Metamodelling for Agent-Based Modelling: An Application for Posted Pricing Institutions

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

[1] GRASIA-Universidad Complutense de Madrid,
Departamento de Ingeniería del Software e Inteligencia Artificial,
Facultad de Informática, c/ Profesor José García Santesmases s/n,
Madrid, 28040, Spain,
ruben@fdi.ucm.es, samer@fdi.ucm.es

[2] INSISOC-Universidad de Burgos, Escuela Politécnica Superior,
c/Villadiego s/n,
Burgos, 09001, Spain,
jmgalan@ubu.es

[3] INSISOC-Universidad de Valladolid,
Escuela de Ingenierías Industriales,
Paseo del Cauce 59, Valladolid, 47011, Spain,
villafafelix@yahoo.com

Abstract: Agent-Based Modelling is gaining wider acceptance as a paradigm for social research. However, it still presents limitations in the management of the process to generate the simulations from the initial conceptual models. This makes it difficult to reuse the knowledge from available models and to adapt it to different hypotheses. This paper proposes the use of metamodels in order to define explicitly the core concepts of a problem domain and to differentiate the aspects involved in the process. A case study for posted pricing institutions 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) is a form of computational simulation that has become over the past years a widely used technique for research in very different disciplines such as Biology [20], Resource Management [22] or Political Science [21]. One of its key advantages lies on its core abstractions for modelling, i.e. agents and their societies. Agents are computational, 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, which makes them especially suitable to be used in the Social Sciences [14]. 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.

This modelling approach has been easily adopted because [1, 4, 9]:

- It leads to natural and, simultaneously, formal descriptions of the target systems that can be understood and validated by modellers and stakeholders.
- It enables to easily model heterogeneity and bounded rationality, making possible to abandon assumptions of representative and optimizing behaviours that are non-realistic in many social contexts.
- It facilitates to integrate an explicit representation of the environment and to model local interaction.
- It allows analysing bottom-up and emergent behaviours.
- It makes possible to create interdisciplinary science integrating models from different fields.

In spite of these advantages and the diffusion of the approach, the actual use of agent-based models still has a lot of room for improvement from a computational point of view [7]. The situation in which agents constitute general and reusable modelling primitives for ABM, with standardised processes to translate them to simulation models has not yet been reached. In fact, researchers in ABM have different perspectives on what agents are, which range from simple sets of key-value pairs to complex rational entities with an explicit representation of their knowledge and

reasoning. Moreover, researchers do not tend to follow a clear and explicit translation of those perspectives into their formal agent models and simulation code. On the contrary, they frequently adopt ad-hoc translations whose details are not made available, as they are considered of secondary relevance for research. This state of affairs makes difficult comparing hypotheses and results, and hinders a wider acceptance of ABM [12].

In order to address these limitations, some researchers [18, 31] have proposed the use of metamodelling techniques. Metamodels define modelling languages, which specify the modelling primitives available to describe a problem. Researchers create their models instantiating these primitives, i.e. creating models that comply with the definition of the metamodel. If required, they can use extension mechanisms to modify the language in a controlled way, introducing new elements or modifying the available ones.

This approach based on metamodels has two key advantages. Firstly, metamodels can be processed by software tools. This allows, for instance, developing graphical editors for these models, and providing 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, as tools make some basic checking of the information. Moreover, this support provides the basis for comparison (and thus replication) among works, as refinement and implementation information is available for examination. Secondly, metamodels make explicit the primitives required to model certain classes of problems under a set of given assumptions. This knowledge crystallizes the experience of a community in a domain, facilitates learning to novices, and encourages discussion and reflection on different approaches.

The main obstacle to apply this approach is to define suitable metamodels and the correspondences between them. 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. There must also be clear correspondences between the metamodels of the different languages involved in a problem (e.g. domain, software-modelling and programming languages). This is not only an issue of making semi-automated translations, but making those proper according to the semantics of the involved elements. Some semantic alignment of the concepts in the different languages facilitates this process.

Our work addresses this problem with a framework for metamodelling in ABM. It includes intermediate languages between the abstract formal models and the simulation code, guidelines to define their metamodels and the correspondences between the different involved languages, and software tools that support these tasks. Its modelling process is conceived as a successive refinement of models in different languages, supported by transformations and tailored tools, until generating the simulation. In order to illustrate this approach, this paper discusses the formalisation of the well-known problem of posted pricing institutions.

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 it 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 conceptualization, design, implementation and use of agent-based models. Initial steps usually begin with non-formal conceptualizations of the target system that are successively 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 details of the migration from conceptualisations to formal models, or from those to actual implemented simulations. The few that discuss the modelling process taking into account the different roles and subtasks

[6, 7, 10] are too general to provide specific steps on how to build those models.

Trying to assist the modeller in the implementation stage, there are also a very wide variety of toolkits, each of them with specific features [26]. Some of these toolkits, such as Repast, Mason and Netlogo, are very successful and widely used in the ABM community [30]. However, in most cases, the aid of these platforms is reduced to provide some libraries for programming. Therefore, they do not assist the modeller in key aspects of the "jump" from the abstract model to the actual implementation, as they are only useful for the late design and coding of the simulation. This implies that modellers lack of guide from platforms to make plenty of relevant decisions in previous development stages. This issue is worsen when considering the frequent lack of software engineering notions of many modellers coming from the Social Sciences [24].

This causes two main consequences. First, ABM computational aspects are frequently described insufficiently if not poorly, leaving the appearance of black-box and reducing its reliability [12]. Second, this fact significantly complicates replication. Replication is considered a highly recommended practice, even with statements such as "unreplicated simulation models cannot be trusted" [8]. However, literature rarely offers modellers the tools for facilitating replication.

Interesting works such as the ODD protocol [17] certainly guide modellers in the whole process with some general principles. However, ODD is only a documentation protocol, still insufficient when trying to reduce the gap between the conceptualization and the implementation.

This transition is tackled very well in the field of Agent-Oriented Software Engineering (AOSE) [19], which is devoted to the development of Multi-Agent Systems (MASs). AOSE and ABM share a common foundation in the use of agents and their societies as the basis for the specification of their systems. For this reason, AOSE has been perceived as a suitable development approach for ABM simulations [2]. Many works in AOSE follow a Model-Driven Engineering (MDE) methodology [3, 10], as our approach does. These works also propose the use of intermediate languages between the conceptualization and the implementation of models, which would facilitate the model description and replicability. Despite these common premises, there is a variety of perspectives on how to apply AOSE and MDE to ABM.

There has been research on the application of general AOSE methodologies to build simulations. For instance, the ADELFE methodology [2] aims to develop Adaptive MASs (AMASs). It tackles successfully the dynamics of these systems, but it encourages the use of classical goal-driven agents and is biased towards them. This bias in modelling is common to these methodologies, as they are based on particular models of agency. Besides, the application of works in this line demands an advanced knowledge of AOSE not common in social modellers. These methodologies are targeted to pure software development, so their processes and languages mainly deal with software concepts and are intended for engineers.

Another group of works pursues defining AOSE and MDE methodologies tailored for ABM. Depending on whether their main concerns are about software engineering or ABM, they offer different tradeoffs.

An example of work emerging from software engineering is that of Sansores et al [31]. It considers the problems due to the usual lack of background in software engineering of social modellers and the difficulties to compare similar models implemented over different platforms. They propose adopting in ABM the methodology INGENIAS [27] for the development of MASs. Their approach extends the INGENIAS modelling language with additional primitives for specific domains. 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 again not very appropriate for social modellers.

On the other hand, Hassan et al [18] defines a MDE approach from the perspective of ABM. Its language has as core concept the *Mentat*, an agent skeleton with mental properties that modellers can extend for their models. However, this work 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 domain approaches, and support alternative uses of models and their reuse.

3. INGENIAS Metamodels for ABM

INGENIAS [27] is a software development methodology for MASs. 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 MASs, 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, which 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 algorithms.

The INGENIAS Development Kit (IDK) software tool allows processing models with its modules. Standard modules available with the distribution support model auto-completion for certain tasks, documentation and code generation. Model manipulation is performed through code, complemented sometimes with the use of templates (e.g. for documentation and coding). A template gives a general description of a modelling structure with slots that are instantiated with information from specific elements in models. These templates allow representing repetitive transformations of these structures to text. For instance, researchers building a simulation for the MASON platform [23] 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 modeller needs. It must be noted that the development of new modules may require the support of engineers.

The approach of the presented work uses these elements to provide Domain-Specific Modelling Languages (DSMLs) and tools, with guidelines for their use [11]. It modifies the INGENIAS metamodel in order to include more flexible extension mechanisms that facilitate the definition of those DSMLs. 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, our work 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 [13] 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 [11].

With this infrastructure, the modelling process starts with the preliminary definition of the metamodels for the DSMLs related to a given domain with certain hypothesis. These DSMLs are usually at the domain level, as the INGENIAS language is suitable for computational aspects. In this preliminary stage, modellers and engineers also develop the transformations that map modelling structures in the DSMLs to the INGENIAS language. After this first step, modelling projects make use of these elements. Modellers use the tailored tools to model their problem and generate the simulation code. In this process, they also check the available infrastructure. For instance, they can find misconceptions in a DSML, missed concepts or inadequate results in the

simulation. They fix these issues with the help of engineers, improving the available languages and transformations. Hence, the definition of these elements for a domain is an iterative and incremental process performed through several projects. After some iterations, these elements are stable enough to remain unchanged in most of projects, so effort is only devoted to model project-specific problems.

This overall approach is expected to increase the autonomy of social researchers and modellers. The final goal is that they mostly develop their models autonomously, requiring only the support of engineers to address new and non-considered features of the domain. The case study chosen here illustrates this to show the advantages and current limitations of the work.

4. Case Study: Posted Pricing Institutions

An Introduction to the Institutions

The Posted-Bid Auction (PBA) and the Posted-Offer Auction (POA) are two complementary types of auctions. They are classified within the group generally known as *Posted Pricing Auctions* (PPAs).

The standard Posted Pricing Auctions [5, 33] consist of an indeterminately repeated series of periods. In each period, the process occurs as a sequence of two steps.

In the case of a POA (respectively a PBA) each seller (buyer) independently selects a "*take-it-or-leave-it*" price offer (bid) for a given quantity of goods, i.e. the seller (buyer) i offers (bids for) Y units of good X at a price p each. Each seller (buyer) chooses her price without knowledge of the prices that are being selected by competitors. When all the sellers (buyers) have had the opportunity to realize their offer (bid), the list of prices (but not the quantity limit, to avoid the phenomena of collusion) are made public to all buyers and sellers, for example posting manually on a blackboard or on computer screens in computerised systems.

In the next step, buyers (sellers) are randomly ordered in a sequence. Iteratively and one at a time according to the sequence, the selected buyer (seller) has the opportunity to make as many purchases as desired from sellers at the posted prices. In order to do it, the buyer can purchase from any of the sellers (buyers) that has not sold her maximum quantity. The buyer (seller) asks the seller (buyer) for a quantity. If that quantity is available, the seller (buyer) responds accepting and forming a binding contract. If some portion of the quantity offer is not accepted, the buyer (seller) may choose a new seller (buyer) and make a quantity offer to fill her unsatisfied demand, and so on.

When a buyer (seller) has finished with her contracts, the foreclosed goods are removed from the auction and the next buyer (seller) is selected from the sequence. This process is repeated during the trading period until all buyers (sellers) have completed their contracts or all sellers (buyers) are out of stock. Participants estimate their benefits and a new period of auctions typically follows.

In both cases, the buyers and the sellers have earnings in a symmetric manner. Each buyer has a private redemption value v for a good unit, and earns the difference between this value and the contract price p for each purchased unit ($v - p$). On the other hand, each seller has a private unit cost c, so earns the difference between the contract price for each selling unit and this cost value ($p - c$).

PPAs can be used in markets close to monopoly situations (few buyers - many sellers, and vice versa). They allow minimizing the dominance position of one side, as its mechanism eliminates price competition on the opposite side. Peters [28] finds that the posted-price mechanism is optimal in a model where sellers cannot communicate with buyers simultaneously and have to do it sequentially.

PBAs are used, for example, in some power markets, like in the case of refiners, in which they post price bids at which they are willing to buy crude oil [32]. POAs are also used for markets where sellers post prices which are not subject to alteration for some considerable period of time. For instance, this is the case of clothes wholesalers trying to sell their production of seasonal clothing to retail markets, in order to exploit economies of scale in the process [32]. Retailers post an offer price, and normally do not specify a quantity (except perhaps to say "while they

last"), but stocks may be exhausted before all buyers are satisfied.

Trying to understand the behaviour in auctions as relevant as the PPAs and the interaction under controlled conditions among the triplet IEA (*Institution* x *Environment* x *Agent's behaviour*) that defines any market [34], leads to complement experimental economics with agent based modelling experiments. This approach has been developed in other well-known types of auctions as the continuous double auction [15, 16, 29] or dynamic online auctions [25].

In order to illustrate the exposed approach, this paper considers its application for the explained type of auctions. The discussion focuses on the definition of the metamodel for the related DSML. The concepts developed give us a common modelling framework to compare and afterwards understand the dynamic behaviour of the market institution depending on the environment and agent's behaviour. Some brief remarks are also included about the definition of the needed transformations, and the use of the DSML and the transformations to study these auctions in specific problems.

main stages: the identification of the core concepts of the domain and the description of their standard interactions. The metamodel for a problem describes these elements as extensions of the primitives available in INGENIAS [27].

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. Another key concept is that of role. Roles are similar to agents but intended to specify knowledge and behaviour without describing their actual implementation. For instance, a buyer role specifies the general behaviour of this trader in an auction, but it is an agent playing that role who gives the actual utility function. Agents and roles that share goals, knowledge, procedures, rules or applications oriented to a given set of tasks, belong to the same *group*; when *groups* share part of these elements they belong to the same *organisation* Organisations have a traversal character, since their features affect different groups of tasks.

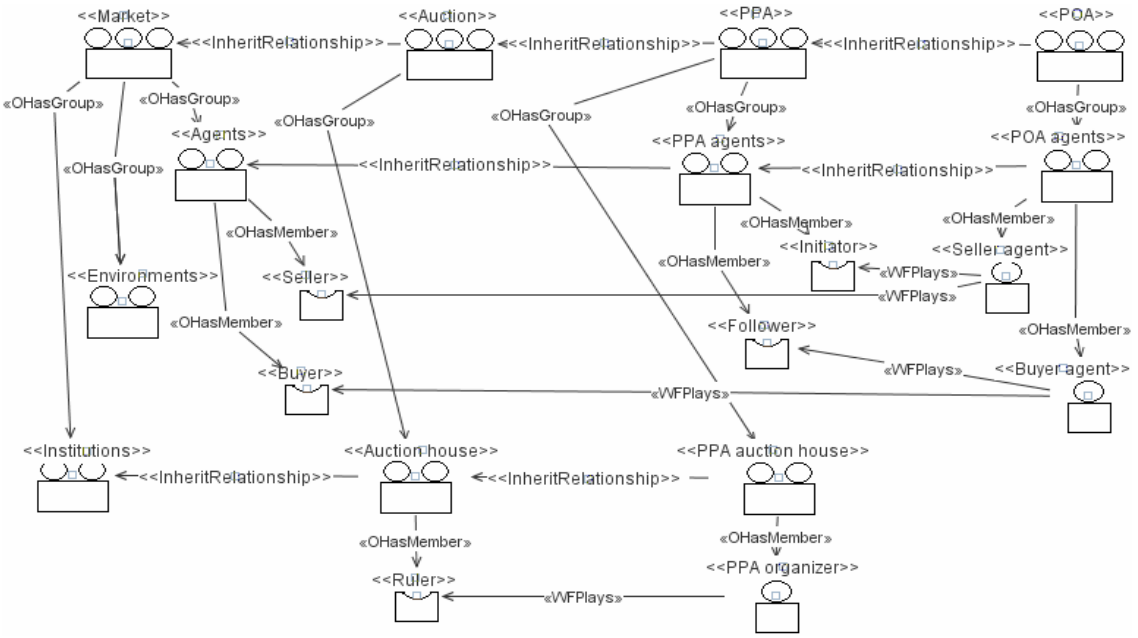

Figure 1. Partial metamodel for the structural components of the PPAs. Stereotypes (i.e. names between guillemets) denote types of entities and relationships.

Metamodelling

The process proposed in [11] to define the metamodels of DSMLs comprehends two

Figure 1 illustrates part of the hierarchies of elements proposed for the modelling language of POAs. This diagram is therefore

focused in the vocabulary of the domain. Following [34], the basis of the hierarchies is the *market* that determines the *institutions*, the *environments* and the *agents*. Agents in markets can play two roles, *buyer* and *seller*, which are not exclusive.

The metamodel represents the *market* as an organisation. It includes global information about its purpose: selling and acquiring goods with some competition in prices. The market comprehends three groups, connected to the organisation with the *OHasGroup* relationship. Each group represents one of the components of the market triple: *institutions*, *environments* and *agents*. Regarding our analysis for POAs, two groups are of special relevance: *institutions* that represent the rulers of the market; and *agents*, which represent buyers and sellers. The members of a group are indicated with the *OHasMember* relationship. As said before, groups are oriented to certain groups of tasks, and their components share common features for that purpose. For instance, institutions have power to impose a given protocol in behaviour, e.g. that of auctions, and common goals of promoting good exchange that maximize benefits of participants.

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 offers). The *ruler* role only specifies these rules, but not their actual implementation, which in this example is the responsibility of the *PPA Organizer* agent.

PBAs are buyers who initially post their bids. This is shown by incorporating a new level of refinement. *PPA agents* are the specific type of the *agents* group (i.e. *InheritRelationship*) that participate in *PPAs*. This group adds two new roles to the *agents* group, those of *iniatiator* (who starts the auction) and *follower* (who selects among the bids or offers posted by the *initiators*). In the case of a POA, a *seller agent* plays both the *seller* and *initiator* roles, and a *buyer agent* the *buyer* and *follower* roles. The *WFPlays* relationship indicates that an agent plays a role. Note that these agents need to give specific implementations of aspects specified by the roles, such as the utility function or the procedure to decide bids and offers. The PPA also adds a specific *PPA organizer* agent to play the role *ruler* and tailor the institutional behaviour to the norms of PPAs.

In this case, there is no need to identify external entities. If any, they could appear in Figure 1 linked to the societies, groups, agents or roles that control or use them.

As previously explained in this section, roles describe knowledge and responsibilities of active participants. Figure 2 shows a partial specification of the *buyer* role. It pursues a high level objective *acquire good*. Although figures do not show it, it is possible to decompose objectives in sub-objectives and alternative objectives to represent complex decision making of roles or agents. The two main tasks of the *buyer* role are *accept offer* to buy the good related with an available offer, and *generate bid* trying to buy some goods at certain price.

Figure 2 does not indicate anything about how the actual decision making for the buyer role takes place. Questions such as when the buyer considers an offer acceptable or when

Figure 2. Especification of the goals and responsabilities of the *buyer* role.

Among auctions, there are PPAs. Depending on the sub-type of PPA (POAs or PBAs), the initiator of the auction changes between buyers and sellers. For POAs, sellers initiate the auction posting theirs offers, while for

it triggers the tasks are represented with mental states. Figure 3 contains examples of them.

The *buyer agent* plays the *buyer* role as seen in Figure 1. This agent includes a mental state

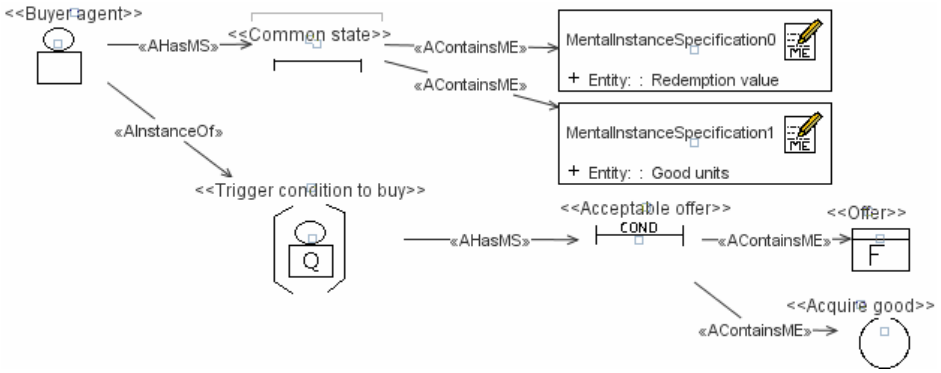

Figure 3. Mental state of the buyer agent and graphical specification of the condition to buy.

called *common state* (see Figure 3). This means that in a common situation, the agent has personal knowledge about the information included in this state. In this case, that mental state contains two pieces of information, the *redemption value* of the agent and the *good units* it tries to acquire. These elements are represented as attributes of *MentalInstanceSpecifications*. Every instance of this type of agent can have its own values for these attributes, that is, they represents individual information and not group information shared by buyers. The agent also has linked a condition *trigger condition to buy*. This condition determines when to trigger the task *accept offer*. The information relevant for the condition is graphically shown with the *AContainsME* relationship: an *offer* and the information related with the goal *acquire good*. The actual condition, i.e. mathematical relationship between values, could be shown in the condition modelling element as text, but there is not a predefined formalism for it.

The previous example is illustrative of certain tradeoffs to consider when defining a modelling language. Models are useful to describe synthesised visions of complex phenomena. As such, they focus on some aspects of problems but disregard others, and must provide a suitable level of abstraction to be a proper analysis tool. For instance, the previous element and also showing the participation of the ruler. It indicates the agents that start exchanges of information (i.e. *UInitiates* relationship) and collaborate in them (i.e. *UIColaborates* relationship). These relationships link the agents with the actual tasks that send or receive the information (e.g. *Generate offer* and *receive offer*) and the exchanged information represented by rectangles with arrows inside. Note that several aspects of the interaction are left out because of space reasons. For instance, the fact that all the sellers post their offers at the same time or that buyers choose offers sequentially require additional diagrams.

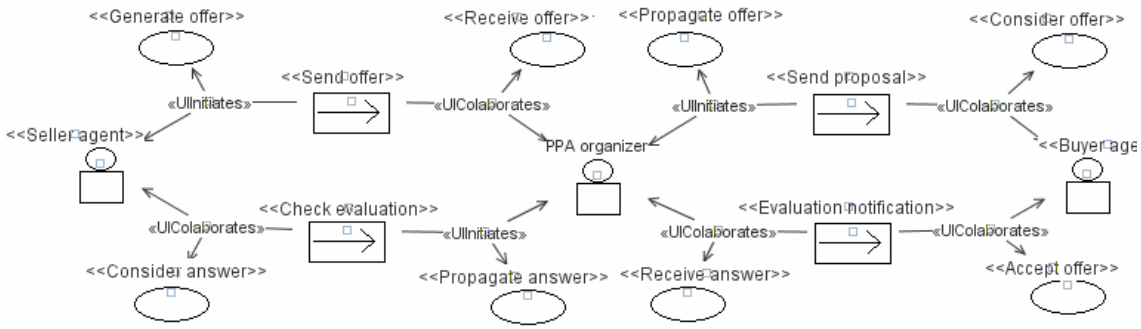

Figure 4. Partial interaction for the exchange of information between ruler, buyer and seller agents.

The final element of the specifications is the interactions describing how agents and roles participate in the global activity of the system. Figure 4 shows an excerpt involving

metamodel could include additional graphical primitives to describe the condition in Figure 3. However, these could be not advisable, as the language would become much more

complex and these graphical specifications would be probably too verbose. Regarding non-considered aspects in the metamodel, the internal evaluation of traders focuses on their utility function, but it does not consider, for instance, how the mood of traders can lead them to mistakes.

This discussion has provided an overview of the metamodel definition process in our work. The overall approach proposed in [11] and outlined in Section 3 also includes the definition of the related transformations and the modelling itself in projects.

Transformations in this case translate the concepts of the DSML for POAs to their super-concepts in the INGENIAS language. This translation is suitable thanks to the close alignment of the concepts in both languages. Nevertheless, other AOSE languages different than the INGENIAS one could be used as basis for this process, as far as they have available a rich enough set of basic modelling primitives and suitable extension mechanism to define new languages. Additionally, transformations add simulation-oriented details for code generation. For instance, INGENIAS agents need to include an explicit management of their mental state, which covers aspects such as removing satisfied objectives or pieces of information already used.

Finally the modelling of specific problems, such as the energy or clothe markets previously mentioned, is done using these DSML and transformations, and their related tools.

5. Conclusions and Discussion

This paper has presented the definition of a metamodel that defines a modelling language for PPAs, and particularly for POAs. 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.

The classical studies on these auctions have been the basis to identify the key concepts present in them, the attributes that can determine the behaviour of their traders, and their interactions among them and with the organizers. 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 information exchange between roles and agents, and satisfaction relationships between tasks and goals, together with particular conditions, as those used for triggering a task.

The metamodel shows the common participants in auctions with the components that researchers need to specify and test different hypotheses. For instance, POAs do not require their agents rationally behave regarding their utility functions. If other traders are going to be tested, researchers may need to add new attributes, different behaviours for tasks, or even additional capabilities. These elements would participate in conditions or change implementations, 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 for new modelling projects.

This case study has adopted the language of the INGENIAS AOSE methodology as the basis to develop its metamodel. Nevertheless, this is not a mandatory choice. Alternative modelling languages could be used if they provide suitable sets of modelling primitives and extension mechanisms. An advantage of INGENIAS in this context is that it is close to ABM concepts and has a full MDE development process for software systems, and therefore for simulations.

The main limitation of the proposed approach comes from the difficulties to formalize the vocabulary used to model a domain. This is a common problem in the definition of modelling languages in general. Our approach tries to limit it with agent-oriented languages, which are close to the target social systems studied with ABM. Improved guidelines for these tasks can also reduce the effort in these tasks. Information for these improvements would be obtained from the exploration of additional domains such as decision making [35] or politics [21].

The advantages of the proposed approach, illustrated with the case study, 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.

Acknowledgements

We acknowledge support from the project "Agent-based Modelling and Simulation of Complex Social Systems (SiCoSSys)", supported by the Spanish Council for Science and Innovation, with grants TIN2008-06464-C03-01 and TIN2008-06464-C03-02, the Junta de Castilla y León with grant GREX251/2009 and Caja Burgos with grant 2009/00148/001.

REFERENCES

1. AXTELL, R. L., **Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences,** in: Macal CM, Sallach D, eds. Proceedings of the Workshop on Agent Simulation: Applications, Models, and Tools. Argonne National Laboratory, Argonne, IL, 2000, pp. 3-24.
2. BERNON, C., M. P. GLEIZES, S. PEYRUQUEOU, G. PICARD, **ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering,** Lecture Notes in Artificial Intelligence 2577, 2003, pp. 156-169.
3. LE BEUX, S., P. MARQUET, J. L. DEKEUSER, **A Model Driven Co-Design Approach for High Performance Embedded Systems Dedicated to Transport,** Studies in Informatics and Control, vol. 17(4), 2008, pp. 407-420.
4. BONABEAU, E., **Agent-based Modeling: Methods and Techniques for Simulating Human Systems,** Proceedings of the National Academy of Sciences of the United States of America, 99(2), 2002, pp. 7280-7287.
5. DAVIS, D. D., C. A. HOLT, **Experimental Economics,** Princeton University Press, Princeton, N. J., 1993.
6. DROGOUL, A., D. VANBERGUE, T. MEURISSE, **Multi-Agent Based Simulation: Where are the Agents?,** in: Sichman JS, Bousquet F, Davidsson P, eds. Proceedings of MABS 2002 Multi-Agent-Based Simulation, Lecture Notes in Computer Science 2581. Springer-Verlag, Bologna, Italy, 2003, pp. 1-15.
7. EDMONDS, B., **The Use of Models - Making MABS Actually Work**, in: Moss S, Davidsson P, eds. Multi-Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1979. Springer-Verlag, Berlin, 2001, pp. 15-32.
8. EDMONDS, B., D. HALES, **Replication, Replication and Replication: Some Hard Lessons from Model Alignment,** Journal of Artificial Societies and Social Simulation, vol. 6(4), 2003.
9. EPSTEIN, J. M., **Agent-based Computational Models and Generative Social Science**, Complexity, vol. 4(5), 1999, pp. 41-60.
10. FRANCE, R., B. RUMPE, **Model-driven Development of Complex Software: A Research Roadmap**, in: Proceedings of the 2007 Future of Software Engineering (FOSE 2007), IEEE Computer Society, 2007, pp 37-54.
11. FUENTES-FERNÁNDEZ, R., J. M. GALÁN, S. HASSAN, A. LÓPEZ-PAREDES, J. PAVÓN, **Application of Model Driven Techniques for Agent-Based Simulation**, Advances in Practical Applications of Agents and Multiagent

Systems, Advances in Soft Computing, Springer 70, 2010, pp. 81-90.

12. GALÁN, J. M., L. R. IZQUIERDO, S. S. IZQUIERDO, J. I. SANTOS, R. DEL OLMO, A. LÓPEZ-PAREDES, B. EDMONDS, **Errors and Artefacts in Agent-based Modelling**, Journal of Artificial Societies and Social Simulation, vol. 12(1), 2009, 1.
13. GARCÍA-MAGARIÑO, I., S. ROUGEMAILLE, R. FUENTES-FERNÁNDEZ, F. MIGEON, M. P. GLEIZES, J. GÓMEZ-SANZ, **A Tool for Generating Model Transformations By-Example in Multi-Agent Systems**, in: Demazeau Y, Pavón J, Corchado JM et al., eds. Proceedings of the 7th International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer, Heidelberg, 2009, pp. 70-79.
14. GILBERT, N., K. G. TROITZSCH, **Simulation for the Social Scientist**, Open University Press, Buckingham, UK, 1999.
15. GODE, D. K., S. SUNDER, **Allocative Efficiency of Markets with Zero-Intelligence Traders - Market As A Partial Substitute for Individual Rationality,** Journal of Political Economy, vol. 101(1), 1993, pp. 119-137.
16. GODE, D. K., S. SUNDER, **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, Addison-Wesley, New York, NY, 1993.
17. GRIMM, V., U. BERGER, F. BASTIANSEN, et al., **A Standard Protocol for Describing Individual-based and Agent-based Models**, Ecological Modelling, vol. 198(1-2), 2006, pp. 115-126.
18. HASSAN, S., L. ANTUNES, J. PAVÓN, **Mentat: A Data-driven Agent-based Simulation of Social Values Evolution**, Lecture Notes in Artificial Intelligence, vol. 5683, 2010, pp. 135-146.
19. HENDERSON-SELLERS, B., P. GIORGIINI, **Agent-Oriented Methodologies,** Idea Group Publishing, 2005.
20. IZQUIERDO, S. S., L. R. IZQUIERDO, F. VEGA-REDONDO, **The Option to Leave: Conditional Dissociation in the Evolution of Cooperation,** Journal of Theoretical Biology, vol. 267(1), 2010, pp. 76-84.
21. JOHNSON, P. E., **Simulation Modeling in Political Science**, The American Behavioral Scientist, vol. 42(10), 1999, pp. 1509-1530.
22. LÓPEZ-PAREDES, A., C. HERNÁNDEZ, **Agent-based Modelling in Natural Resource Management**, Insisoc, Madrid, 2008.
23. LUKE, S., C. CIOFFI-REVILLA, L. PANAIT, K. SULLIVAN, G. BALAN, **MASON: A Multi-Agent Simulation Environment**, Simulation, vol. 81(7), 2005, pp. 517-527.
24. MINAR, N., R. BURKHART, C. LANGTON, **The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations**, Santa Fe Institute Working Paper 96-06-042, 1996.
25. MIZUTA, H., K. STEIGLITZ, **Agent-based Simulation of Dynamic Online Auctions,** in: Joines JA, Barton RR, Kang K et al., eds. Proceedings of the 2000 Winter Simulation Conference, IEEE, Orlando, FL, 2000, pp. 1772-1777.
26. NIKOLAI, C., G. MADEY, **Tools of the Trade: A Survey of Various Agent Based Modeling Platforms**, Journal of Artificial Societies and Social Simulation, vol. 12(2), 2009, 2.
27. PAVÓN J., J. J. GÓMEZ-SANZ, R. FUENTES, **The INGENIAS Methodology and Tools**, in: Henderson-Sellers B, Giorgini P, eds. Agent-Oriented Methodologies, Idea Group Publishing, Hershey, PA, 2005, pp. 236-276.
28. PETERS, M., **Equilibrium Mechanisms in a Decentralized Market,** Journal of Economic Theory, vol. 64(2), 1994, pp. 390-423.

29. POSADA, M., A. LÓPEZ-PAREDES, **How to Choose the Bidding Strategy in Continuous Double Auctions: Imitation versus Take-the-best Heuristics**. Journal of Artificial Societies and Social Simulation, vol. 11(1), 2008, 6.

30. RAILSBACK, S. F., S. L. LYTINEN, S. K. JACKSON, **Agent-based Simulation Platforms: Review and Development Recommendations**, Simulation, vol. 82(9), 2006, pp. 609-623.

31. SANSORES, C., J. PAVÓN, **Agent-Based Simulation Replication: A Model Driven Architecture Approach**, Lecture Notes in Computer Science 3789, 2005, pp. 244-253.

32. SMITH, V. L,. **Bidding and Auctioning Institutions: Experimental Results**, in: Yakov A., ed. Bidding and Auctioning for Procurement and Allocation. New York University Press, New York, NY, 1976.

33. SMITH, V. L., **An Empirical Study of Decentralized Institutions of Monopoly Restraint**, in: Quirk J, Horwich G, eds. Essays in Contemporary Fields of Economics in Honor of E. T. Weiler (1914-1979). Purdue University Press, West Lafayette, IN, 1981.

34. SMITH, V. L., **Microeconomic Systems as an Experimental Science**, American Economic Review, vol. 72(5), 1982, pp. 923-955.

35. ZARATÉ, P., **Decision Making Process: A Collaborative Perspective**, Studies in Informatics and Control, vol. 17(2), 2008, pp. 225-230.