Degree project

Design Patterns for Multi-Agent Systems

Author: Joanna Juziuk
Date: 2012-06-26
Subject: Computer Science
Level: Master
Course code: 5DV00E
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, for example in manufacturing, robotics, ecommerce, traffic control and coordination, science simulations, it is important to provide a clear overview of existing patterns to make this knowledge accessible. To that end, a systematic literature review was performed 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 the study, a number of guidelines is outlined that are important for future work on design patterns for MAS and their adoption in practice.

Keywords

design patterns, multi-agent systems, classification, guidelines
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Agent</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>MAS - associated disciplines</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Multi-agent system</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Pattern anatomy</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>Master-Slave - class diagram</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>SRL process</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>ER diagram of the database</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Interface of the application - pattern list view</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Interface of the application - publication view</td>
<td>15</td>
</tr>
<tr>
<td>3.5</td>
<td>CakePHP architecture</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Included publications in digital databases/search engines</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Publications and patterns per year</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>Popular paragraphs in pattern templates</td>
<td>21</td>
</tr>
<tr>
<td>4.4</td>
<td>Pattern space</td>
<td>22</td>
</tr>
<tr>
<td>4.5</td>
<td>Pattern application domains</td>
<td>23</td>
</tr>
<tr>
<td>4.6</td>
<td>MAS patterns dimensions</td>
<td>24</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Objects vs Agents - comparison ............................... 4
3.1 Data collection form ........................................... 13
4.1 Design patterns for MAS (chronological order) .............. 17
Contents

1 Introduction .......................................................... 1
  1.1 Problem .......................................................... 1
  1.2 Purpose and research questions ............................... 1
  1.3 Significance of the study .................................... 2
  1.4 Outline .......................................................... 2

2 Background ................................................................ 3
  2.1 Agent ................................................................. 3
    2.1.1 Definition ...................................................... 3
    2.1.2 Agent properties ........................................... 4
    2.1.3 Agent vs object ............................................. 4
  2.2 Multi-agent system ................................................ 5
    2.2.1 Origins ........................................................ 5
    2.2.2 Definition .................................................... 6
  2.3 Design pattern ...................................................... 7
  2.4 Design pattern for MAS ......................................... 8
    2.4.1 Early works .................................................. 8
    2.4.2 Short example .............................................. 8

3 Research method and realization .................................. 10
  3.1 Overview .......................................................... 10
  3.2 Research questions .............................................. 11
  3.3 Review protocol .................................................. 11
    3.3.1 Inclusion and exclusion criteria ......................... 11
    3.3.2 Search strategy and selection process .................. 12
    3.3.3 Data collection items .................................... 12
    3.3.4 Data collection application ............................. 13
    3.3.5 Data analysis .............................................. 16

4 Results ..................................................................... 17
  4.1 Publications overview .......................................... 17
  4.2 Patterns overview ............................................... 17
  4.3 RQ1: How are the patterns documented and what pattern templates are used? .......... 20
  4.4 RQ2: How are the design patterns connected? ............ 21
  4.5 RQ3: For what types of systems have the design patterns been applied? ............. 23
  4.6 RQ4: What dimensions of the design patterns exist? .......... 24

5 Discussion and conclusions ....................................... 26
  5.1 Threats to validity and limitations .......................... 26
  5.2 Discussion on the results ...................................... 26
  5.3 Future research .................................................. 27

Bibliography ................................................................. 28
1 Introduction

Chapter 1 highlights the problem, presents aim and research questions of the thesis.

1.1 Problem

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. Cruz Torres et al. (2011) 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.

First aspect of the topic - multi-agent systems, is connected to cloud and smart grid computing, robotics, manufacturing, traffic control, ecommerce and any other application domains that are becoming more important because of the rapid growth of the big data and need for automatization. Multi-agent systems and agent-oriented paradigm are more adequate and effective means to model such distributed, autonomous systems than object-oriented paradigm that in past twenty years has been achieving its peak popularity.

Second aspect of the topic - design patterns is a practice of capturing design knowledge in the form of patterns. It is a common practice in mainstream software engineering (Alur et al., 2003; Gamma et al., 1995; Fowler, 2003; Shaw and Clements, 2006). Design patterns allow reuse of best practices and avoiding worst. The usefulness of patterns has been proved empirically (Prechelt et al., 2002; Ng et al., 2006; Yskout et al., 2012). 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 financial costs.

In conclusion, as MAS have features that are widely considered as key to engineering complex distributed applications (Kramer and Magee, 2007; Weyns et al., 2012), it is important to provide a clear, holistic overview of existing patterns to make this knowledge accessible to practitioners and unlock the potential.

1.2 Purpose and research questions

The goal of the thesis is to get an overview of the existing patterns for MAS and their use, then outline a number of guidelines that would be important for future work and their adoption in practice. The study addresses the following questions:

- How are the patterns documented and what pattern templates are used?
- How are the design patterns connected?
- For what types of systems have the design patterns been applied?
- How can the design patterns be classified?
1.3 Significance of the study

Since the significance of the problem is dictated by the audience and benefits if the study is done, it could be claimed that that study is significant on a large scale. The significance of this study is anchored in the empirical findings, so that it is potentially useful to every practitioner in the area of multi-agent systems.

1.4 Outline

The thesis is organized by five chapters. Chapter 1 highlights the problem to the reader. It also presents research questions and goal of the thesis. Chapter 2 presents the theoretical background of agents, multi-agent systems and design patterns. Chapter 3 contains the description of the research method - systematic literature review and the review protocol. Chapter 4 focuses on the results and includes four sections dedicated for each research question. Finally, chapter 5 ends the thesis with a discussion on the results, guidelines and conclusions.
2 Background

This chapter highlights agent, multi-agent systems and design patterns as the cornerstone concepts to this thesis.

2.1 Agent

This subchapter focuses on the agent concept. It gives an overview of different agent definitions, outlines a number of agent characteristics and presents the comparison of agent and object as object-oriented paradigm has been dominant in past twenty years.

2.1.1 Definition

Several agent definitions have been proposed:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. (Wooldridge, 2009)

An agent is a physical or virtual entity
- Which is capable of acting in an environment,
- Which can communicate directly with other agents,
- Which is driven by a set of tendencies,
- Which possesses resources of its own,
- Which is capable of perceiving its environment,
- Which has only a partial representation of this environment,
- Which possesses skills and can offer services,
- Which may be able to reproduce itself,
- Whose behavior tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representations and the communications it receives. (Ferber, 1999)

Generally, an agent is a computer system that acts on human behalf to do delegated tasks in semi-intelligent way and is situated in the environment (Figure 2.1).

![Agent Diagram](image)

Figure 2.1: Agent

Agents are the recent links in software engineering evolution from monolithic systems, following their predecessors - objects, abstract data types, procedures &
2.1.2 Agent properties

According to Odell (2007) among key agent attributes are:

- autonomous - capable of to autonomous actions without external intervention in order to achieve goals,
- interactive - communicating with other agents and/or environment,
- adaptive - changing, improving behavior in response to changes.

Other attributes:

- intelligent - capable of using autonomous and flexible actions for achieving goals and its state is dictated by knowledge,
- flexible - combining reactive, proactive and social attributes,
- reactive - monitoring the environment and responding, replanning and reorganizing in order to adapt to environment changes,
- proactive - setting a plan to achieve a goal and working towards the plan,
- social - developing social relations,
- mobile - capable of transporting itself,
- trustworthy - truthful and don’t intentionally act in unexpected ways.

2.1.3 Agent vs object

The main difference between objects and agents (Table 2.1) is that an object reflects ideas of devices with well-known interfaces and is an analogy to the real-world object while an agent resembles a human being and is actively pursuing its own agendas. Moreover, objects have autonomies over their states, not over their behaviors, while agents are autonomous in both aspects.

<table>
<thead>
<tr>
<th>Element</th>
<th>Objects</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Classes</td>
<td>Domain specific role</td>
</tr>
<tr>
<td>Responsibility</td>
<td>Design by contract</td>
<td>Pre-/post-conditions</td>
</tr>
<tr>
<td>Delegation</td>
<td>Task</td>
<td>Goal</td>
</tr>
<tr>
<td>Reuse</td>
<td>Inheritance</td>
<td>Composability, Role multiplicity</td>
</tr>
<tr>
<td>Messaging</td>
<td>Request for service with certain parameters</td>
<td>Exchange of parts of the knowledge base</td>
</tr>
<tr>
<td>State</td>
<td>Properties and values</td>
<td>Knowledge base</td>
</tr>
</tbody>
</table>

Table 2.1: Objects vs Agents - comparison

The fundamental property of an object is that it encapsulates some state - meaning that an object has a control over its state as there are some variables internally in the object that are only accessible to that object. Agents also have their private
state, for example expressed by their beliefs. Furthermore, objects communicate by message passing represented by for example method invocations. On the contrary, agents are more flexible and do not act on every request, but rather decide if it should be done (Wooldridge, 2009; Odell, 2002).

2.2 Multi-agent system

Second core concept concept - multi-agent system is highlighted in this subchapter. Apart form the definition, the origins of the field are presented. As MAS emerged from many disciplines, it is important to show connections between those different fields in order to emphasize how complex those systems can be.

2.2.1 Origins

Wooldridge (2009) states that multi-agent systems emerged as a new field in computer science from following trends:

- ubiquity: spreading of computer processing power everywhere,
- interconnection: communication between processes, building distributed systems,
- intelligence: constructing devices capable of human-like intelligence,
- delegation: handing over the control, automating and delegating human tasks to computers and robots,
- human-orientation: the way we interact with people becomes the way we interact with computers.

Multi-agent systems are also inspired by many other disciplines (as depicted in Figure 2.2): Artificial Intelligence, Social Sciences, Distributed Computing, Philosophy, Ecology, Game Theory. This multidisciplinary character of MAS brings well-defined models and methodologies from other fields, but many different perspectives as well which makes MAS more complex and difficult to capture as a whole.
2.2.2 Definition

A multi-agent system is a system that consists of several interacting agents (Figure 2.3). In order to interact in human-like ways, agents must have capability to negotiate, cooperate, and coordinate. Hence main types of multi-agent systems are cooperative and competitive. For example, cooperative MAS concerns distributed problem solving and planning. This type is mainly used in context of track vehicle movements, sensor networks, distributed delivery. Competitive or self-interested MAS are focused on voting, actions, contract nets and deal with fairness and stability (Wooldridge, 2009).

Figure 2.3: Multi-agent system
2.3 Design pattern

The concept of design pattern was introduced in architecture and urban planning by Alexander in the 1970s (Alexander, 1979). According to Alexander, building structures and planning should be supported by design patterns. These patterns must consist of three layers as shown in Figure 2.4 - A recurring problem arises in a situation known as a context and has well-known and proven solution.

![Pattern anatomy](image)

Figure 2.4: Pattern anatomy

Design patterns have been adopted in many disciplines, from Human–computer Interaction studies to psychology and social sciences. In the context of software engineering, patterns support better design decision, improve communication among stakeholders, and save time by reusing proven solutions. Gabriel (cited in Alur et al., 2003) defines a 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 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 (Gamma et al., 1995). This pattern catalog uses a two dimensional classification based on scope and purpose. The catalog contains 23 design patterns that were previously undocumented. Buschmann et al. (1996) classification 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 (Fowler, 2003) and J2EE blueprints (Alur et al., 2003) for large scale enterprise applications, and Schmidt et al. (2000) that documents a catalog of design patterns for concurrency.

In conclusion, a wide range of pattern catalogs have been proposed, but none of them is related to some official standard. Moreover, the scopes are often limited to class or object compositions that are inadequate for modeling multi-agent systems as agents can exhibit more properties than objects such as flexibility, reactivity or proactivness, can be formed as agent societies and can have organizational or even political properties (Odell, 2002).
2.4 Design pattern for MAS

This subchapter contains a review of early works in identifying design patterns for MAS from 1998 and exemplifies one of the first MAS pattern. The main goal of this subchapter is to expose inadequateness of object-oriented paradigm while modeling MAS and forming MAS design patterns.

2.4.1 Early works

Early works in identifying design patterns for multi-agent systems have object-oriented flavor since many authors used classes and objects to document those patterns.

Aridor and Lange (1998) published one of the first proposal to identify and document agent-oriented software engineering experiences in a form of patterns. Their work includes a catalogue of domain-independent patterns for mobile agents. The patterns are filed into three categories: traveling patterns that focus on mobility management (e.g. Itinerary pattern), task patterns (e.g. Master-Slave pattern) that concern agent task’s division and allocation and interaction patterns (e.g. Meeting, Messenger patterns) that facilitate agents’ interactions. Although, Aridor and Lange distinguished three categories of patterns, they don’t describe any classification criteria. Therefore, this limitation makes it difficult to find needed patterns quickly. Those patterns address mainly implementation issues leaving problems from other phases of software development. Furthermore, only three patterns are discussed in more detail with examples using an abridged (six elements) GoF form.

Another catalog was introduced by Kendall et al. (1998). The catalog covers design patterns for internal architecture of agents, therefore those design patterns can be only applied at the source level as well. In order to build MAS architecture, they propose to use several patterns using building blocks as a metaphor. Each pattern belongs to one of the seven layers concerning mobility, translation, collaboration, actions, reasoning, beliefs, and sensory. Although the layered model helps to establish structure and order in the system’s decomposition and cognitive agency, it fails to separate various concerns, like autonomy. The template used to describe consists of five elements: problem, forces, solution, variations and known uses. The use of template paragraphs is inconsistent since most of the patterns, are described without variations and some without known uses. Furthermore, there are no guidelines that would help to extend or remove other agent aspects. Hence it is difficult to understand, maintain and reuse.

The main problem with those early design patterns is that they support object-oriented paradigm that is inadequate and outdated for modeling agent systems, because an agent as the core concept to MAS has a dual nature and can be defined either from Artificial Intelligence (AI) or Software Engineering (SE) viewpoints. In AI domain, agents must be intelligent and autonomous, so that they converse instead of non-proactive computing. In SE domain, agents are perceived as components with threads of execution that can be engaged in interaction protocols.

2.4.2 Short example

Master-Slave design pattern proposed by Aridor and Lange (1998) is an example of a design pattern for mobile agents based on cooperation. Aridor and Lange (1998) claim that "the Master-Slave pattern provides a fundamental way to reuse
code among agent classes”. It is applicable when an agent needs to delegate the tasks, because it has other parallel tasks or the tasks need to be performed at a remote destination. The Master agent sends the Slave agent to defined place, where it should succeed in doing the task and then the slave agent should return to the source with the results. Finally, the Slave agent destroys itself. The localization and specification of handing the tasks are supported by using abstract classes - Master and Slave (as in Figure 2.5)

![Figure 2.5: Master-Slave - class diagram](image-url)
3 Research method and realization

Chapter 3 focuses on the description of the methodology used in this thesis - systematic literature review and its review protocol. Firstly, an overview of the review process is presented, followed by research questions’ motivations and review protocol details - inclusion and exclusion criteria, search strategy, data collection and data analysis.

3.1 Overview

The research method used in this study is a systematic review of the literature (SLR). The study aims in providing an appropriate background in order to position future research, concerning popularization of MAS design patterns in the industry and standardization. The main benefit of incorporating SLR is that it decreases likelihood that the results of the literature review will be biased (Kitchenham, 2004).

Figure 3.1: SRL process
SRL process (Fig. 3.1) started with establishing initial protocol development, then retrieving and selecting publications, data analysis and report writing. The harvesting of the publications was divided into four iterations. This approach was inspired by Agile system development methodologies as their core principles are: adaptive planning, time-boxed iterations and rapid/flexible response to change (Martin, 2003). Since systematic literature reviews are more time consuming than normal literature reviews (Kitchenham, 2004) and the time of the study is relatively short, it is necessary to adapt these principles as they promote pragmatism and are effect-oriented.

### 3.2 Research questions

Following research questions are defined:

**RQ 1:** How are the patterns documented and what pattern templates are used?

**RQ 2:** How are the design patterns connected?

**RQ 3:** For what types of systems have the design patterns been applied?

**RQ 4:** How can the design patterns be classified?

As the simplicity and intelligibility of a template are crucial factors in reusing patterns by practitioners, the motivation behind RQ1 is to study how pattern templates are used, find out if they require standardization and what may constitute as a common and universal vocabulary for a future pattern language. The answer of RQ2 would benefit in enhancing learnability and quicker orientation in the domain of design patterns for MAS. Visualizing associations between patterns would help in capturing a holistic picture for learners. RQ3 aims to identify for which application domains design patterns have been applied in practice. The answer may reveal potential domains for future application of design patterns for MAS. RQ4 aims in identifying 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 identifying areas that deserve further attention in future work on patterns for MAS.

### 3.3 Review protocol

The review protocol as essential to any systematic literature review (Kitchenham, 2004). 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, there are detailed explanations of how the different steps of the protocol have been applied.

#### 3.3.1 Inclusion and exclusion criteria

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

**IC 1:** The study concerns design patterns for multi-agent systems,

**IC 2:** is published between 1998 and 2012,
IC 3: and the abstract and content are written in English.

Grey literature, not obtainable through mainstream publication channels, if found was also included. A study was excluded if it fulfilled one of the exclusion criteria, i.e.:

EC 1: The patterns are not described in detail, or a structured template is lacking,
EC 2: a newer study exists that documents the same patterns,
EC 3: or the paper concerns a review or evaluation of existing patterns for MAS.

3.3.2 Search strategy and selection process

During all four iterations a mixed search strategy was adopted in order to get more varied results. The first iteration included manual searches in the *International Journal of Agent-Oriented Software Engineering* (IJAOSE) and an automatic literature search based on a list of keywords in the electronic databases: ACM Digital Library, Science Direct and Lib Hub. To get a general overview of the existing contributions, following Boolean search strings were forged:

- (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”)

During the second iteration, the automatic search was continued in other databases: IEEE Xplore, SpringerLink and GoogleScholar. In the third iteration, after initial selection and creation of preliminary list with patterns, other search techniques were incorporated - targeted search, ancestor search in the lists of references and descendant search aiming in finding early works and locating later articles that cite them in their references, mainly using CiteSeerX. In the fourth iteration, targeted searches were performed and other journals searched. *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 manually searched.

The selection process was split into two phases and done concurrently and recursively during all four phases. Firstly, publications were assessed taking their title and abstract into consideration. Secondly, the full texts were screened.

3.3.3 Data collection items

Table 3.1 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. A list of categories have been built 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 item (F10) refers to specific field of application where the pattern was initially applied. Options are:

- domain-independent (not specified),
- industrial applications
  (process control and manufacturing, air traffic control, traffic and transportation),
- commercial applications
  (information management, electronic commerce, business process management),
- robotics,
- entertainment (games, interactive theater and cinema),
- simulation.

The options were elicited from papers commenting MAS practical applications (Jennings and Wooldridge, 1998; Wooldridge, 2009; Dignum and Dignum, 2010) as there is no explicit taxonomy of software application domains (Glass and Vessey, 2011). Pattern associations item (F11) and design pattern name/alias (F7, F8) are needed to collect all related, referenced patterns from template’s paragraphs such as ‘See also’ or ‘Related patterns’ (RQ2). Finally, pattern template details (F12) that consist of text paragraphs and supplied graphical descriptions, are needed in resolving RQ1.

<table>
<thead>
<tr>
<th>ITEM ID</th>
<th>FIELD</th>
<th>CONCERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Author(s)</td>
<td>Documentation</td>
</tr>
<tr>
<td>F2</td>
<td>Year</td>
<td>Documentation</td>
</tr>
<tr>
<td>F3</td>
<td>Title</td>
<td>Documentation</td>
</tr>
<tr>
<td>F4</td>
<td>Venue</td>
<td>Documentation</td>
</tr>
<tr>
<td>F5</td>
<td>Keywords</td>
<td>Documentation</td>
</tr>
<tr>
<td>F6</td>
<td>Design pattern name</td>
<td>RQ2</td>
</tr>
<tr>
<td>F7</td>
<td>Design pattern alias</td>
<td>RQ2</td>
</tr>
<tr>
<td>F8</td>
<td>Catalog pattern categories</td>
<td>RQ4</td>
</tr>
<tr>
<td>F9</td>
<td>Short pattern description</td>
<td>RQ3, RQ4</td>
</tr>
<tr>
<td>F10</td>
<td>Pattern application domains</td>
<td>RQ3</td>
</tr>
<tr>
<td>F11</td>
<td>Pattern associations</td>
<td>RQ2</td>
</tr>
<tr>
<td>F12</td>
<td>Pattern template details</td>
<td>RQ1</td>
</tr>
</tbody>
</table>

Table 3.1: Data collection form

3.3.4 Data collection application

The collected data was stored in a SQL-based relational database (Figure 3.2) and modified via a self-made MVC (Model-View-Controller) web application (Figures: 3.3, 3.4) to provide more flexibility when validating. The web application was made using PHP framework - CakePHP.
Figure 3.2: ER diagram of the database
This framework architecture follows the Separation of Concerns principle and isolates business logic from other aspects by implementing Model-View-Controller pattern. The pattern mechanism is illustrated in the Figure 3.5 - (1) a client sends a HTTP request to the web server where it is parsed by a Router (2), then using routes the request is mapped to a controller action (3). Then the controller uses specific models to get the asked data (4). Models send queries to the database and return the requested data to the controller (5) that instructs specific view how to display the data (7). Finally, the response is sent to the client by the view (8).
3.3.5 Data analysis

Deduction of conclusions and recommendations for future research were based on the synthesis. The process of synthesizing the data collection included the following:

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

The data analysis of patterns incorporated three different methods: meta analysis for RQ1 and RQ3, cluster analysis based on graph model for RQ2, and data classification for RQ4. Firstly, meta analysis relied on a qualitative coding schema with dichotomous parameters. In case of RQ1, if a pattern contained a structured template, it was coded as 1, otherwise as 0. As for the specific templates, any contained paragraph in a template was coded as 1, otherwise 0. Any mentions of pattern application domain, resulted coding specified option as 1, otherwise as 0. The data was further analyzed using descriptive statistics. Secondly, graph-based model for cluster analysis aimed in establishing what groups of patterns were most influential. Thirdly, the goal of taxonomical organization of design patterns was to create a topical classification that would help learners to find quickly searched pattern without knowing details. However, the main drawback of this method is that it relies on subjective categorization.
4 Results

Chapter 4 focuses on the results and includes four sections dedicated for each of four research question.

4.1 Publications overview

Overall, 815 papers were proceeded - 526 articles from journals, 289 papers were retrieved from digital databases. By the end of the data retrieval, 83 papers were qualified after applying inclusion criteria, out of which 39 were included after applying exclusion criteria (47%). Figure 4.1 presents the distribution of included publications in electronic sources - academic databases and search engines.

![Figure 4.1: Included publications in digital databases/search engines](image)

4.2 Patterns overview

In general, 206 patterns were found in 39 articles written by 95 researchers as listed in Table 4.1. 93% of the patterns have unique names which indicates the possibility that some existing patterns are only rewritten.

<table>
<thead>
<tr>
<th>PATTERNS</th>
<th>AUTHOR(S)</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itinerary, Master-Slave, Meeting, Plan, Ticket, Facilitator, Forwarding, Organized group, Locker, Messenger</td>
<td>Aridor and Lange</td>
<td>1998</td>
</tr>
<tr>
<td>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</td>
<td>Kendall et al.</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Design patterns for MAS (chronological order)
<table>
<thead>
<tr>
<th>Patterns</th>
<th>Author(s)</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Mobility, Itinerary, Star-shaped Movement, Branching, Contract net protocol, Cooperation Protocol Pattern, Direct Interaction, Mediation, Dispatching</td>
<td>Tahara et al.</td>
<td>1999</td>
</tr>
<tr>
<td>Direct Coupling Pattern, Proxy Agent Pattern, Communication Sessions Pattern, Badges, Event Dispatcher Pattern</td>
<td>Deugo et al.</td>
<td></td>
</tr>
<tr>
<td>Sentinel Agent Behavior Pattern</td>
<td>Chacón et al.</td>
<td>2000</td>
</tr>
<tr>
<td>Receptionist, Session Pattern, Secretary, Mobile Session, Antenna, Private Session, Meeting with Moderator</td>
<td>Silva et al.</td>
<td></td>
</tr>
<tr>
<td>Blackboard, Market Maker, Meeting, Master-Slave, Negotiating Agents</td>
<td>Deugo et al.</td>
<td>2001</td>
</tr>
<tr>
<td>Synchronizer, Environment Mediated Communication, Updating shared state, Behavior-based Decision</td>
<td>Schelthout et al.</td>
<td>2002</td>
</tr>
<tr>
<td>InterRap, Contract net protocol</td>
<td>Lind</td>
<td></td>
</tr>
<tr>
<td>Agency Guard, Agent Authenticator, Sandbox, Access Controller</td>
<td>Mouratidis et al.</td>
<td>2003</td>
</tr>
<tr>
<td>Role Agent Pattern</td>
<td>Cabri et al.</td>
<td></td>
</tr>
<tr>
<td>Monitor, Broker, Matchmaker, Mediator, Embassy, Booking, Call-For-Proposal, Bidding</td>
<td>Kolp et al.</td>
<td></td>
</tr>
<tr>
<td>Reflective Blackboard</td>
<td>Silva et al.</td>
<td></td>
</tr>
<tr>
<td>Organisation schemes, Protocols, Marks, Influences, BDI architecture, Vertical architecture, Horizontial architecture, Recursive architecture, Iniquity, Discretisation, Physical entity</td>
<td>Sauvage</td>
<td>2004</td>
</tr>
<tr>
<td>Hybrid Recogniser, Sense-and-Infer, Assisted Sense and Infer, Ecological Recogniser, Assisted Ecological Recogniser, Clairvoyant</td>
<td>Heinze</td>
<td></td>
</tr>
<tr>
<td>Market-organiser, Agent-side comparison-shopping, Server-side comparison-shopping pattern, Itinerary-balancing pattern</td>
<td>Nguyen and Yang</td>
<td></td>
</tr>
<tr>
<td>Pipeline</td>
<td>Gonzalez-Palacios and Luck</td>
<td></td>
</tr>
<tr>
<td>Agent Society, Agent as Delegate, Agent as Mediator, Common Vocabulary, User Agent, Task Agent, Resource Agent</td>
<td>Weiss</td>
<td></td>
</tr>
<tr>
<td>Patterns</td>
<td>Author(s)</td>
<td>Year</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Rule Composer, Decision Tree</td>
<td>Taylor and Evans-Greenwood, Müller et al.</td>
<td></td>
</tr>
<tr>
<td>Service Client Pattern, Service Representative</td>
<td>Mohamad et al.</td>
<td>2006</td>
</tr>
<tr>
<td>Structure-in-5, Pyramid, Chain of values, The matrix, Bidding, Joint venture, Arm’s-length, Hierarchical contracting, Co-optation</td>
<td>Kolp et al.</td>
<td></td>
</tr>
<tr>
<td>Define Actors, Refine Actor Goals, Means-End Analysis, Contribution Analysis, AND/OR Decomposition, Ask for Help, Refine Models, System-to-Be</td>
<td>Mouratidis and Weiss</td>
<td></td>
</tr>
<tr>
<td>Replication, Collective Sort, Evaporation, Aggregation, Diffusion</td>
<td>Gardelli et al.</td>
<td>2007</td>
</tr>
<tr>
<td>Behaviour Helper Pattern</td>
<td>Charan Ojha et al.</td>
<td></td>
</tr>
<tr>
<td>Gradient Fields, Market-Based Control</td>
<td>De Wolf and Holvoet</td>
<td></td>
</tr>
<tr>
<td>Agent Interface, SCADA control sequence agent, Agent optimization, Agent NET-MAP pattern</td>
<td>Eichelkraut and Etzkorn</td>
<td></td>
</tr>
<tr>
<td>Virtual Environment, Situated Agent, Selective Perception, Roles &amp; Situated Commitments, Protocol-based Communication</td>
<td>Weyns</td>
<td>2009</td>
</tr>
<tr>
<td>Perception Memory Pattern, Exponential Growth Pattern</td>
<td>Klügl and Karlsson</td>
<td></td>
</tr>
<tr>
<td>Disciplined flood, Propertinerary, Smart message, Delegate ant MAS, Delegate MAS</td>
<td>Holvoet et al.</td>
<td>2010</td>
</tr>
<tr>
<td>Selection of Relevant Source Material Policy-based</td>
<td>Cheah et al.</td>
<td></td>
</tr>
<tr>
<td>Scheduler Scramble, Context and Projection Hierarchy, Diffuser, Strategy, Logo World, Learning, Model-View-Controller, Double Buffer</td>
<td>Guo et al.</td>
<td>2011</td>
</tr>
<tr>
<td>Aggregation, Spreading, Gossip</td>
<td>North and Macal</td>
<td></td>
</tr>
<tr>
<td>Composite DelegateMAS</td>
<td>Fernandez-Marquez et al.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cruz Torres et al.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.2 depicts efforts in documenting design patterns over the years. Most of the papers and identified design patterns are from years: 1998, 1999, 2004 and 2010. There are no selected publications from 2008.

<table>
<thead>
<tr>
<th></th>
<th>Publications</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.79</td>
<td>14.71</td>
</tr>
<tr>
<td>Median</td>
<td>3</td>
<td>12.50</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1.25</td>
<td>9.92</td>
</tr>
<tr>
<td>Variance</td>
<td>1.57</td>
<td>98.37</td>
</tr>
<tr>
<td>Minimum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>Total</td>
<td>39</td>
<td>206</td>
</tr>
</tbody>
</table>

Figure 4.2: Publications and patterns per year

4.3 RQ1: How are the patterns documented and what pattern templates are used?

The state of the documentation of the patterns that would indicate the need of standardization was derived from the pattern template details (F12). Overall, 69% of all patterns are described using structured templates, out of which 53% contains some graphical descriptions as diagrams. 73% of visual explanations are modeled using UML. Six is the average number of paragraphs in a template excluding name (assuming that every template contains this element).

102 unique paragraphs were identified. This clearly indicates that there are ambiguities in pattern templates. As pointers to other patterns paragraphs with names: Related patterns, Associated patterns, See also, References, represent one thing, so that they can be grouped without further analysis. However, there are many paragraphs that can have the same semantic meaning, but their grouping is not obvious as: Context/Applicability, Problem/Intent/Purpose, Examples/Known Uses or Participants/Entities.
Among most popular paragraphs, as illustrated in Figure 4.3, are: Solution, Problem, Forces, Context and Consequences. Those paragraphs should be essential while documenting, especially Problem, Context and Solution, because a design pattern follows a *three-part rule* (Alexander, 1979). Least popular paragraphs include: Technical Issues, Reasoning capabilities, Temporal context or Configuration.

### 4.4 RQ2: How are the design patterns connected?

To answer this question and to create the graph, we used data extracted from items: Pattern associations (F11), Design pattern alias (F7) and Design pattern name (F6). A cluster analysis is used to establish what groups of patterns are most influential among patterns. Figure 4.4 presents a directed graph (114 nodes, 137 edges, modularity: 0.771) illustrating associations between found patterns used for the analysis. Three main clusters can be distinguished.

First cluster contains patterns that were inspired by object-oriented patterns from GoF catalog (Gamma et al., 1995), as Proxy, Mediator and by concurrency patterns (Schmidt et al., 2000) as Broker, Active Object. Those patterns mainly concern low-level issues and are found in early papers: Tahara et al. (1999); Kendall et al. (1998); Hayden et al. (1999); Schelfhout et al. (2002).

Second cluster is built around bio-inspired concepts as ants, pheromones or stigmergy. It includes recent patterns concerning self-organization and adaptive behavior imitating mechanisms found in the nature: Gardelli et al. (2007); Fernandez-Marquez et al. (2011); De Wolf and Holvoet (2007); Holvoet et al. (2010); Guo et al. (2011); Cruz Torres et al. (2011); Sauvage (2004).

Third cluster covers patterns referencing to mobile agents and Aglets patterns (Aridor and Lange, 1998), for example Messenger, Itinerary. It contains patterns from: Silva et al. (2000); Deugo et al. (2001); Nguyen and Yang (2004); Holvoet et al. (2010).
Figure 4.4: Pattern space

Other groups are built, for example around FIPA protocols (Chella et al., 2010; Lind, 2003; Tahara et al., 1999) or are closed clusters and were not included in the graph. A closed cluster includes patterns only from one catalog and contains only relationships within the catalog. It indicates unextended pattern language that was not connected to existing pattern space. Those patterns are mostly domain specific - applying to military simulations, e-commerce, intelligent manufacturing, security (Heinze, 2004; Nguyen and Yang, 2004; Shu and Norrie, 1999; Eichelkraut and Etzkorn, 2007; Mouratidis et al., 2003).
4.5 RQ3: For what types of systems have the design patterns been applied?

Data items: Short pattern description (F9) and Pattern application domain (F10) were used to extract the data and to answer the third question. To sum up, there is not dominant type of system for which MAS design patterns have been used. As depicted in Figure 4.5: 59% of the patterns can be applied in every field, 41% of the patterns are focused on more specific uses.

![Pattern application domains](image)

Figure 4.5: Pattern application domains

For industrial applications, main subdomains are: process control and manufacturing (6%) and traffic and transportation (5%), for commercial applications: information management (4%) and electronic commerce (3%). No patterns were identified in correlation with entertainment (games, interactive theater and cinema) domain. Nevertheless, there is a balance and diversity of application domains for MAS design patterns revealing high potential and attractiveness of MAS patterns for software practitioners.
4.6 RQ4: What dimensions of the design patterns exist?

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 (Aridor and Lange, 1998; Schelfthout et al., 2002; Chella et al., 2010; Kendall et al., 1998; Tahara et al., 1999; Oluyomi, 2006).

![Design Patterns for Multi-Agent Systems](image)

Figure 4.6: MAS patterns dimensions

The data for the classification was derived from Catalog pattern categories (F8) and Short pattern description (F9). Based on the analysis of the data, four dimensions of patterns for multi-agent systems were identified: inspiration, abstraction,
focus, and granularity. Figure 4.6 shows a graphical overview of the dimensions, illustrated with example patterns.

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 (De Wolf and Holvoet, 2007) that is inspired by the fields in nature, and delegate ant MAS (Holvoet et al., 2010) that is inspired by behavior of social insects. Examples of society-inspired patterns are emergent society (Kendall et al., 1998) and holonic society (Chella et al., 2010) that get their inspiration from the way societies emerge and structure themselves. Examples of human-inspired patterns are receptionist (Silva et al., 2000) and gossip (Fernandez-Marquez et al., 2011). Finally, locker (Aridor and Lange, 1998) and blackboard (Deugo et al., 1999) are example of patterns that get there inspiration from artifacts in our environment.

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 (Mouratidis and Weiss, 2006) and pyramid (Kolp et al., 2006) are examples of patterns that are useful in early phases in the life cycle, while broker (Hayden et al., 1999) and synchronizer (Schelfhout et al., 2002) can be applied in detailed design and the implementation phase.

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 (Silva et al., 2003) and joint venture (Kolp et al., 2006), 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 (Deugo et al., 1999), negotiating agents (Deugo et al., 1999), and conversation (Kendall et al., 1998). These patterns provide different approaches to support interactions among agents.

Granularity

Finally, the granularity dimension refers to the scope of the patterns, i.e., the system or parts of the system. Hierarchical contracting (Kolp et al., 2006) and emergent society (Kendall et al., 1998) are examples of patterns that apply to a multi-agent system as a whole. Messenger (Aridor and Lange, 1998) and virtual environment (Weyns, 2009) are patterns that apply to parts of a multi-agent system. Intention (Kendall et al., 1998) and InteRRaP (Lind, 2003) are patterns that support the design of individual agents.
5 Discussion and conclusions

Chapter 5 ends the thesis with a discussion on the method, results, guidelines and conclusions directing to the future research.

5.1 Threats to validity and limitations

The main limits of the study lie in the number and quality of search engines and digital databases, so that 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, pilot searchers were performed to tune the search criteria. Furthermore, manual searches for the journal articles were performed. Theses and technical reports were omitted as it was assumed that the patterns would eventually be published in journals or conference proceedings. The time frame of searching was limited to the period 1998-2011. This is motivated by the fact that the Agents conference started around 1998. Before that, documented design patterns for MAS could not be found.

Among reliability procedures was making sure if there were no mistakes while collecting the data in the database. To ensure the validity and strengthen reliability of the research, various strategies were followed:

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

5.2 Discussion on the results

The objective of this literature study was to summarize existing design patterns for Multi-agent systems. To that end a systematic literature review aiming to answer four research questions was performed.

The first research question (subsection: 4.3) 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, it was 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. It is suggested that the pragmatic base for the pattern scheme should contain: Name, Problem, Solution, Context, Forces, Consequences, 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 (subsection: 4.4) was to investigate how the design patterns for MAS are connected. It was observed that there is a strong coupling between patterns from the same catalog. However, more effort should be put in documenting associations between patterns. Moreover, duplicated efforts in describing the same patterns should be avoided.

The third research question (subsection: 4.5) 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 (subsection: 4.6) 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, four dimensions that allows to classify patterns were identified. 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.

5.3 Future research

To conclude, future research directions should include:

1. Formalization and standardization:
   eliminate ambiguity, documentation must be more rigorous and precise;

2. Empirical evaluation of the patterns is required to demonstrate the added value of the patterns (as well as their tradeoffs);

3. Building bridges: creating catalogs and pattern languages will help to better connect patterns, and enhance their combined use,
   generalizing domain-specific patterns, linking other disciplines;

Other opportunities for future research could be to report MAS anti-patterns - design patterns that were unsuccessful (Brown et al., 1998), to evaluate patterns using frameworks for evaluating design patterns (Petter et al., 2010), to evaluate in the context of MAS patterns formal specifications of design patterns that are intended to complement textual descriptions, eliminate ambiguity and facilitate automation (Khwaja and Alshayeb, 2011; Taibi and Check Ling Ngo, 2003) or to create more CASE tools allowing to graphically generate agents from patterns as Chella et al. (2010).
Bibliography


