Metamodelling for Agent Based Modelling: An Application for Continuous Double Auctions

Rubén Fuentes-Fernández[1], José M. Galán[2], Samer Hassan[1], Juan Pavón[1] and Felix A. Villafañez[3]

[1] GRASIA, Universidad Complutense de Madrid, Spain
{ruben, samer, jpavon}@fdi.ucm.es
[2] INSISOC, Universidad de Burgos, Spain
jmgalan@ubu.es
[3] INSISOC, Universidad de Valladolid, Spain
villafafelix@eis.uva.es

Abstract. Agent-Based Modelling is gaining wider acceptance as a paradigm for social research. However, it still present limitations in the management of the process to generate the simulations from the initial conceptual models. Thus, it is difficult to reuse the knowledge from available models and adapt it to different hypotheses. This paper proposes the use of metamodels in order to define explicitly the core concepts of a problem and to differentiate the aspects involved in the process. A case study for continuous double auctions shows how to define the related metamodel and use it to address alternative situations in these auctions. The case study drives the discussion on the advantages and limitations that metamodelling can bring to social simulation.

1 Introduction

Agent-Based Modelling (ABM) has become over the past years a widely used technique for research in Social Sciences [4]. One of its key advantages lies on its core abstractions for modelling, i.e. agents and their societies. Agents are intentional and social entities, which are capable of rational and complex individual behaviour. As they share these features with humans, they are expected to facilitate the specification of the human target systems. Besides, these abstractions can be refined with concepts closer to the target simulation platforms, bridging the gap between the conceptual models and the simulation ones.

The actual use of agent-based models still has a lot of room for improvement [1]. The situation in which agents constitute general and reusable modelling primitives for ABM, with standardise processes to translate them to simulation models has not been reached. In fact, researchers in ABM have different perspectives on what agents are. Moreover, they do not tend to follow a clear translation of those perspectives in their formal agent models, frequently adopting ad-hoc translations of them in the simulation code.

In order to address these limitations, some researchers [7, 12] have proposed the use of metamodelling techniques. Metamodels define modelling languages, which

describe the modelling primitives available to describe a problem. Researchers create their models instantiating these primitives. If required, they can use extension mechanisms to modify the language in a controlled way, introducing new elements.

This approach has several advantages. Firstly, metamodels can be processed by software tools. This allows, for instance, the building of graphical editors for these models, and to provide automated transformations that partly carry out the propagation of information from abstract formal models to actual simulations. This tool support reduces the probability of making unintended mistakes when modelling and provides the basis for comparison (and thus replication) among works.

The main obstacle to apply this approach is to define suitable metamodels. There is an inherent difficulty in capturing the knowledge to describe complete domains of problems with a formal definition. A metamodel must be rich enough to capture all the variability of a domain, but also to constrain modellers to produce correct models that can be translated to simulation code in a semi-automated way.

Our work attempts to provide guidelines, intermediate languages for metamodelling in ABM and software tools that facilitate the definition of these metamodels. In order to illustrate this approach, this paper discusses the formalisation of the well-known problem of continuous double auctions.

The rest of the paper is organised as follows. Section 2 provides a brief introduction on ABM. Section 3 considers how our approach use metamodels in ABM and Section 4 applies such approach to model auctions. Finally, Section 5 discusses the implications of the process together with some concluding remarks.

2 Related Work

ABM describes models for social analysis using agents as the key abstraction. The complete process entails different stages of designing, implementing and using agent-based models. Initial steps usually begin with non-formal conceptualizations of the target system that successively are refined to shape a formal model that can be computationally implemented. This process is often got around in literature and only few works explore the migration from conceptualisations to formal models.

[12] considers the problems emerging from the usual lack of background in software programming of social researchers and the difficulties to compare similar models implemented over different platforms. They propose adapting the model-driven engineering methodology INGENIAS [10], which focuses on the development of Multi-Agent Systems (MAS). Their approach extends the INGENIAS modelling language with additional primitives for specific problems. It also uses the INGENIAS process to generate the simulation source code. The main limitation of this work is its use of predefined extension mechanisms of the INGENIAS language, with concepts with an inherent software bias, which are not very appropriate for social researchers.

[7] defines a MDE approach for ABM. Its language has as core concept the *Mentat*, an agent skeleton with mental properties that researchers can extend for their models. The main issue of this work is that it is constrained to data-driven simulation, which complicates its application for other kind of problems.

Looking to overcome these limitations, our approach tries to achieve a MDE approach for ABM with two main features: it must be adaptable to different agent-oriented approaches and support alternative uses of models and their reuse.

3 INGENIAS Metamodels for ABM

INGENIAS [10] is a software development methodology for MAS. It adopts a MDE approach with two basic components: a modelling language and software tools.

A metamodel specifies the INGENIAS modelling language. It defines the available concepts and relationships, together with their properties and constraints. As it is aimed for modelling MAS, it includes the concepts of agent and group. An agent is characterised in terms of its goals and the capabilities it has to accomplish them. Groups include agents and external resources, coined environment applications, that agents can use. Agents participate in interactions with other agents to achieve global goals. Models can also include code components to specify low-level details that depend on the platform, such as formulas about preference policies or location algorithms.

The INGENIAS Development Kit (IDK) software tool allows processing models with their modules. Standard modules available with the distribution support model auto-completion for certain tasks, documentation and code generation. Both of them require the availability of templates. A template gives a general description of a primitive available for modelling with slots that are instantiated with information of specific elements in models. For instance, researchers building a simulation for the MASON platform [9] specify a general template for a MASON agent; when generating code, the IDK instantiates that template with the information of every agent in the specification, generating the source code for those specific agents. The IDK allows the development of new modules that access and change its models, so automated transformations can be adapted to the researcher needs.

The approach of this work uses these elements to provide domain-specific modelling languages and tools, with guidelines for their use [8]. It modifies the INGENIAS metamodel in order to include more flexible extension mechanisms. In particular, inheritance relationships are allowed for any concept, so it can be tailored with additional properties and constraints in common models, without changes in the metamodel. Inheritance implies that a concept acquires all the features of its super-concept. Moreover, it adopts a more declarative approach concerning the tool. It specifies transformations with transformation languages instead of code, which provides a definition closer to the artefacts (i.e. metamodels, models, grammars, code...) that these transformations manipulate. Besides, it proposes developing model transformations (i.e. those that only involve models) with a Model Transformation by Example (MTBE) approach. MTBE [3] defines transformations through prototype pairs of source and target models compliant with certain metamodels. A MTBE tool processes these pairs and automatically generates the resulting transformation. The final component of the approach is a process to guide researchers in the application of this framework [8]. The case study chosen here illustrates this to show the advantages and current limitations of the work.

These decisions are expected to increase the autonomy of social researchers. The final goal is that they mostly develop their models autonomously, requiring only the support of engineers to address new and not-considered features of the domain.

4 Case Study: Continuous Double Auction

4.1 Case Study: Continuous Double Auction

In order to illustrate the exposed approach, this paper considers an auction as paradigmatic case in Economics, in particular the Continuous Double Auction (CDA). As CDA is one of the institutions used more often for real trading, it has been thoroughly analyzed from the experimental and computational point of view [11]; it is also complex enough to exemplify the capabilities of metamodelling. CDA fixes the *Institution* in the triplet IEA (Institution x Environment x Agent's behaviour) that defines any market [13]. This study takes into account the other two dimensions from the classical work of Gode and Sunder [5, 6] where zero-intelligence agents interact in using a single unit per trader. The main features of this CDA are described below.

The CDA considers buyers and sellers. 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. Each buyer agent is informed of a redemption value v according to a demand function unknown for the other players. The profit obtained by a buyer that buys the good at a price p is v-p. Similarly, each seller agent is endowed with a cost c for the unit of the good. The profit obtain by a seller that sells the good at price p is p-c.

From the point of view of the institution, the auction runs 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 its identity and 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. In the case that 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 several periods of specified duration.

The agents considered in this version [6] are zero-intelligence traders subject to a budget constraint (ZI-C agents). This implies that after a certain amount of time, either a buyer or seller randomly forms a bid or an ask. A seller forms an ask price between its cost and a maximum value (usually the maximum redemption value). A buyer forms a bid price between its redemption value and 0. These constraints suppress the possibility for agents to have losses.

In the game dynamics, each buyer compares its bid with the current state of the market. If its bid is above the best ask, it accepts the best ask and the transaction occurs, while the state of the market is updated. If the buyer's bid is below the best ask and above the best bid or there is not any ask yet, that bid becomes the current best bid; otherwise, in the case that the buyer's bid is below the best bid and according to the auction rules, the agent does not send the bid. The asking process is analogous in the case of sellers.

4.2 Metamodelling

The process proposed in [8] comprehends two main stages: the identification of the core concepts of the domain and the description of their standard interactions. The metamodel for the problem describes these elements as extensions of the primitives available in INGENIAS [10].

INGENIAS has two main components that can perform tasks: *agents* are proactive entities able to initiate tasks following their own and explicit agenda; *applications* are usually reactive (i.e. used by agents), though they can raise events in case of changes in the environment. Agents that share an application belong to the same *group*, and when they also share knowledge, procedures and rules, to the same *organisation.*

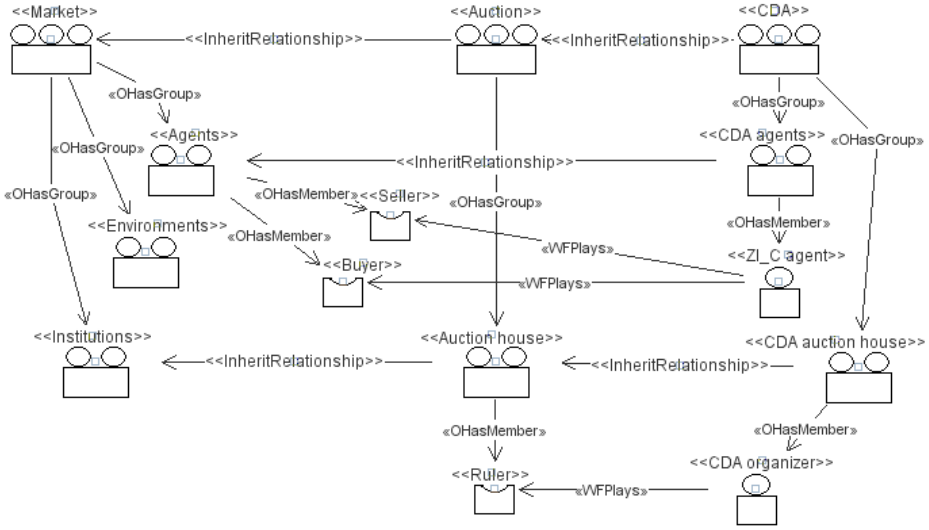

Fig. 1. Partial metamodel for the structural components of the CDA. Stereotypes (i.e. names between guillemets) denote types of entities and relationships.

Fig. 1 illustrates part of the hierarchies of elements proposed for the modelling language of CDA. Following [13], the basis of the hierarchies is the *market* that determines the *institutions*, the *environments* and the *agents*. Agents in market can play two roles, *buyer* and *seller*, which are not exclusive. The *auction* is a particular type of market. The diagram shows this through the *InheritRelationship* between these concepts. The auction is characterised for its institution, the *auction house*. A *ruler* governs this house and implements the common rules for auctions (e.g. turns between buyers and sellers, transactions, availability of goods, or relationships between bids and asks). The most specific market in this hierarchy is the CDA. Following [6], this auction uses a particular type of agents (i.e. *ZI_C agents*) to play the two roles in the group agents of the market. The CDA also adds a specific *CDA organiser* agent to play the ruler role and tailor the institutional behaviour to the norms of CDA.

In this case, there is no need to identify external entities. In order to keep the model simple for this discussion, this metamodel does not explicitly represents the

units of goods. These are represented as attributes of the traders. The internal values of these agents (i.e. the redemption and cost value) are also represented as a unique limit value attribute. These attributes appear in Fig. 3 as the components *good units* and *value limit* of the internal state of the ZI_C agent.

The next step is refining the agents and roles in Fig. 1. This refinement implies defining their goals and the capabilities they have available in order to achieve them. Roles define the general features of these elements, which agents implement with specific policies. Fig. 2 shows them for the buyer role. 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 tasks, *generate bid* and *accept ask* respectively.

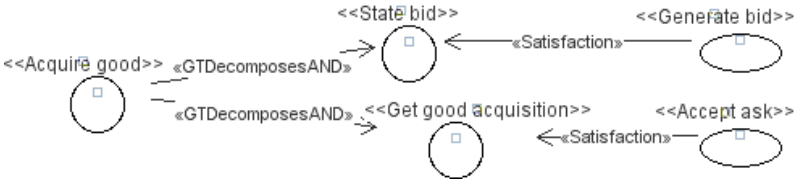

Fig. 2. Goals and tasks for the buyer role and their relationships.

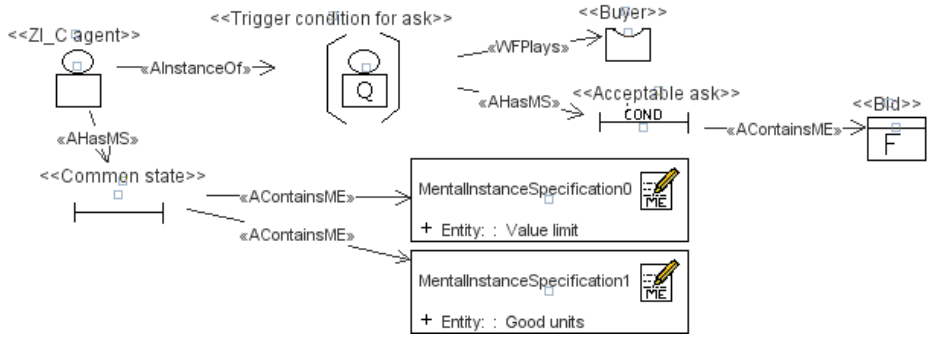

Fig. 3. Mental states for the buyer.

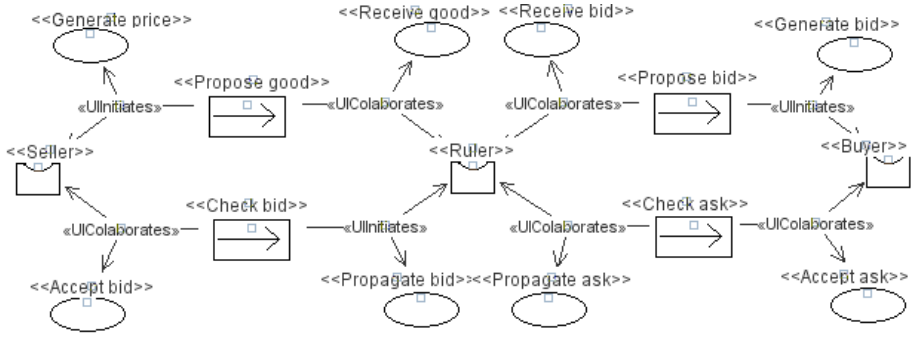

Fig. 4. Metamodel for interactions. Roles execute the activities that trigger the communications represented by the arrows.

Besides specifying the involved goals and tasks, roles need to describe when they initiate the tasks and when the goals are satisfied according to the current system state. Fig. 3 shows an example of these specifications. The *trigger condition for ask* is an instance of the internal state of the ZI_C agent that indicates when the task *accept ask* can be initiated. The condition states that the agent must be playing the role buyer and there must be a bid. The specific condition involving the bid and the limit value of the agent would be recorded in the *acceptable ask*. It can be specified, for instance, as formulae or code. The metamodel could include additional primitives to describe this condition, although it is not advisable: there must be a trade-off between the expressiveness of the language and its value to present a synthesised vision of the problem. Adding primitives to the low-level details would make specifications too verbose, losing their abstraction value.

The final element of the specifications is the interactions describing how agents and roles participate in the global activity of the system. Fig. 4 shows an excerpt involving the previous element and also showing the participation of the ruler.

5 Conclusion and Discussion

This paper has presented the definition of a metamodel that defines a modelling language for CDA. This metamodel illustrates our vision that ABM can benefit from an explicit definition of its modelling languages in several ways: facilitating the specification of problems, their translation to code and the analysis of their results.

Several classical works on CDA have been the basis to identify the key concepts present in this kind of auctions, the attributes that determine their behaviour and their interaction. These elements have been structurally formalised as roles and agents, their goals and capabilities, and groups and organisations. The dynamic behaviour of these elements in auctions has been established through satisfaction relationships between tasks and goals, together with particular conditions, as those used for triggering a task. This metamodel shows the common participants in auctions with the components that researchers need to specify and test different hypotheses. For instance, CDA does not require their agents being zero-intelligence traders; if other traders are going to be tested, researchers may need to add new attributes. Those would participate in conditions or change specific tasks, but the defined roles, interactions and groups would remain unchanged. Besides, the metamodel can define the well-formedness rules for models, allowing the checking of modelling mistakes. For instance, if the model does not include the participation of the buyer proposing bids, it should be revised and corrected.

As stated in sections 1 and 2, the formal models of auctions facilitate applying standardised transformations. Transformations are used to check complex errors, generate documentation, translate them to another modelling language, generate source code, or output result data from models. For instance, a transformation could check that all the traders are going to participate in at least two different auctions if researchers decide so. These transformations are developed and refined over different projects, fixing the potential errors and simplifying the processing.

These advantages are expected to make researchers more autonomous and productive in ABM. These are two necessary conditions to mitigate the main drawbacks of the approach: the high costs of development models and the difficulties to guarantee that the final simulation models are a close translation of the original conceptual models.

References

1. Edmonds, B.: The Use of Models - Making MABS More Informative. In: Moss, S., Davidsson, P, (eds.) Multi-Agent-Based Simulation, LNAI, vol. 1979, pp. 269--282. Springer, Heidelberg (2001).
2. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: 2007 Future of Software Engineering (FOSE 2007), pp. 37--54. IEEE Computer Society (2007).
3. García-Magariño, I., Rougemaille, S,, Fuentes-Fernández, R,, Migeon, F,, Gleizes, M. P.: A Tool for Generating Model Transformations By-Example in Multi-Agent Systems. In: Demazeau, Y,, Pavón, J,, Corchado, J. M., Bajo, J, (eds.) Proceedings of the 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), Advances in Soft Computing, vol. 55, pp. 70--79. Springer, Heidelberg (2009).
4. Gilbert, N,, Troitzsch, K. G.: Simulation for the social scientist. Open University Press, Buckingham, UK (1999).
5. Gode, D. K., Sunder, S.: Allocative Efficiency of Markets with Zero-Intelligence Traders - Market as a Partial Substitute for Individual Rationality. Journal of Political Economy 101(1), 119--137 (1993).
6. Gode, D. K., Sunder, S.: Lower Bounds for Efficiency of Surplus Extraction in Double Auctions. In: Friedman, D., Rust, J. (eds.) The Double Auction Market: Institutions, Theories, and Evidence. Santa Fe Institute Series in the Sciences of the Complexity, Proceedings, vol. XV, pp. 199--219 (1993).
7. Hassan, S., Antunes, L., Pavón, J.: Mentat: A Data-Driven Agent-Based Simulation of Social Values Evolution. Multi-Agent-Based Simulation X. LNCS. In press, 2010.
8. Hassan, S., Fuentes-Fernández, R., Galán, J. M., López-Paredes, A., Pavón, J.: Reducing the Modeling Gap: On the use of metamodels in agent-based simulation. In: 6th Conference of the European Social Simulation Association (ESSA 2009), pp. 1--13(2009).
9. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A Multiagent Simulation Environment. Simulation 81(7), 517--527 (2005).
10. Pavón, J., Gómez-Sanz, J. J., Fuentes, R.: The INGENIAS Methodology and Tools. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies , chapter IX, pp. 236--276. Idea Group Publishing (2005).
11. Posada, M., López-Paredes, A.: How to choose the bidding strategy in continuous double auctions: Imitation versus take-the-best heuristics. Journal of Artificial Societies and Social Simulation 11(1), 6 (2008).
12. Sansores, C., Pavón, J.: Agent-Based Simulation Replication: A Model Driven Architecture Approach. In: Gelbukh, A. F., de Albornoz, A., Terashima-Marín, H. (eds.) MICAI 2005: Advances in Artificial Intelligence. LNCS, vol. 3789, pp. 244--253. Springer, Heidelberg (2005).
13. Smith, V. L.: Microeconomic Systems as an Experimental Science. American Economic Review 72(5), 923--955 (1982).
14. Wilensky, U., Rand, W.: Making Models Match: Replicating an Agent-Based Model. Journal of Artificial Societies and Social Simulation 10(4), 2 (2007).