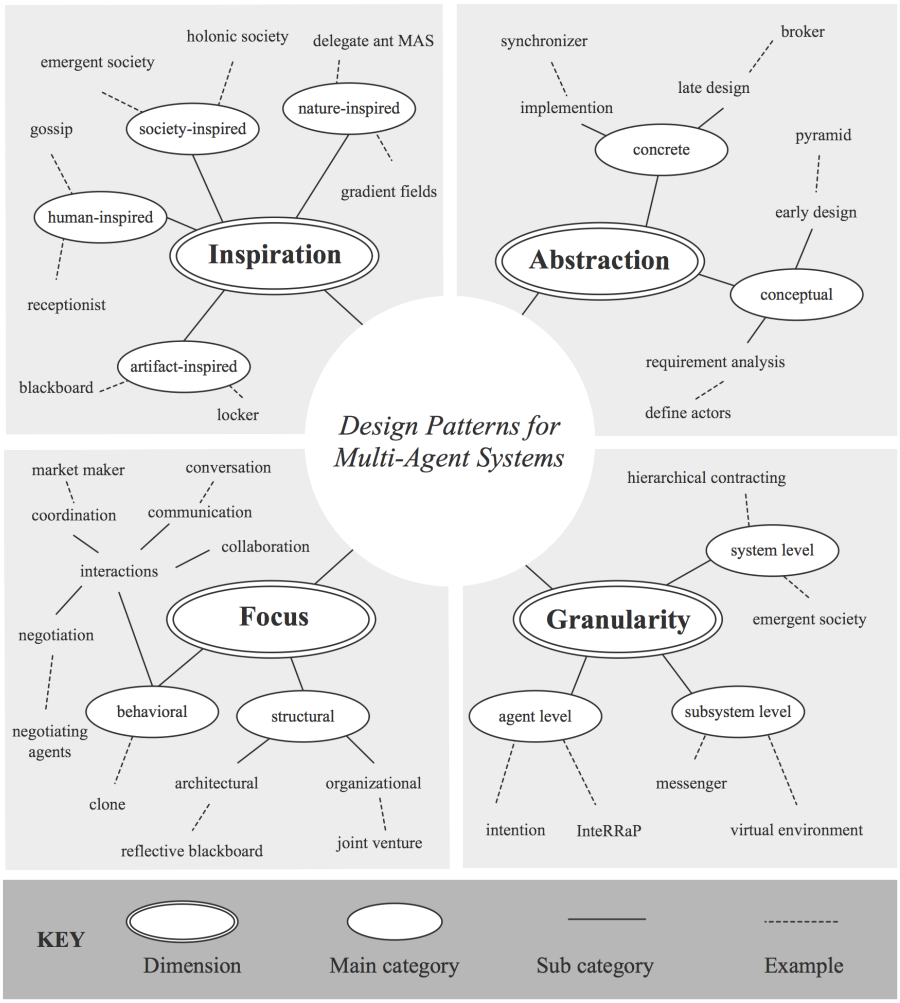


Fig. 8 Classification of patterns for multi-agent systems with example patterns.

4.4.1 Inspiration

Metaphors and analogies help in understanding complex systems. The *inspiration* dimension has four categories that provide intuitive domains from which patterns are derived. Examples of nature-inspired patterns are gradient fields [12] that is inspired by the fields in nature, and delegate ant MAS [27] that is inspired by behavior of social insects. Examples of society-inspired patterns are emergent society [29] and holonic society [10] that get their inspiration from the way societies emerge and structure themselves. Examples of human-inspired patterns are receptionist [53] and gossip [17]. Finally, locker [3] and blackboard [13] are example of patterns that get there inspiration from artifacts in our environment.

4.4.2 Abstraction

The *abstraction* dimension classifies patterns either as conceptual or concrete. Both these main categories are further refined in subcategories that refer to stages in the software life cycle where the patterns can be used. Define actors [40] and pyramid [34] are examples of patterns that are useful in early phases in the life cycle, while broker [25] and synchronizer [49] can be applied in detailed design and the implementation phase.

4.4.3 Focus

The *focus* dimension has two categories: structural and behavioral. Structural patterns are useful to deal with the decomposition of a system, while behavior patterns are useful to deal with interaction aspects. Examples of structural patterns are reflective blackboard [54] and joint venture [34], the former focusing a particular coordination structure for an agent system, the latter focusing on the way a community of agents is organized. Examples of behavioral patterns are market maker [13], negotiating agents [13], and conversation [29]. These patterns provide different approaches to support interactions among agents.

4.4.4 Granularity

Finally, the *granularity* dimension refers to the scope of the patterns, i.e., the system or parts of the system. Hierarchical contracting [34] and emergent society [29] are examples of patterns that apply to a multi-agent system as a whole. Messenger [3] and virtual environment [60] are patterns that apply to parts of a multi-agent system. Intention [29] and InteRRaP [36] are patterns that support the design of individual agents.

5 Threats to Validity

The main threat to validity of the study is a potential lack of accuracy of search results due to the search strategy. To anticipate missing papers during automatic search, we performed pilot searches to tune the search criteria. Furthermore, we performed manual searches for the journal articles. We omitted theses and technical reports as we assumed that the patterns would eventually be published in journals or conference proceedings. We limited the time frame of searching to the period 1998-2011. This is motivated by the fact that the Agents conference started around 1998. Before that, we could not find documented design patterns for MAS. Finally, the data was collected by a single reviewer, which may result in a bias. To anticipate this threat to validity, we used various strategies, including:

- triangulation of data,
- crosschecking data from multiply sources,
- member checking,
- using rich and graphical descriptions to convey the findings,
- peer examination and reviewing.

6 Conclusions and Recommendations

The objective of this literature study was to summarize existing design patterns for multi-agent systems, to make this knowledge accessible to practitioners. To that end we performed a systematic literature review aiming to answer four research questions.

The first research question was concerned with the templates used to document the patterns. Analysis of the collected data shows that there are currently no agreed pattern templates to document design patterns for MAS. In addition, we observed that many patterns are documented without structured templates. This observation hampers the accessibility of the patterns for practitioners as well as students. Hence, there is a need for standard templates to document patterns for MAS. Such template should clearly define the semantics of the different paragraphs. We suggest to start from a state of the art pattern template, and include at least the following paragraphs: Name, Problem, Solution, Context, Forces, Consequences, and Related Patterns. The first four paragraphs are essential as any design pattern should document a solution to recurring problem in a given context. Context and forces should be also included as these aspects are essential for software engineers to select proper design patterns and it also supports learning of the patterns. Documenting consequences is crucial to understand the design trade-offs implied by patterns. Finally, related patterns enable to connect patterns and build up pattern languages.

The goal of the second research question was to investigate how the design patterns for MAS are interconnected. We observed a strong coupling between patterns from the same catalog. However, more effort should be put in documenting associa-

tions between patterns. Moreover, duplicated efforts in describing the same patterns should be avoided.

The third research question identified the type of systems and application domains for which design patterns for MAS have been used. The collected data shows that the patterns have a wide range of applications. Nevertheless, although MAS are suggested to be primary candidates to design emerging domains such as cloud computing and smart grids, no applications of patterns for these systems are reported. Empirical evidence is required that demonstrates the usefulness of design patterns for MAS in these areas.

Finally, the fourth question was concerned with classifying design patterns for MAS. The data collected to answer this question shows that the space of design patterns for MAS is huge and very diverse. Patterns are related to a variety of disciplines, including Systems and Organization Theory, Social Sciences, Biology, Psychology or Ecology. To bring order in this huge space, we identified four dimensions that enable classification of the patterns. This classification allows engineers to better browse the pattern space and helps those who are new to the field to better find their way through the myriad of patterns.

We conclude with some ideas for future work in the area of patterns for MAS:

1. Formalization and standardization of patterns (and pattern templates) will contribute to improve the quality of pattern documentation and eliminate ambiguity.
2. Empirical evaluation of the patterns is required to demonstrate the added value of the patterns (as well as their tradeoffs).
3. Creating catalogs and pattern languages will help to better interconnect patterns, and enhance their combined use.

Other opportunities for future research are identification and documentation of anti-patterns for MAS, i.e., design patterns that have proven to be unsuccessful [4], evaluation of patterns using standard frameworks for evaluating design patterns [46], and development of CASE tools to support engineers with applying patterns during system development [10, 30, 56].

References

1. Alexander, C.: The Timeless Way of Building. Oxford U.P., New York, NY, USA (1979)
2. Alur, D., Crupi, J., Malks, D.: Core J2EE patterns: best practices and design strategies, 2 ed. edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (2003)
3. Aridor, Y., Lange, D.B.: Agent design patterns: elements of agent application design. In: Proceedings of the 2nd International Conference on Autonomous Agents, pp. 108–115. ACM, New York, NY, USA (1998)
4. Brown, W.J., Malveau, R.C., McCormick, H.W., Mowbray, T.J.: AntiPatterns: refactoring software, architectures, and projects in crisis. John Wiley & Sons, New York, NY, USA (1998)
5. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented software architecture. [Vol. 1], A system of patterns. John Wiley & Sons, Chichester, UK (1996)
6. Cabri, G., Ferrari, L., Leonardi, L.: Role agent pattern: a developer guideline. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 5, pp. 4114 – 4119 (2003)

7. Chacï¿œn, D., Mccormick, J., Mcgrath, S., Stoneking, C.: Rapid application development using agent itinerary patterns. Technical Report 01, Lockheed Martin Advanced Technology Laboratories (2000)
8. Charan Ojha, A., Kumar Pradhan, S., Ranjan Patra, M.: Pattern-based design for intelligent mobile agents. In: Proceedings of the 4th International Conference on Innovations in Information Technology, pp. 501 –505 (2007)
9. Cheah, W., Sterling, L., Taveter, K.: Task knowledge patterns reuse in multi-agent systems development. In: Proceedings of the 13th International Conference on Principles and Practice of Multi-Agent Systems, PRIMA'10, pp. 459–474. Springer, Berlin (2010)
10. Chella, A., Cossentino, M., Gaglio, S., Sabatucci, L., Seidita, V.: Agent-oriented software patterns for rapid and affordable robot programming. J. Syst. Softw. **83**, 557–573 (2010)
11. Cruz Torres, M., Van Beers, T., Holvoet, T.: (No) more design patterns for multi-agent systems. In: Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, VMIL'11, SPLASH '11 Workshops, pp. 213–220. ACM, New York, NY, USA (2011)
12. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. In: Engineering Self-Organising Systems, *Lecture Notes in Computer Science*, vol. 4335, pp. 28–49. Springer, Berlin, Heidelberg, Germany (2007)
13. Deugo, D., Oppacher, F., Kuester, J., Otte, I.V.: Patterns as a means for intelligent software engineering. In: Proceedings of the 1999 Conference on Artificial Intelligence, pp. 605–611 (1999)
14. Deugo, D., Weiss, M., Kendall, E.: Reusable patterns for agent coordination. In: Coordination of Internet agents, chap. Reusable patterns for agent coordination, pp. 347–368. Springer-Verlag, London, UK (2001)
15. Dignum, V., Dignum, F.: Designing agent systems: state of the practice. Int. J. Agent-Oriented Softw. Eng. **4**(3), 224–243 (2010)
16. Eichelkraut, C., Etzkorn, L.: Describing agent based real-time distributed systems using design patterns. In: Proceedings of the 45th Annual Southeast Regional Conference, ACM-SE 45, pp. 156–161. ACM, New York, NY, USA (2007)
17. Fernandez-Marquez, J.L., Arcos, J.L., Di Marzo Serugendo, G., Casadei, M.: Description and composition of bio-inspired design patterns: The gossip case. In: Engineering of Autonomic and Autonomous Systems (EASe), 2011 8th IEEE International Conference and Workshops on, pp. 87 –96 (2011)
18. Fowler, M.: Patterns of enterprise application architecture. Addison-Wesley, Boston, MA, USA (2003)
19. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA, USA (1995)
20. Gardelli, L., Viroli, M., Omicini, A.: Design patterns for self-organising systems. In: H.D. Burkhard, G. Lindemann, R. Verbrugge, L.Z. Varga (eds.) Multi-Agent Systems and Applications V, *LNCS (LNAI)*, vol. 4696, pp. 123–132. Springer-Verlag, Berlin, Heidelberg, Germany (2007)
21. Girardi, R., Marinho, L.B., de Oliveira, I.R.: A system of agent-based software patterns for user modeling based on usage mining. Interact. Comput. **17**(5), 567–591 (2005)
22. Glass, R.L., Vessey, I.: Naivete squared: In search of two taxonomies and a mapping between them. IEEE Softw. **28**(5), 14–15 (2011)
23. Gonzalez-Palacios, J., Luck, M.: A framework for patterns in gaia: A case-study with organisations. In: Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering (2004)
24. Guo, Y., Mao, X., Hu, C.: Design pattern for self-organization multi-agent systems based on policy. In: Proceedings of the 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1572 –1577 (2011)
25. Hayden, S.C., Carrick, C., Yang, Q.: Architectural design patterns for multiagent coordination. In: Proceedings of the 3rd international conference on autonomous agents (1999)
26. Heinze, C.: Modelling intention recognition for intelligent agent systems. Research Report DSTO-RR-0286, Air Operations Division Systems Sciences Laboratory (2004)

27. Holvoet, T., Weyns, D., Valckenaers, P.: Delegate MAS patterns for large-scale distributed coordination and control applications. In: Proceedings of the 15th European Conference on Pattern Languages of Programs. ACM (2010)
28. Jennings, N., Wooldridge, M.: Agent technology: foundations, applications, and markets. Springer, Berlin, Germany (1998)
29. Kendall, E.A., Murali Krishna, P.V., Pathak, C.V., Suresh, C.B.: Patterns of intelligent and mobile agents. In: Proceedings of the 2nd International Conference on Autonomous Agents, pp. 92–99. ACM (1998)
30. Khwaja, S., Alshayeb, M.: Towards design pattern definition language. Softw: Pract. Exper (2011)
31. Kitchenham, B.: Procedures for performing systematic reviews. Technical Report TR/SE-0401, Department of Computer Science, Keele University, Department of Computer Science, Keele University, UK (2004)
32. Klï¿œgl, F., Karlsson, L.: Towards pattern-oriented design of agent-based simulation models. In: Proceedings of the 7th German conference on Multiagent system technologies, MATES'09, pp. 41–53. Springer-Verlag, Berlin, Heidelberg, Germany (2009)
33. Kolp, M., Do, T.T., Pirotte, A.: Social patterns for designing multiagent systems. In: Proceedings of the 15th International Conference on Software Engineering & Knowledge Engineering (SEKE'2003), pp. 103–110 (2003)
34. Kolp, M., Giorgini, P., Mylopoulos, J.: Multi-agent architectures as organizational structures. Autonomous Agents and Multi-Agent Systems **13**, 3–25 (2006)
35. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: FOSE '07: Future of Software Engineering. IEEE Computer Society (2007)
36. Lind, J.: Patterns in agent-oriented software engineering. In: Proceedings of the 3rd international conference on Agent-oriented software engineering III, AOSE'02, pp. 47–58. Springer-Verlag, Berlin, Heidelberg, Germany (2003)
37. Martin, R.: Agile software development : principles, patterns, and practices. Prentice Hall, Upper Saddle River, NJ, USA (2003)
38. Mohamad, R., Deris, S., Ammar, H.H.: Agent design patterns framework for MaSE/POAD methodology. In: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, AICCSA '06, pp. 521–528. IEEE Computer Society, Washington, DC, USA (2006)
39. Mouratidis, H., Giorgini, P., Schumacher, M.: Security patterns for agent systems. In: Proceedings of the 8th European Conference on Pattern Languages of Programs (2003)
40. Mouratidis, H., Weiss, M.: Patterns for modelling agent systems with tropos. In: A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, A. Romanovsky (eds.) Software Engineering for Multi-Agent Systems IV, *Lecture Notes in Computer Science*, vol. 3914, pp. 207–223. Springer-Verlag (2006)
41. Mï¿œller, I., Braun, P., Kowalczyk, R.: Design patterns for agent-based service composition in the web. In: Proceedings of the International Conference on Quality Software (QSIC 2005), pp. 425 – 430 (2005)
42. Ng, T.H., Cheung, S.C., Chan, W.K., Yu, Y.T.: Work experience versus refactoring to design patterns: a controlled experiment. In: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering, SIGSOFT '06/FSE-14, pp. 12–22. ACM, New York, NY, USA (2006)
43. Nguyen, A., Yang, X.: Design patterns for mobile agent-mediated e-business. In: Proceedings of the 10th Australian World Wide Web Conference (2004)
44. North, M.J., Macal, C.M.: Product design patterns for agent-based modeling. In: Proceedings of the 2011 Winter Simulation Conference (WSC), pp. 3082 –3093 (2011)
45. Oluyomi, A.: Patterns and protocols for agent oriented software development. Phd, The University of Melbourne (2006)
46. Petter, S., Khazanchi, D., Murphy, J.D.: A design science based evaluation framework for patterns. SIGMIS Database **41**(3), 9–26 (2010)

47. Prechelt, L., Unger-Lamprecht, B., Philippsen, M., Tichy, W.F.: Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. IEEE Trans. Softw. Eng. **28**(6), 595–606 (2002)
48. Sauvage, S.: Design patterns for multiagent systems design. In: MICAI 2004: Advances in Artificial Intelligence, *Lecture Notes in Computer Science*, vol. 2972, pp. 352–361. Springer Berlin / Heidelberg (2004)
49. Schelfthout, K., Coninx, T., Helleboogh, A., Holvoet, T., Steegmans, E., Weyns, D.: Agent implementation patterns. In: Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, pp. 119–130 (2002)
50. Schmidt, D.C., Stal, M., Rohnert, H., Buschmann, F.: Pattern-oriented software architecture. Vol. 2, Patterns for concurrent and networked objects. John Wiley & Sons, Chichester, UK (2000)
51. Shaw, M., Clements, P.: The golden age of software architecture. IEEE Softw. **23**(2), 31–39 (2006)
52. Shu, S., Norrie, D.H.: Patterns for adaptive multi-agent systems in intelligent manufacturing. In: Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems, pp. 67–74 (1999)
53. Silva, I.C., da Silva, A.R., Meira, N.: A set of agent patterns for a more expressive approach. In: Proceedings of the EuroPLOP 2000, pp. 331–346 (2000)
54. Silva, O., Garcia, A., Lucena, C.: The reflective blackboard pattern: Architecting large multi-agent systems. In: A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, J. Castro (eds.) Software Engineering for Large-Scale Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 2603, pp. 73–93. Springer Berlin, Heidelberg (2003)
55. Tahara, Y., Ohsuga, A., Honiden, S.: Agent system development method based on agent patterns. In: Proceedings of the 21st International Conference on Software Engineering, ICSE '99, pp. 356–367. ACM, New York, NY, USA (1999)
56. Taibi, T., Check Ling Ngo, D.: Formal specification of design patterns - a balanced approach. Journal of Object Technology **2**(4), 127–140 (2003)
57. Taylor, P.R., Evans-Greenwood P.and Odell, J.: The genesis of a pattern language for agent-based enterprise systems. In: Proceedings of the 5th International Conference on Quality Software, QSIC '05, pp. 395–400. IEEE Computer Society, Washington, DC, USA (2005)
58. Weiss, M.: A pattern language for motivating the use of agents. In: P. Giorgini, B. Henderson-Sellers, M. Winikoff (eds.) Agent-Oriented Information Systems, *Lecture Notes in Computer Science*, vol. 3030, pp. 142–157. Springer Berlin / Heidelberg (2004)
59. Weyns, D.: An architecture-centric approach for software engineering with situated multi-agent systems. The Knowledge Engineering Review **22**(4), 405–413 (2007)
60. Weyns, D.: A pattern language for multi-agent systems. In: Proceedings of Joint Working IEEE/IFIP European Conference on Software Architecture, pp. 191 –200 (2009)
61. Weyns, D., Helleboogh, A., Holvoet, T.: How to get multi-agent systems accepted in industry? Int. J. Agent-Oriented Softw. Eng. **3**, 383–390 (2009)
62. Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Goschka, K.: On patterns for decentralized control in self-adaptive systems. Lecture Notes in Computer Science vol. 7475. Springer-Verlag, Berlin, Heidelberg, Germany (2012)
63. Wooldridge, M.J.: An introduction to multiagent systems, 2 ed. edn. John Wiley & Sons, Chichester, UK (2009)

Appendix

Year	Ref.	Venue
1998	[3, 29]	AGENTS'98: The 2nd International Conference on Autonomous Agents
1999	[55]	ICSE'99: The 21st International Conference on Software Engineering
	[25]	AGENTS'99: The 3rd International Conference on Autonomous Agents
	[13]	IC-AI'99: The International Conference on Artificial Intelligence
	[52]	The 2nd International Workshop on Intelligent Manufacturing Systems
2000	[53]	EuroPloP'2000: The 5th European Conference on Pattern Languages of Programs
2002	[49]	OOPSLA'02: Workshop on Agent-Oriented Methodologies
	[36]	AOSE'02: The 3rd International Conference on Agent-oriented Software Engineering
2003	[33]	SEKE'03:The 15th International Conference on Software Engineering and Knowledge Engineering
	[39]	EuroPLoP'03: The 8h European Conference on Pattern Languages of Program
	[6]	IEEE SMC'03: IEEE International Conference on Systems, Man and Cybernetics
2004	[23]	AOSE'04: The 5th International Workshop on Agent-Oriented Software Engineering
	[43]	AusWeb04: The 10th Australian World Wide Web Conference
	[48]	MICAI'04: The 3rd Mexican International Conference on Artificial Intelligence
2005	[57]	QSIC 2005: The 5th International Conference on Quality Software
2006	[38]	ACS/IEEE International Conference on Computer Systems and Applications
2007	[8]	The 4th International Conference on Innovations in Information Technology
	[16]	ACM-SE 45: The 45th annual ACM southeast regional conference
	[20]	CEEMAS'07: 5th International Central and Eastern European Conference on Multi-Agent Systems
2009	[60]	WICSA/ECSA'09: Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture
	[32]	MATES'09: The 7th German Conference on Multi-Agent System Technologies
2010	[27]	EuroPLoP'10: The 15th European Conference on Pattern Languages of Programs
	[9]	PRIMA'10: The 13th international conference on Principles and Practice of Multi-Agent Systems
2011	[11]	AGERE!: Programming Systems, Languages, and Applications based on Actors, Agents, and Decentralized Control
	[17]	EASe: The 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems
	[24]	TrustCom'11: The 10th International Conference on Trust, Security and Privacy in Computing and Communications
	[44]	WSC'11: Winter Simulation Conference