Comput Math Organ Theory (2012) 18:91–112
DOI 10.1007/s10588-012-9110-5

SI:EPISTEMOLOGICAL PERSPECTIVES SIMULATION

Metamodels for role-driven agent-based modelling

Rubén Fuentes-Fernández · Samer Hassan · Juan Pavón · José M. Galán · Adolfo López-Paredes

Published online: 14 March 2012

Abstract A major challenge in agent-based modelling is the management of the process to generate executable simulations from the initial conceptual models. This process is complex and usually involves several roles, which may raise communication problems due to the diverse backgrounds and perspectives of participants and the use of non-explicit knowledge. This situation demands a clear separation and precise definition of the multiple aspects of the process, in order to facilitate their understanding, grasp their relationships and develop them. This paper addresses this goal with a fine-step refinement process for information based on the use of domain-specific languages. It considers analysis contexts that include a particular theoretical framework, domain, type of problem and target platform. For a given context, the process formally defines modelling languages conceptually close to the different aspects relevant to it. It also defines mappings between concepts in those languages. Researchers develop simulations by specifying models with the languages, and share and refine information by using mappings between these models. This infrastructure provides

R. Fuentes-Fernández (✉) · S. Hassan · J. Pavón
Dept. Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Madrid, Spain
e-mail: ruben@fdi.ucm.es
url: http://grasia.fdi.ucm.es

S. Hassan
e-mail: samer@fdi.ucm.es

J. Pavón
e-mail: jpavon@fdi.ucm.es

J.M. Galán
INSISOC, Universidad de Burgos, Burgos, Spain
e-mail: jmgalan@ubu.es

A. López-Paredes
INSISOC, Universidad de Valladolid, Valladolid, Spain
e-mail: adolfo@insisoc.org

guidance throughout the process and makes the information involved explicit. A case study of continuous double auctions illustrates the approach.

Keywords Agent-based modelling · Metamodelling · Domain specific language · Transformation · Social simulation · Double auction · Auction

1 Introduction

Agent-Based Modelling (ABM) has become a mainstream technique in many scientific fields, especially in Social Sciences (Gilbert and Troitzsch 2005). When compared with other approaches (Edmonds 2001), it has two key advantages. Firstly, ABM deals simultaneously with group and individual aspects. This is useful when modelling non-linear complex systems. Secondly, it facilitates complementing conceptual models with computational information. Agent abstractions can be used to model people and also software, following the agent paradigm (Fuentes-Fernández et al. 2009; Weiss 1999). This feature enables gradually adding implementation details to agent models, providing a more seamless transition to simulation models (or code) than other approaches.

Despite its powerful capabilities, the application of ABM is not problem-free. This study is particularly concerned with the proper specification and communication of information in the modelling processes. There are at least three problems here. In the first place, previous studies only share a shallow agreement on the conceptualisation of agents as intentional and social abstractions. See for instance the differences between the already cited papers (Edmonds 2001; Gilbert and Troitzsch 2005). Secondly, the development of large complex models usually requires experts with diverse backgrounds and competences in ABM and other paradigms. According to Galán et al. (2009), while *social researchers* (i.e. domain experts or *thematicians*, and *modellers*) are closer to the domain, *computational experts* (i.e. *computer scientists* and *programmers*) are closer to the target platform. One person can hardly combine all the required expertise and, if several experts play these roles, difficulties in communication and misunderstandings are more likely to arise. Thirdly, processes to articulate ABM modelling offer limited support for this teamwork, as they include hardly any guidance on specific tasks (Drogoul et al. 2003). Moreover, simulation platforms (North et al. 2006) are useful for computational experts, but they do not provide advice on how to model different conceptual abstractions. All these issues frequently make it hard to guarantee that the final models really correspond to the initial requirements (Axtell and Epstein 1994).

To address these problems, our research considers that ABM processes need to make all the information involved in modelling explicit. For this purpose, it distinguishes *simulation contexts* characterised by their domain, the type of problem addressed, applied theory and target platform, among other aspects. Each context has a related *infrastructure* of modelling languages, automated mappings between their concepts, and frequently specific software support tools. The elements of an infrastructure are grouped in bundles to support the work of each role and the interactions between roles. This creates an overall process organized around roles (i.e. *role-driven*).

The infrastructure is developed using techniques from Domain-Specific Languages (DSLs) (Mernik et al. 2005) and Model-Driven Engineering (MDE) (France and Rumpe 2007). It also takes advantage of the similarities between abstract agents in ABM and computational agents in Agent-Oriented Software Engineering (AOSE) (Henderson-Sellers and Giorgini 2005) to reuse part of the infrastructures available in that field.

The development and use of this infrastructure are interleaved processes. When several experts work on a simulation, they specify models conforming to certain languages of the context. They communicate that information to other experts applying mappings that change other models. This process is iterative and incremental, as changes in one stage can call for more information, either closer to the conceptual level or to the simulation. The infrastructure describes explicitly most of the details and assumptions of the modelling process. This allows experts their examination and discussion, and facilitates them to provide feedback that is used to create and adapt the infrastructure. Moreover, though developing this infrastructure requires a great deal of effort, it can be reused for different projects in the context and even partially for different contexts, e.g. when they have the same theoretical background or target simulation platform.

A case study on Continuous Double Auctions (CDAs) illustrates this process. CDAs are often used for real trading and for this reason they have been thoroughly analysed from the experimental and computational point of view (Posada and López-Paredes 2008). They are also complex enough to exemplify the capabilities of our approach.

The rest of the paper further examines the issues introduced here. Section 2 discusses the problems in ABM that our work tries to address and the requirements they impose on the proposed solution. Then, Sects. 3 and 4 give some background on our technological basis, Sect. 3 dealing with metamodels and transformations, and Sect. 4 with AOSE. Section 5 describes the proposed process and Sect. 6 applies it to the case study. This approach is compared with related work in Sect. 7. Finally, Sect. 8 discusses some conclusions and future work.

2 Requirements analysis

This section offers an overview of the ABM process envisioned from our approach. It analyses the requirements of the software support tools, which Table 1 summarises.

We consider *simulation contexts* defined by a theoretical framework, domain, type of problem and target platform. In one of these contexts, experts work with a certain modelling vocabulary and there are some repetitive refinements of information, e.g. adding certain features to a concept or some coding details. This knowledge is captured by the *infrastructure* for a *context* in the form of modelling languages (R1) and mappings (R3) respectively.

The proposed process has two levels. The *simulation level* deals with the development of the models and simulations for the problem in hand. It requires software support tools (R1, R3, R5 and R6) that help experts to save time on these tasks. Ideally, these tools should be close to the domain of the experts to boost their productivity (R5 and R6). The *infrastructure level* develops these tools for modelling

Table 1 Requirements of technological components of the ABM process. Type can be Functional (F) or Non-Functional (NF), and priority can be Must-Have (MH), Should-Have (SH) or Nice-to-Have (NH)

Id.	Type	Description	Prior.	Measurement indicators
R1	F	Modelling languages have a formal definition that can be automatically processed.	MH	Available functionality
R2	F	Modelling languages can be defined incrementally.	SH	Available functionality
R3	F	Mappings have a formal definition that can be automatically processed for models of the modelling languages considered	MH	Available functionality
R4	F	Mappings can be defined incrementally.	SH	Available functionality
R5	F	Customised graphical editors for the modelling languages defined can be developed. These editors support the creation, visualisation and modification of models.	MH	Available functionality
R6	F	Customised graphical editors for mappings can be developed. These editors support the description of mappings from the specification of the source and target models they should relate.	MH	Available functionality
R7	F	The definition of modelling languages and mappings supports several layers of modelling languages connected by mappings.	MH	Available functionality
R8	NF	The development of the customised graphical editors for modelling languages can be semi-automated to a large extent from the definition of these languages.	SH	Available functionality
R9	NF	The development of the customised graphical editors for mappings can be semi-automated to a large extent from the definition of these mappings.	SH	Available functionality
R10	NF	The definition of modelling languages can largely reuse existing definitions.	NH	Available functionality
R11	NF	The technologies and frameworks used in development are well-established and supported by large communities.	SH	Online community > 1000 users

and generation of simulations (R8 and R9), although it does not develop simulation platforms.

When the infrastructures of different contexts are developed using the same technologies, it is possible to have customised *meta-tools* that help to develop the tools used in specific *simulation projects* (R1, R3, R8 and R9). For instance, a meta-editor could process the definition of a modelling language to generate parts of a customised editor for it. Additionally, this enhances the possibilities of reusing fragments of available infrastructures that work on the same domain according to a given school of thought (i.e. the conceptual models), or have the same target simulation platform (i.e. the computational models), to build the new ones (R2, R4, R7 and R10).

Work in a context starts by developing its infrastructure. Then, different projects in that context develop their simulations by using it. Each project consists of the specification of information with models conforming to the modelling languages (R1). Experts use mappings for automated changes to these models and to propagate information, but also as documentation of the modifications allowed (R3). If required, the process can reconsider previous models or add new ones, and develop new trans-

formations (R2 and R4). A project can provide new insights into the use of the infrastructure in a given context (e.g. missing elements or mistakes). In that case, the infrastructure is reviewed to address them (R2, R4, R7, R8, R9 and R10). To explicitly have all the knowledge used in the contexts facilitates this kind of reflection (R1 and R3).

This ABM process is an application of DSLs (Mernik et al. 2005) and MDE (France and Rumpe 2007) to make all the information used in the modelling process explicit and largely automate it. As such, it can benefit from available frameworks in those fields that already address some of the required functions (R11). They commonly use metamodels to define the modelling languages and transformations for the mappings, and provide some core functionalities to process them. In that way, our research can focus on the particular aspects to be considered in ABM.

This choice saves a huge amount of development work but introduces the need for additional roles with respect to Galán et al. (2009). There are also *infrastructure experts* who work at the *infrastructure level* and know about these frameworks. For instance, *social researchers* define hypotheses about traders' behaviour in auctions, *computational experts* describe them with simulation models for a target simulation platform, and *infrastructure experts* develop the graphical editors used by the previous experts. An individual can play more than one of these roles, but this is less likely the more demanding and specialised the roles and the bigger the teams.

3 Metamodels and transformations

MDE (France and Rumpe 2007) is based on formal definitions of modelling languages and mappings between their models, which engineers use with the support of software tools. Engineers specify problems with models and modify and propagate their information applying mappings. Some models and mappings, and less frequently some languages, are developed for the project in hand, but others are reused from previous projects. MDE considers that this approach reduces development work and encourages reusability.

Metamodels constitute the main technique used to specify modelling languages in MDE (García-Magariño et al. 2010). They are intended to define the abstract syntax of graph-oriented modelling languages. These languages regard models as graphs of nodes and relationships with attributes and constraints. Other less common uses include the definition of the notation of modelling languages (García-Magariño et al. 2010). Metamodels are defined using metamodelling languages. The ECore (Moore et al. 2004) language of the Eclipse Modelling Framework (EMF) is probably the most popular of these languages despite its limitations, as it has the fullest tool support.

Transformations automate mappings. They generate models from text (i.e. T2M transformations), models from other models (i.e. M2M), or text from models (i.e. M2T) (Czarnecki and Helsen 2003). Text can be, for instance, documentation or code. The processed models are usually specified with modelling languages defined by metamodels, while text is defined with grammars, templates or implicitly in transformations. The transformations are implemented with tool modules (i.e. coding) or

using specific languages. This latter option is becoming predominant, as it facilitates understanding of the mapping between source and target elements, and prevents one getting stuck in the low-level details of the processing. Nevertheless, the limitations to the functions of the tools to write transformations are an important handicap for their use. The ATLAS Transformation Language (ATL) (Eclipse M2M Project 2011), for M2M transformations of ECore-based languages, and Xtext (Eclipse TMF Project 2011), for M2T and T2M transformations related to grammars, are popular examples of these languages.

The development of transformations itself is usually a manual task, but there are some Model Transformation By-Example (MTBE) approaches (García-Magariño et al. 2009). For M2M transformations, MTBE graphically defines the kind of change with prototype pairs of source and target models, and a tool generates the corresponding transformation in the target language. There are no mature MTBE approaches for M2T and T2M. Different environments support their development using a mixture of modules, templates and specific languages that require manual programming to some extent. This is the case of Eclipse with languages such as Xtext (Eclipse TMF Project 2011) and the INGENIAS Development Kit (IDK) tool (Pavón et al. 2005).

The use of MDE in ABM offers several advantages. Firstly, experts can easily extend metamodels: if their form is not suitable to model a given problem, new elements can be introduced as extensions or specialisations of the existing ones. Such reusability facilitates adapting languages through the modelling process, and introducing intermediate languages between the abstract conceptual ones and the simulation if required to guide model development. Extension mechanisms such as chaining and superimposition are also available for transformations. Secondly, there is a wide range of support software tools that can be reused. In particular, ABM can benefit from graphical tools to define metamodels and generate from them customised model editors, and from reusable automated transformations to transfer information between models.

4 AOSE methodologies and INGENIAS

AOSE methodologies (Henderson-Sellers and Giorgini 2005) are intended for the development of Multi-Agent Systems (MASs). MASs (Weiss 1999) are composed of agents and other computational artefacts. These agents are intentional components, i.e. they are modelled as entities that pursue goals and choose to execute those actions that will potentially help to achieve them. Their choices depend on information about the environment, past experiences, and their capabilities and state. Agents are also social because they interact with other agents to achieve goals. These interactions are modelled in terms of knowledge and requests. Hence, as modelling abstractions, AOSE and ABM agents share important features. Moreover, an increasing number of AOSE methodologies are adopting MDE principles, so they use the same kind of frameworks as our research. Using AOSE methodologies as the basis for our process provides us with a basic infrastructure whose level of abstraction is closer to ABM needs than general frameworks, which saves development and specification work. Among these methodologies, we have chosen INGENIAS (Pavón et al. 2005), whose

modelling language and tools are well-suited to meet the requirements outlined in Sect. 2.

INGENIAS defines its own modelling language with an ECore (Moore et al. 2004) metamodel. It is a comprehensive language for most of the concepts used in AOSE. It includes, among others, primitives for agents and their goals, knowledge, capabilities, interactions and societies. Those referred to in this paper are discussed when first used, but the reader can find their complete description in the paper (Pavón et al. 2005).

The INGENIAS Development Kit (IDK)[1] is an open source software tool that allows the graphical specification of models compliant with the INGENIAS metamodel and the execution of transformations coded as modules. Researchers can also work with INGENIAS models using any infrastructure compatible with ECore, e.g. EMF (Moore et al. 2004), as its metamodel is defined with this language.

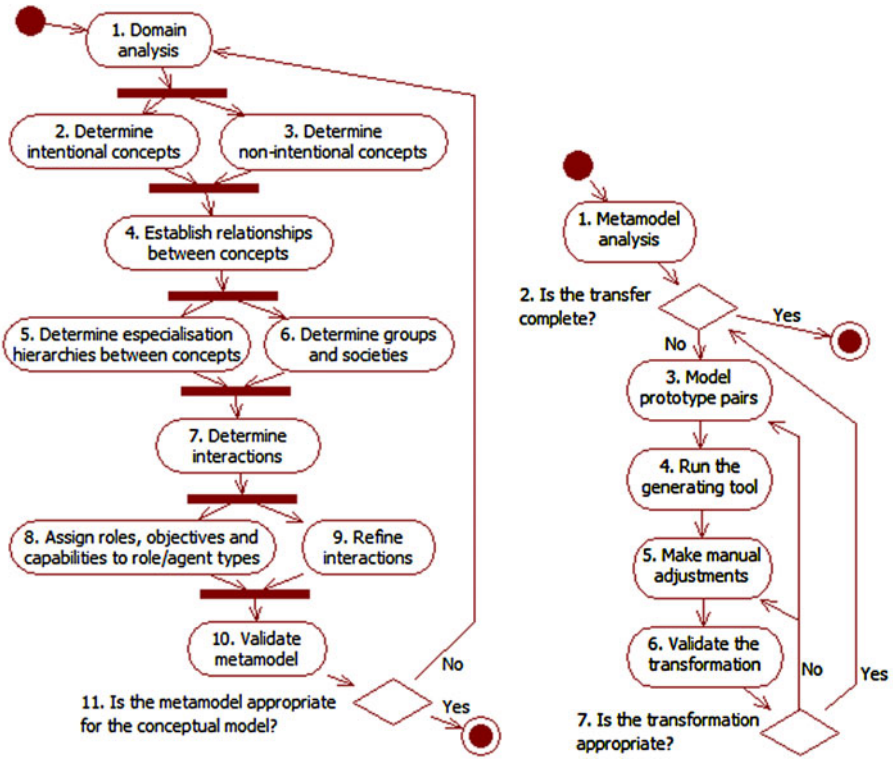
INGENIAS has already been used in ABM (Sansores and Pavón 2005), but without specific adaptations. Given its focus on MAS development, its modelling language and tools present a software bias that makes them unsuitable for social researchers. The purpose of our work is to shift this focus to Social Sciences providing tailored infrastructures that increase social researchers' autonomy when modelling. Infrastructure experts are only required when simulation projects demand low-level software-oriented changes in the INGENIAS language and tools. Examples of such changes could be new modules to analyse or generate reports on models, or to integrate third-party libraries.

5 Modelling process

The modelling process proposed in Sect. 2 differs from common ABM approaches in the use of *infrastructures* specific to *simulation contexts*. These infrastructures include tailored modelling languages defined with metamodels, mappings for their models described with transformations, and tools to manipulate them (e.g. editors, verifiers and animators). When social researchers and computational experts work with an infrastructure, they are guided by the knowledge explicit in the infrastructure and their modelling is made more efficient by the use of tools. These benefits require these experts to cooperate with *infrastructure experts* to develop the infrastructure.

Our adoption of INGENIAS (Pavón et al. 2005) as the basis for our work already provides a meta-editor to generate customised editors from metamodels. The development of metamodels and transformations is still pending. It is considered in Sects. 5.1 and 5.2 respectively and its processes appear in Fig. 1. The discussion of the process for metamodels focuses on conceptual ones, as INGENIAS provides a suitable metamodel for computational experts (Sansores and Pavón 2005). The section finishes with some notes on modelling with the infrastructure in Sect. 5.3.

[1] Available at http://grasia.fdi.ucm.es/.



(a) Development of conceptual metamodels. (b) Development of transformations.

Fig. 1 Activity diagrams of the processes for the development of the infrastructure

5.1 Development of conceptual metamodels

The development of metamodels is a difficult task, which requires us to formalise a conceptual framework that is usually imprecise and complex. This has been acknowledged as one of the main limitations of DSL approaches (Mernik et al. 2005). The current study offers a guideline to perform this task in ABM using the INGENIAS (Pavón et al. 2005) language to define the new DSLs. The INGENIAS metamodel is described with ECore (Moore et al. 2004), so experts can use ECore to modify it and define their own language. However, this process considers it more effective to extend the components of the INGENIAS metamodel, as they already provide part of the semantics required in ABM. Thus, in the following discussion, names in italics correspond to primitives of the INGENIAS metamodel (i.e. metatypes), and the terms *sub-type* and *super-type* refer to metatypes connected by inheritance relationships (INGENIAS extensions of the ECore supertype property). The process is as follows (see Fig. 1(a)):

1. *Domain analysis*. Thematicians consider the concepts required to express their hypotheses and the related information in the group or society to be studied. This task

is related more to the conceptualisation of the problem than to the formalisation of models, so specific support for it is beyond the scope of this study.

2. *Determine intentional concepts*. Thematicians focus their analysis on entities that are (or are conceptualised as) decision makers who act on the environment. Their answers/outputs follow a certain rationale and are linked both to external stimuli and their internal states. They are also proactive, i.e. able to initiate interactions on their own. These concepts are subtypes of the INGENIAS *role* or *agent* metatypes in the new language. The former only specifies behaviour (i.e. what is done) while the latter also describes implementation (i.e. how it is done).
3. *Determine non-intentional concepts*. Elements that do not make decisions are regarded as part of the environment. They are modelled as subtypes of the INGENIAS *environment application*. An environment application allows roles and agents to act on the environment, and notifies them of events in the environment.
4. *Establish relationships between concepts*. Thematicians reflect on the associations that, given a set of concepts, allow to find other concepts and their meaning. These relationships are semantic and are refined later on as types of groups or societies (see activity 6) and interactions (see activity 7).
5. *Determine specialisation hierarchies between concepts*. The concepts identified usually share some features that are highlighted through inheritance hierarchies of metatypes. A super-type contains all the attributes, has all the capabilities, and participates in all the relationships common to its sub-types. Sub-types only modify their own specific features, adding to or constraining the features of the super-type.
6. *Determine groups and societies*. If agents share goals, rules or environment applications, they are brought together in INGENIAS *groups*. If they also share an organisational structure, they constitute a *society*. The metamodel specifies both types as aggregations of their constituents.
7. *Determine interactions*. Agents act on environment applications, receive information from them, and communicate with other agents, exchanging elements of the system, such as physical artefacts or pieces of information. These interconnected activities, aimed at meeting specific goals, constitute *interaction* metatypes.
8. *Assign roles, objectives and capabilities to role/agent types*. A role/agent definition is refined by assigning it *goals* it pursues and *tasks* it is able to carry out. The elements related to the internal state of agents (e.g. goals and information) can be grouped into *mental states*, and all the previous elements can be grouped and assigned together as *roles*. Additionally, this activity establishes the relationships between these types of element. Goals are linked to the tasks able to *satisfy* them. These tasks *produce* artefacts (either physical or information). The presence or absence of artefacts and events (produced by environment applications) is *evidence of* success or failure in the achievement of goals.
9. *Refine interactions*. The refinement of an interaction indicates the agents and environment applications that participate in it, the tasks agents execute, the goals they pursue with that, and the elements produced and consumed throughout. The interaction metatype brings together the metatypes allowed in these relationships.
10. *Validate metamodel*. The metamodel is a refinement of the thematicians' non-formal models for the domain and theory. It should represent them properly, while providing additional details that facilitate the transition to the computational sys-

tem. Thematicians and the other experts review it to ensure that it meets these requirements to the best of their knowledge.

11. *Is the metamodel appropriate for the conceptual model?* According to the results of activity 10, the metamodel is valid for the domain and the process finishes, or it requires further modifications, in which case the process must continue.

Thematicians and modellers are the main roles involved in this process. Thematicians propose the abstract concepts and hypotheses on which modellers ground metamodels. Infrastructure experts can advise them in metamodelling issues.

The process has been described as sequential, but experts do not need to follow this order. For instance, activity 6 may require partially performing activity 7 to identify the relationships of agents and roles with environment applications, and activity 8 to identify their goals. Another example is the possibility of starting with activity 7, if there is a precise idea of the existing interactions, and then using this information to discover the agents of activity 2 and the environment applications of activity 3.

This process focuses on the conceptual metamodel, which is the closest to thematicians. There is at least one other metamodel for computational experts, that of INGENIAS. Experts developing simulations should be able to translate information between models conforming to these metamodels using transformations and additional models. If they need additional intermediate metamodels to guide this refinement, the process of this section can be generalised and made recursive. That is, a metamodel can represent the abstract conceptualisation of a domain in activity 1, from which experts refine new and more platform-oriented metamodels.

Finally, we would remark that this process is a first approach to a guideline to develop metamodels in ABM. A complete process needs to include more detailed activities that provide further advice to experts.

5.2 Transformation development

Transformations automate repetitive tasks of modification and transfer of information in models and text. As mentioned in Sect. 3, there are three types of transformation (i.e. M2M, M2T and T2M), and their development in our work differs.

The development of M2M transformations follows an MTBE approach using the *MTGenerator* tool[2] (García-Magariño et al. 2009). The tool works with ECore (Moore et al. 2004) modelling languages and the M2M transformation language ATL (Eclipse M2M Project 2011). The process is as follows (see Fig. 1(b)):

1. *Metamodel analysis*. Modellers and computer scientists examine the modelling languages involved in the transfer of information.
2. *Is the transfer complete?* Experts consider whether there is any remaining group of concepts in the source language with a standard correspondence in the target language but without a related transformation. If there is, the process continues.
3. *Model prototype pairs*. Experts define the transformation as prototype pairs of source and target models. Source models are examples of the models that should

[2] Available at http://grasia.fdi.ucm.es/.

be accepted by the transformation. Target models represent the result of the application of that transformation. The pair models can share elements to indicate how the transfer of information is. For instance, the name of a person in the source model and an agent in the target model can be the same to indicate that the conceptual person is represented by the computational agent in the simulation. Disjoint sets of pairs are used in activities 4 (i.e. generation) and 6 (i.e. validation).
4. *Run the generating tool*. The MTBE tool takes as input the metamodels of the source and target modelling languages (see Sect. 5.1) and the prototype pairs (see activity 3), and generates as result the transformation in the target language.
5. *Make manual adjustments*. Depending on the algorithm and tool used, the transformation generated may need additional constraints.
6. *Validate the transformation*. Experts use the prototype pairs from activity 3 not used in activity 4 to check that the transformation works as expected. If the transformation is correct, it should be able to process correctly the test prototype pairs, matching their sources against the models and generating the elements indicated in their targets in the resulting models.
7. *Is the transformation appropriate?* Experts need to consider whether the results generated by the transformation are suitable for their goals. For instance, if the simulation model can be generated but does not contain all the expected details for code generation, experts need to modify the prototype pairs from activity 3 and the constraints from activity 5.

Note that though the process focuses on translations between languages, the approach is also applicable to validation. In this case, experts develop transformations whose source pattern corresponds to a test on models and there is no target.

This work adopts the IDK (Pavón et al. 2005) for M2T and T2M transformations. Its application to a simulation platform just requires us to specify the *code template* files that indicate the mappings between types of concepts in models and code.

Developing the transformations in this section requires the collaboration of different experts. Social researchers and computational experts can deal with most of the tasks, but activities like 5 or programming IDK modules also require infrastructure experts.

5.3 Model project

The modelling process itself in our approach is quite similar to that in other ABM studies (see for instance those in the paper, Drogoul et al. 2003). The main steps are:

1. *Is code available and suitable for simulation?* Experts analyse whether the simulation is useful to study the thematicians' hypotheses and whether the results seem to be appropriate. If the code needs modifications or it is not yet available, the process continues.
2. *Refine a model*. Experts add new models or information to the existing ones. Depending on the level of abstraction, different experts perform this task.
3. *Apply transformations*. Transformations can be used to transfer information between models, validate them or produce code.
4. *Back to decision 1*.

The two key differences with other ABM approaches are that specific languages guide and constrain each expert when modelling in activity 2, and transformations partly automate the processing of models in activity 3. Of course, this automation does not prevent the appearance of conceptual mistakes: the hypotheses of the simulation may be wrong, for instance. However, the guide provided by the infrastructure largely reduces the potential mistakes that experts can make in the process. Moreover, automation reduces development work.

This process mainly involves simulation experts. The collaboration of infrastructure experts is only required in two situations. Firstly, when mistakes or new needs are detected regarding the tools to generate the infrastructure. Secondly, when modelling requires the application of the processes in Sects. 5.1 and 5.2, and low-level details appear there, for instance to manually modify generated transformations.

6 Case study: continuous double auction

This paper illustrates the approach with a case study on auctions, in particular CDAs (Posada and López-Paredes 2008), applied to the negotiation of emission permits (Posada 2008). Auctions and CDAs constitute the general framework that determines the domain (thus the conceptual modelling languages and transformations) and emission permits are a particular case described with specific models. The presentation is divided into five sections. Section 6.1 describes the original statement of the problem from the thematicians and modellers' point of view. The next three sections correspond to the application of the stages of the process described in Sect. 5. In order to make the differences between the metamodel and model levels clear, metamodel diagrams enclose metatype names between guillemets (≪ and ≫). Finally, Sect. 6.5 offers additional discussion on the process and results in the case.

6.1 Case context

CDA is one of the institutions most often used for real trading (Posada 2008). It fixes the *Institution* in the triplet IEA (Institution × Environment × Agent's behaviour) that defines any microeconomic system (Smith 1982). This study takes into account the other two dimensions from the classical study of Gode and Sunder (1993a, 1993b), where zero-intelligence agents interact using a single unit per trader. The main features of this CDA, as conceptualised by thematicians and modellers, are described below.

The CDA considers traders who can play the roles of buyers and sellers in it, without these roles being mutually exclusive. In its basic form, all the sellers are endowed with a unit of a good that is indistinguishable from any other; the buyers want to obtain a unit of this good. The decisions of sellers and buyers depend on certain private values that determine their costs and benefits.

The auction is as follows. Any buyer can send a *bid* for a single unit by stating its identity and price. Any buyer can raise this bid. Correspondingly, any seller can *ask* (offer) by stating her/his identity and a price. If asks and bids match or cross, a transaction takes place and both buyer and seller leave the market, cancelling any

unaccepted bids and asks. If a bid and ask do not match but cross, the transaction price is equal to the earlier of the two. After this, the process begins again with the remaining agents. The whole procedure is run for periods of specified duration.

The agents considered in this version are zero-intelligence traders subject to a budget constraint (i.e. *ZI_C agents*) (Gode and Sunder 1993b). This implies that after a certain amount of time, either a buyer or seller randomly submits a bid or an ask. A seller sets an ask price between her/his cost and a maximum value (usually the maximum redemption value). A buyer forms a bid price between the redemption value and 0. These constraints eliminate the possibility of agents having losses.

6.2 Metamodel development

The development of the conceptual metamodel follows the process in Sect. 5.1. Activity 1 is the domain analysis that thematicians and modellers carry out to extract the key information from the description in Sect. 6.1.

Activities 2 and 3 are carried out in parallel to identify concepts relevant to the domain. In this case, there are at least two active elements able to initiate interactions: the traders and the organiser of an auction. Traders initiate interactions to sell or buy goods; organisers represent the institution and perform interactions to set up the auction and rule it. All these elements are modelled as role metatypes, *seller* and *buyer* for traders and *ruler* for organisers. The choice of roles instead of agents is because they specify external behaviour, but not the internal logic or actual implementation. To describe the logic, the model introduces two agent metatypes: *ZI_C agents* (Gode and Sunder 1993b) play the first two roles; the *CDA organiser* plays the *ruler* role.

This part of the metamodel does not consider any environment application. The organisers directly update and publish the information about the state of the auction.

Activity 4 identifies relationships between the previous elements. Following the IEA triplet (Smith 1982), there are relationships between all the traders and organisers in an auction. People can participate in different auctions, or other kinds of market, and there is a need to indicate the specific scenario where they are acting.

Activity 6 identifies potential groups and societies. The traders constitute a group as they share rules of behaviour in auctions. This group metatype is called *agents*. Another group called *auction house* includes the organisers, who have common goals for maximising interactions among traders and exclusive access to the resources used to govern auctions. A society metatype called *auction* groups all these elements.

Modellers may specify inheritance hierarchies of metatypes in activity 5 to make the features they share explicit. For instance, the CDA is a specific type of auction, which in turn is a type of market, so specifications of these scenarios can be incrementally built. Figure 2 shows this hierarchy. Following Smith (1982), the model includes a society metatype *market* with three groups: *institution*, *environment* and *agents*. The traders in a market can play the roles of *seller* or *buyer*. The diagram shows that an *auction* is a particular case of *market* through the *InheritRelationship* between them. This relationship is an INGENIAS extension of the ECore supertype property. The institution of an *auction* is an *auction house*. The members of the *auction house* play the role metatype *ruler* with power to govern the auction. The *CDA* in this case study is an *auction* where participants are *ZI_C agents* and the *CDA organiser* playing the role of *ruler* implements the rules of CDAs.

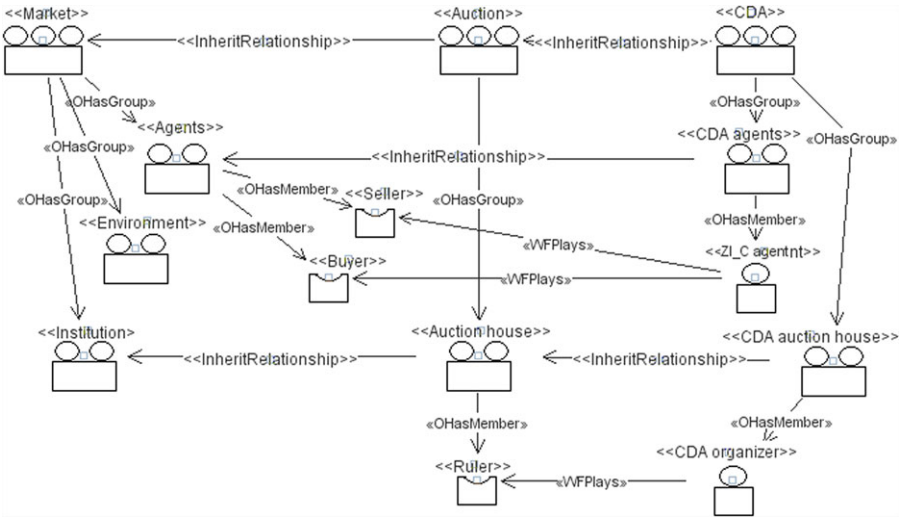


Fig. 2 Partial metamodel for the structural components of the CDA

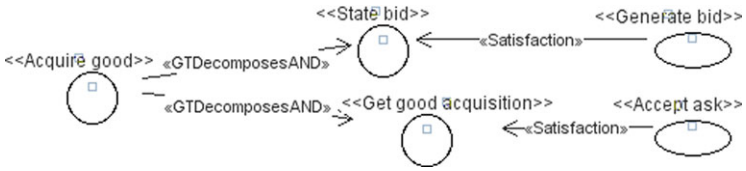


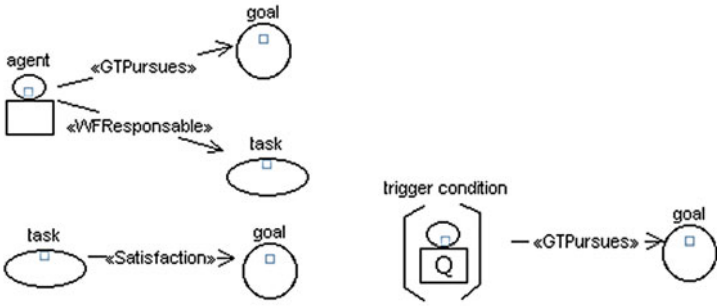
Fig. 3 Goals and tasks for the *buyer* role and their relationships

Activity 7 identifies the relevant interactions for the problem. These interactions are refinements of those first identified in activities 2 and 3. Experts determine that this domain contains at least three basic interaction metatypes: the set up of an auction; joining an auction; participating in an auction. Experts further specify them later.

The refinement of roles and agents implies specifying their goals, roles and capabilities in activity 8. Figure 3 shows some results for the role metatype *buyer*. It pursues the goal *acquire good*, which requires stating bids and accepting some of the asks when they are suitable, according to its internal demand function. These goals are achieved through two types of tasks: *generate bid* and *accept ask* respectively.

The other elements of the metamodel that need refinement are the interactions from activity 7. This is done in activity 9, where experts indicate the tasks the previous role and agent metatypes execute when communicating and the goals they pursue with them. This part of the metamodel is omitted for the sake of brevity.

This part of the process provides the conceptual metamodel for the CDA study. Thematicians and modellers will use the corresponding modelling language to build the models to study different problems in this context.



(a) Source model. (b) Target model.

Fig. 4 Prototype pairs for the transformation of the management of goals

6.3 Transformation development

The development of transformations for a given domain automates repetitive tasks that do not require complex human decisions. Section 5.2 indicates that this study adopts an MTBE approach for M2M transformations.

An example of a possible transformation is for the management of the mental state of agents. Agents are regarded as rational by default, i.e. an agent will try a task if it helps to achieve a still unfulfilled goal. The metamodel of Sect. 6.2 includes subtypes of agents, goals, and capabilities, but not this mental processing. However, INGENIAS agents need to include it explicitly for code generation. This information can be added with a transformation when refining the modellers' models to produce INGENIAS models.

Activity 1 is the analysis of the metamodels involved, both the conceptual one from Sect. 6.2, and that of INGENIAS in Sect. 4 in this case. The answer to the question in step 2 is negative, so experts need to develop a new transformation.

Experts choose in activity 3 the prototype model pairs for the transformation. Figure 4 shows them. The source of the pair contains the elements and relationships that must exist in the INGENIAS model for the transformation being applied: an *agent* has to pursue a *goal* and be responsible for a *task* to attain this *goal*. The target shows the elements to create in the target model when the source model contains the previous configuration. In this case, the transformation adds a new mental state *trigger condition* to the agent, which is true when the *goal* is not fulfilled and is the condition for the *agent* to begin the execution of the *task*. If there are more constraints to trigger the *task*, the pairs of the transformation can include them.

Names in the prototype models are variables used to transfer information from the source to the target models. For instance, the *goal* in the source can match a goal of type *state bid* in the actual models of a given problem (see Fig. 3), and this actual goal will be connected to the instance of *trigger condition* in the target model.

Activity 4 generates the transformation from that pair using the *MTGenerator* tool (García-Magariño et al. 2009). Infrastructure experts manually adjust the transformation in activity 5. As INGENIAS does not have graphical primitives to indicate that the goal has not been fulfilled, experts add this condition as a textual constraint to the mental state.

Activities 6 and 7 check the transformation. As it is a simple one, the check can omit the use of additional prototype pairs for validation.

Note that these models and those in Sect. 6.2 are at different modelling levels. The ones from the previous section are metamodels describing metatypes of components and relationships. The ones in this section are models where elements represent instances of the metatypes in the metamodels previously defined, i.e. they represent types of elements. The types can be used to specify different problems in this context, while metatypes establish the possible categories of these types.

All the resulting transformations can be applied whenever required for different projects. This reduces the need for manual work and the number of potential errors.

6.4 Model project

The last step is the development of the models for a given problem. Both the metamodel and the transformations are general for studies of CDAs that follow the hypotheses outlined in Sect. 6.1. This information is particularised for the specific problem in hand, in this case the analysis of emission permit auctions according to Posada (2008).

The models of this problem include the types of traders and authorities for this market, and instances of them representing particular actors. Thematicians refine these elements for the particular features of the problem. For instance, they include types of *ZI_C agent* that include laws applied in this market, and link their instances according to the geographical region and activity sector of the firms studied.

When the models are complete, thematicians execute the code generation tools to produce the simulation. Then, they run it and study its results. To change the simulation, thematicians just modify their own models and regenerate the simulation. This process is applicable when initial data change, but also to try other hypotheses.

6.5 Discussion

The simple setting of the case study allows us to discuss some differences from the original works of Posada (2008) and Posada and López-Paredes (2008).

The key advantage of the proposed approach is that it describes the common knowledge and expertise about a domain and the migration to a target platform. The use of the conceptual and INGENIAS metamodels obliges modellers to think about the relevant reusable abstractions of the domain instead of the concepts of the specific problem at hand. Thus, it encourages the reuse of knowledge in a wider range of problems. For instance, our conceptual metamodel explicitly represents the interactions between agents in a way that can be extended with new features, such as intermediate agents representing other players in the emission market, while the original emission permits model (Posada 2008) gives only conceptual details, which makes it hard to reuse its simulation beyond the original hypotheses. The use of transformations makes changes in models explicit (France and Rumpe 2007), facilitating the detection of mistakes when compared to traditional codification methods (North et al. 2006).

The use of the MDE *infrastructure* in this *context* also implies saving effort. Experts can easily modify models and transformations to match new requirements. For

instance, new types of agents can be introduced to play the roles of traders. This just implies modelling the new agent, as the rest of the simulation remains exactly the same. Another situation is a change of simulation platform. This just requires that experts create the code templates for that platform if they are not available yet. Reusing the *infrastructure* in different projects also shows up potential problems with it.

Adopting the INGENIAS metamodel facilitates the definition of the new DSL. Its modelling primitives are suitable super-types for the concepts in the case study, which only need to consider their own specific features. Moreover, as INGENIAS models are used for code generation, they include primitives to model relevant parts of the decision-making process of agents. These primitives reduce the need to represent the rationality of agents in ABM with formulae or code, using instead visual representations.

Finally, some decisions on whether to put a concept at the metamodel or model level can be the object of discussion. Only experiments can lead to the better choice, and for this reason it is important to be able to change easily modelling languages.

7 Related work

The approach introduced in this paper is connected with research in modelling processes and languages for ABM. The following subsections study works on these issues.

7.1 Modelling processes

ABM is used to study a variety of phenomena with little in common. The features of the domains, the relevant variables or the aspects of interest are so heterogeneous that it is hard to give useful guidelines beyond a reduced set of problems. Moreover, most available research focuses on introducing particular simulations or general principles, which makes it difficult to talk about complete modelling processes.

Some studies consider the roles involved in ABM projects, their responsibilities, the exchange of information between them, and general principles or advice for modelling. For instance, the study of Drogoul et al. (2003) and its extension by Galán et al. (2009) discuss the roles mentioned in Sect. 1, but do not provide details about how they perform their tasks. Others consider the objectives that guide the tasks of these roles, such as Tesfatsion (2006), or introduce criteria for their correct execution, such as Richiardi et al. (2006) and partially Lorscheid et al. (2011). The ODD (Overview, Design concepts, and Details) protocol (Polhill et al. 2008) can also be considered here. It is in fact a model documentation protocol, but the information it requires provides hints about how social experts should model. Although these works provide useful information to organise and carry out the modelling process, they are not detailed about how to build and use the models.

Some extra support can be obtained for specific tasks in the literature. Again, it usually takes the form of principles or guidelines. For instance, Brenner (2006) gives guidelines for modelling the learning behaviour of agents in economics research. Though the scope of these works is narrow, this kind of information could be incorporated in complete modelling processes to provide effective guidelines for experts.

Our approach helps to fill these gaps providing a process to establish information exchange between roles and to bring together their knowledge. However, it does not propose any particular modelling process, as long as it follows DSL and MDE principles. Other works, such as Lorscheid et al. (2011), can complement it by introducing processes that advice experts on model conceptualisation.

Earlier work mainly considers the perspective of social researchers. In general, the literature pays little attention to computational experts but to simulation platforms such as GAMA (Amouroux et al. 2009), MASON (Luke et al. 2005) or Repast (North et al. 2006). It is assumed that computer-oriented roles must be able to make correct readings of conceptual models and translate them to suitable simulation models. This way of working may cause problems, as it is very difficult to guarantee the correctness of simulation models regarding the hypotheses of social researchers (Axtell and Epstein 1994). A second problem is the replicability of experiments (Wilensky and Rand 2007). Even if models faithfully reflect the initial requirements, the simulation output is also the result of many additional factors. For instance, programmers can make undocumented assumptions or the implementation can bias the choice of equally applicable rules. When the simulation platform automatically generates part of the code, the situation is even worse, because that process is a black box whose detailed implementation is unknown. This is the case, for instance, of AnyLogic (XJ Technologies 2010).

Our MDE approach improves the traceability between the conceptual and simulation models. The models and transformations include all the information to generate the simulations, and experts can examine them. Moreover, the metamodels and transformations are reused in multiple projects, so they can be incrementally developed, tested and fixed in different contexts.

7.2 Languages

ABM uses a variety of languages to document its models. The languages applied largely depend on the roles and the level of abstraction where they are used.

For the conceptual models, mathematics and logic are the most widely used languages. Examples of these are (Brenner 2006; Polhill et al. 2008; Posada 2008; Tesfatsion 2006). While this approach is consistent with the tradition of simulation models (mainly analytical) in Social Sciences, it presents several disadvantages. Firstly, it is hard to transform these equations to computational models. Equations rule out certain aspects of initialisation, constraints in the representation of data types or the discretisation of events and time. There are no simple ways to document the assumptions required to consider these aspects in computational models. Secondly, equations are useful to document simple agent-oriented models, with low levels of heterogeneity and few features to consider. The management of complex models requires the use of mechanisms to describe them at different levels and to encapsulate the description of their parts. The experience of Software Engineering (France and Rumpe 2007) shows that modelling languages are best suited for this purpose.

The literature includes few examples of the use of modelling languages for conceptual models, for instance (Sansores and Pavón 2005; XJ Technologies 2010). Here experts can choose between general-purpose languages and DSLs. The trade-offs

between them are well-known (Mernik et al. 2005). General-purpose languages are widely-known, used and supported, e.g. UML (OMG 2009). As a drawback, they lack specialised primitives to represent domain-specific information, which can make representing it complex or lead to us consider additional assumptions that are not part of the language. Some studies extend UML to provide specialised primitives for modelling, e.g. AnyLogic (XJ Technologies 2010). The problem of these approaches is that they need to stay close to UML to benefit from its wide use, so the additions they can introduce are minimal. On the other hand, DSLs facilitate modelling for a given domain with specialised primitives, but they can be too constrained when new needs appear. The combined use of DSLs and MDE, as proposed in our work, addresses these problems. Experts have standard software tools available that facilitate language adaptation to emerging requirements.

Regarding computational models, the literature does not usually discuss their development except when introducing simulation platforms, e.g. MASON (Luke et al. 2005) or Repast (North et al. 2006). Some of these platforms are currently incorporating limited graphical modelling languages to specify models from which to generate code partially. Examples include AnyLogic (XJ Technologies 2010), which extends UML (OMG 2009), and Repast Symphony (Repast 2008), which uses EMF (Moore et al. 2004). They are oriented to computational abstractions, and not to social experts.

There are also studies focusing on computational models with a perspective closer to ours: they try to provide modelling languages useful for social experts but with a computational meaning, and adopt MDE approaches to generate code. Among them are Sansores and Pavón (2005), based on INGENIAS (Pavón et al. 2005), and ADELFE (Camps et al. 2005), which is another AOSE methodology with a particular focus on emergent behaviours. Our work differs from these in proposing the use of DSLs oriented to thematicians and modellers instead of just adopting AOSE modelling languages. This is expected to improve the autonomy of social experts in the development of their models. Besides, our work proposes the use of declarative transformations and an MTBE approach for their generation. This dramatically decreases the use of program modules for transformations as in Sansores and Pavón (2005), and their manual development with specific languages, as in Camps et al. (2005). It also aims to facilitate the greater involvement of social experts in the generation of the simulation, as it reduces the need to master programming in a simulation platform.

8 Conclusions

This work aims to perform the ABM modelling process as a fine-step refinement of models. This requires the support of linked layers of DSLs whose focus ranges from the social domain to the target simulation platform. AOSE languages are proposed for the intermediate stages in order to facilitate the definition of these DSLs and the transition from them to computational models. This refinement is supported by explicit mappings between groups of concepts in modelling languages. A refinement step applies some of these mappings and manually adds information using models.

Following an MDE approach, the layers of modelling languages are specified with metamodels and the correspondences between them with transformations. The

feasibility of the approach is supported by available technologies, and it has been attempted here with a specific MDE agent-oriented framework, INGENIAS, and MTBE tools. It is illustrated with a case study on auctions, which shows how the approach allows the specification and simulation of social systems with graphical modelling languages.

This approach offers several advantages over standard ABM practices. First, providing languages tailored for specific needs will guide their users in the modelling and reduce the appearance of unintended biases. Moreover, it aids experts in the identification and analysis of social patterns at a macroscopic level, in terms of the atomic elements of the social system's specification. Second, it reduces the conceptual gap between successive models and non-documented changes, which improves the traceability of artefacts. Third, it makes simulation knowledge explicit through metamodels and transformations. This encourages the reusability of previous experience in new projects, and offers the possibility of testing and improving that knowledge with every new development. The INGENIAS metamodel has shown its suitability as the basis for the definition of new ABM languages and its potential for reducing work by reusing already available metamodels. Fourth, the automation of some transformations also reduces the workload of computational experts. It also facilitates the replication of experiments in order to contrast results and strengthens the validity of complex agent-based models.

The effort of learning a new modelling language has still to be evaluated. In principle, a visual language should be easier to use than typical programming, mathematical or logical languages. However, the main issue is the effort required to develop the metamodels for DSLs and their support. In our case study, both the INGENIAS language and software tools easily allow extensions to introduce new concepts and relationships, together with graphical icons for them, and the related transformations.

Despite these issues, we consider this approach to be a step forward in the search for more reliable and transparent agent-based models. The increase in formalisation associated with it, together with its suitability for replication, would answer the common criticism of complex models as obscure black boxes.

References

Amouroux E, Chu T, Boucher A, Drogoul A (2009) GAMA: an environment for implementing and running spatially explicit multi-agent simulations. Lect Notes Comput Sci 5044:359–371

Axtell R, Epstein J (1994) Agent-based modeling: understanding our creations. Bull St Fe Inst 9:28–32

Bernon C, Camps V, Gleizes M, Picard G (2005) Engineering adaptive multi-agent systems: the ADELFE methodology. In: Henderson-Sellers B, Giorgini P (eds) Agent-oriented methodologies. Idea Group Publishing, Hershey, pp 172–202

Brenner T (2006) Agent learning representation: advice on modelling economic learning. Handbook Comput Econ 2:895–947

Czarnecki K, Helsen S (2003) Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA workshop on generative techniques in the context of the model driven architecture, pp 1–17

Drogoul A, Vanbergue D, Meurisse T (2003) Multi-agent based simulation: where are the agents? Lect Notes Comput Sci 2581:1–15

Eclipse M2M Project (2011) The ATLAS transformation language (ATL), v3.2.0. http://www.eclipse.org/atl/. Accessed 1 December 2011

Eclipse TMF Project (2011) Xtext, v2.0.0. http://www.eclipse.org/Xtext/. Accessed 1 December 2011

Edmonds B (2001) The use of models—making MABS actually work. Lect Notes Artif Intell 1979:15–32

France R, Rumpe B (2007) Model-driven development of complex software: a research roadmap. In: 2007 Future of software engineering, proceedings of the 2007 future of software engineering (FOSE 2007). IEEE Comput Soc, Los Alamitos, pp 37–54

Fuentes-Fernández R, Gómez-Sanz J, Pavón J (2009) Requirements elicitation and analysis of multiagent systems using activity theory. IEEE Trans Syst Man Cybern, Part A 39(2):282–298

Galán J, Izquierdo L, Izquierdo S, Santos J, del Olmo R, López-Paredes A, Edmonds B (2009) Errors and artefacts in agent-based modelling. J Artif Soc Soc Simul 12(1):1

García-Magariño I, Rougemaille S, Fuentes-Fernández R, Migeon F, Gleizes M, Gómez-Sanz J (2009) A tool for generating model transformations by-example in multi-agent systems. Adv Soft Comput 55:70–79

García-Magariño I, Fuentes-Fernández R, Gómez-Sanz J (2010) A framework for the definition of metamodels for computer-aided software engineering tools. Inf Softw Technol 52(4):422–435

Gilbert G, Troitzsch K (2005) Simulation for the social scientist. Open University Press, Buckingham

Gode D, Sunder S (1993a) Allocative efficiency of markets with zero-intelligence traders: market as a partial substitute for individual rationality. J Polit Econ 101(1):119–137

Gode D, Sunder S (1993b) Lower bounds for efficiency of surplus extraction in double auctions. In: Friedman D, Rust J (eds) Proceedings of the double auction market: institutions, theories, and evidence, Santa Fe Institute, vol XV, pp 199–219

Henderson-Sellers B, Giorgini P (eds) (2005) Agent-oriented methodologies. Idea Group Publishing, Hershey

Lorscheid I, Heine B, Meyer M (2011) Opening the black box of simulations: increased transparency and effective communication through the systematic design of experiments. Comput Math Organ Theory. doi:10.1007/s10588-011-9097-3, pp 1–41

Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) MASON: a multiagent simulation environment. Simulation 81(7):517–527

Mernik M, Heering J, Sloane A (2005) When and how to develop domain-specific languages. ACM Comput Surv 37(4):316–344

Moore B, Dean D, Gerber A, Wagenknecht G, Vanderheyden P (2004) Eclipse development using the graphical editing framework and the eclipse modeling framework. IBM Redbooks

North M, Collier N, Vos J (2006) Experiences creating three implementations of the repast agent modeling toolkit. ACM Trans Model Comput Simul 16(1):25

OMG (2009) UML, unified modeling language superstructure, version 2.2 (February 2009). http://www.omg.org/spec/UML/. Accessed 1 December 2011

Pavón J, Gómez-Sanz J, Fuentes R (2005) The INGENIAS methodology and tools. In: Henderson-Sellers B, Giorgini P (eds) Agent-oriented methodologies. Idea Group Publishing, Hershey, pp 236–276

Polhill J, Parker D, Brown D, Grimm V (2008) Using the ODD protocol for describing three agent-based social simulation models of land-use change. J Artif Soc Soc Simul 11(2):3

Posada M (2008) Emissions permits auctions: an agent based model analysis. In: Social simulation: technologies, advances and new discoveries, pp 180–191. IGI global

Posada M, López-Paredes A (2008) How to choose the bidding strategy in continuous double auctions: imitation versus take-the-best heuristics. J Artif Soc Soc Simul 11(1):6

Repast (2008) Repast Symphony 1.2. http://repast.sourceforge.net. Accessed 1 December 2011

Richiardi M, Leombruni R, Saam N, Sonnessa M (2006) A common protocol for agent-based social simulation. J Artif Soc Soc Simul 9(1):15

Sansores C, Pavón J (2005) Agent-based simulation replication: a model driven architecture approach. Lect Notes Comput Sci 3789:244–253

Smith V (1982) Microeconomic systems as an experimental science. Am Econ Rev 72(5):923–955

Tesfatsion L (2006) Agent-based computational economics: a constructive approach to economic theory. Handbook Comput Econ 2:831–880

Weiss G (ed) (1999) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press, Cambridge

Wilensky U, Rand W (2007) Making models match: replicating an agent-based model. J Artif Soc Soc Simul 10(4):2

XJ Technologies (2010) AnyLogic 6.5. http://www.xjtek.com/. Accessed 1 December 2011

Rubén Fuentes-Fernández holds a Ph.D. in Computer Science from the Complutense University in Madrid, Spain. He is Associate Professor at this university and member of the GRASIA research group. He has published more than 50 research papers, amongst others in the journals "IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans", "Requirements Engineering", "Computers & Education" and "International Journal of Agent-Oriented Software Engineering". Previously, he worked as a consultant in database systems for four years. His main interests are concerned with the application of Social Sciences to software development, model-driven engineering, agent-oriented methodologies, and social simulation.

Samer Hassan holds a Ph.D. in Artificial Intelligence obtained after his research in Social Simulation in the Universidad Complutense de Madrid (Spain) and the University of Surrey (UK), in which he simulated the postmodernisation process in Spain from a data-driven agent-based modelling approach. He has a multidisciplinary background in Computer Science, Artificial Intelligence and Political Science. His research interests include the potential applications of computational sociology, the modelling of social network communities, and commons-based peer production.

Juan Pavón holds a Ph.D. degree in Computer Science from Universidad Politécnica Madrid (1988). From 1987 to 1997 he was working in R&D departments of Alcatel in Spain, France and Belgium, and in Bellcore (USA), in the development of component-based architectures for distributed systems, and their application to multimedia services on broadband networks and mobile systems. Currently he is Full Professor at Universidad Complutense Madrid, where he leads the GRASIA research group on multi-agent systems, and works on simulation of complex systems, agent-oriented software engineering, and artificial intelligent applications.

José M. Galán is trained in Industrial Engineering from the University of Valladolid. He works as Associate Professor lecturing Economics and Complex Systems at the University of Burgos (Spain). He obtained his Ph.D. from the University of Burgos, supervised by Ricardo del Olmo and by Adolfo López-Paredes of the INSISOC Research Group. He is interested in complex systems, agent-based modelling, game theory, social networks and the use of models in general.

Adolfo López-Paredes is Associate Professor at the University of Valladolid (Spain). He graduated from the University of Oviedo in Industrial Engineering (1994) and obtained his Ph.D. in 2000 on applications of agent-based simulation to economic analysis, in the University of the Basque Country. His research interests within the INSISOC Group ("Engineering Social Systems Group") include project management and computer simulation of social and economic behavior and industrial policy: auctions, financial markets, natural resources management, supply chain management.