Simulation Modelling Practice and Theory 000 (2018) 1–14

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Multi-level agent-based simulations: Four design patterns

Philippe Mathieu[a], Gildas Morvan[b,*], Sebastien Picault[a,c]

[a] Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL (équipe SMAC) Centre de Recherche en Informatique Signal et Automatique de Lille, Lille F-59000, France
[b] Univ. Artois, EA 3926, Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A), Béthune, France
[c] BIOEPAR, INRA, Oniris, La Chantrerie, Nantes 44307, France

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Agent-based simulation
Multi-level
Design patterns
Architecture

ABSTRACT

This paper describes four design patterns that aim at systematizing and simplifying the modelling and the implementation of multi-level agent-based simulations. Such simulations are meant to handle entities belonging to different, yet coupled, abstractions or organization levels. The patterns we propose are based on minimal typical situations drawn from the literature. For each pattern, we present use cases, associated data structures and algorithms. For genericity purposes, these patterns rely on a unified description of the capabilities for action and change of the agents. Thus, we propose a precise conceptual and operational framework for the designers of multi-level simulations.

1. Introduction

Multi-Agent Systems (MAS) are intrinsically built as two-level systems: the "microscopic" level, where the agents are endowed with a specific behaviour, and the "macroscopic" level, where the system is seen as a whole. But often this latter level remains *implicit* in the sense that it is "outside" the agents (heterogeneous to them): it is of a *different nature* (set of specifications or issues to be resolved before the design of the MAS, set of descriptors aggregated during or after the functioning of the MAS), and when it exerts a *feedback* on the behaviour of the agents, it is often only in an *implicit* way.

Making the link between the microscopic and macroscopic levels explicit remains an open question [1].

Not only do we not have at the present time any general method to answer it, but the scientific questions underlying this issue are also still rather widely ignored. However, in the current period, the use of MAS is changing and so are the corresponding research subjects. From disciplines such as ethology and sociology, which required relatively small numbers of agents (up to hundreds or thousands), new applications of MAS are now moving to large-scale systems (ecology, molecular biology or cellular, social networks, financial markets, transportation, etc.) where classical approaches encounter many limitations. Issues related to the organization of systems, and specifically their organization into subsystems, take precedence over those for individual decision architectures.

Those new needs are responsible for the development of *Multi-Level Agent-Based Simulations* (MLABS), i.e. MAS which try to provide an explicit representation of the macroscopic level and to each relevant intermediary level, possibly as new agents. This can be done through many forms to address very diverse objectives, such as: introducing in the model abstraction levels which are "useful" or "relevant" to experts in the field [2]; dynamically adaptating the level of detail of the simulation, for instance to save computation time when possible [3]; coupling heterogeneous models to simulate processes

* Corresponding author.
E-mail addresses: philippe.mathieu@univ-lille.fr (P. Mathieu), gildas.morvan@univ-artois.fr (G. Morvan), sebastien.picault@univ-lille.fr (S. Picault).

https://doi.org/10.1016/j.simpat.2017.12.015
1569-190X/

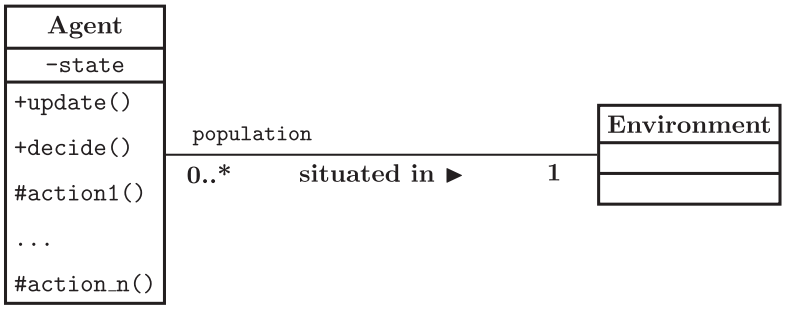

Fig. 1. Elementary structure of an agent and its relationship to the environment.

at different scales [4–6]; or even detecting interesting collective behaviours or emerging spatial structures, either with, or without, reification (agentification) [7].

As observed by [8], there is currently no precise or unambiguous characterization of what is covered by the "multi-level" concept, either in terms of objectives, or system structure, or agent architecture, or algorithms. At best, recurrent problems can be identified, for which many implementations are proposed.

Our goal here is to show that multi-level agent-based simulations involve the combination of elementary situations, which can be characterized by a small number of general criteria, and for which recurrent implementation solutions can be proposed. In other words, we seek to identify and describe *Design Patterns* for multi-level agent-based simulations, in the same sense as in Software Engineering [9].

This paper is organized as follows: in the next section, we present several operational prerequisites on which we rely, and describe our method for classifying concrete problems and identifying patterns. Then, we provide a detailed description of each pattern (name, features, algorithms, use cases) and show how they can be composed in a single simulation. Finally, we discuss methodological, theoretical or practical issues before concluding.

2. Proposed approach

The approach we advocate here consists in characterizing recurrent situations observed in practice in MLABS, in order to build a typology, with the purpose of determining which answers are the most appropriate for each kind of needs. To do so, it is necessary to settle in advance some minimal assumptions regarding our conceptual and operational tools, namely agents and simulation engines. Any multi-agent simulation that supports these criteria should be able to implement the patterns we propose here without any other prerequisites.

2.1. Operational requirements

What we present here is not intended to provide a final definition of an agent in a MAS, or even just in a multi-agent simulation. Instead, our purpose is to resort to the *parsimony principle* (Occam's razor) in order to identify, within multi-agent simulations, a set of features which can be a common foundation for building consistent design patterns in this field.

Then, in what follows, an agent is considered as an entity situated in an environment, with a state (not directly reachable) affected by its actions and the actions of other agents, and which furthermore may be subject to a dynamic evolution of its own (*thereafter $s_t(a)$ denotes the state of agent a at time t*). An agent has several primitives, reflecting its elementary (atomic) capabilities of perception and action. To trigger appropriate changes in its state over time, the agent provides *at most one* public entry point (method, procedure, function...), which we call `update` by convention. Similarly, to trigger action selection, i.e. the choice of primitive actions as a response to its state and its perceptions, the agent provides *at most one* public entry point which we call `decide`. The aim of this entry point is to perform the classical perception-decision-action sequence. By "public entry point", we refer to platform-dependent software control mechanisms, which cannot in general be reduced to language-dependent features such as the well-known `public` vs. `private` Java keywords.

We also focus on situations, quite prevalent even in the MLABS literature, where agents belong to *only one environment* (Fig. 1), without any further assumption regarding the nature of that environment or its implementation (which can also follow design patterns, as shown in [10]). Alternative situations, allowing agents to belong to several environments at the same time, are addressed in e.g. [11–13].

The action primitives of the agents are not intended to be executed directly by the simulation engine, but their execution must occur during the interactions between agents. Thus, we can assume that their access is only allowed for the effective realization of the behaviours of agents.

The autonomy of an agent essentially lies in the fact that the only public entry points are `update` and `decide`. This ensures that there is no other way to manipulate the state of the agent, or distort its perceptions, or cause its actions. In addition, these two entry points should be managed by a *scheduler* which is the only authority to determine when the

Please cite this article as: P. Mathieu et al., Multi-level agent-based simulations: Four design patterns, Simulation Modelling Practice and Theory (2018), https://doi.org/10.1016/j.simpat.2017.12.015