Reducing the Modeling Gap: On the Use of Metamodels in Agent-Based Simulation

Samer Hassan[1], Rubén Fuentes-Fernández[1], José M. Galán[2], Adolfo López-Paredes[3], Juan Pavón[1]

[1] GRASIA: Grupo de Agentes Software, Ingeniería y Aplicaciones, Dept. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, c/ Profesor José García Santesmases s/n, 28040, Madrid, Spain
{samer, ruben, jpavon}@fdi.ucm.es
[2] INSISOC: Ingeniería de los Sistemas Sociales. Dept. de Ingeniería Civil, Universidad de Burgos, Ed. La Milanera, c/ Villadiego s/n, 09001, Burgos, Spain
jmgalan@ubu.es
[3] INSISOC: Ingeniería de los Sistemas Sociales. ETS Ingenieros Industriales. Universidad de Valladolid, Paseo del Cauce s/n, 47011, Valladolid, Spain
adolfo@insisoc.org

Abstract. Although there are many methodologies for the development of multi-agent systems, these are scarcely applied for agent-based simulation. The different activities for the creation of an agent based model and its simulation for the study of a social system, which include abstraction, design, approximation and coding, are usually performed in a monolithic way. Besides, when several roles take part in this process, communication problems may arise between people with different backgrounds and perspectives. This work proposes the use of agent oriented software engineering methods and tools, relying on a modeling middle-layer to solve or reduce these communication and integration problems. In particular, a meta-modeling approach using INGENIAS is applied for multi-agent systems, as a high-level formal methodological language, to define languages in the form of diagrams that are conceptually close to the domain-expert, in this case for simulation of social systems. This paper shows how this approach can facilitate the communication, specification, implementation and validation of social simulation models. A case study of an urban dynamics data-driven model illustrates this discussion.

Keywords: Agent-based modeling, agent-oriented software engineering, metamodeling, model driven development, social simulation, urban dynamics.

1   Introduction: roles in agent-based modeling

In the lasts decades, computer simulation in general, and agent based modeling (ABM) in particular, has become one of the mainstream modeling techniques in many scientific fields, especially in Social Sciences such as Economy [1] or Sociology [2, 3]. ABM allows modelers to complement non-formal models, usually expressed in

natural language, with computer models, which are more formal. This combination avoids, at least partially, including brave assumptions to make it analytically tractable [4]. One of the main advantages of ABM, and in our opinion the one that distinguishes it from other modeling paradigms, is that it facilitates a direct correspondence between the entities in the target system, and the parts of the computational model that represent them (*i.e.* the agents) [5].

The process of abstraction to transform the real target system into a simulation model is a complex one. The process involves different subtasks and roles, which need diverse backgrounds and competences in the design, implementation and use of an archetypical agent based simulation. Drogoul et al. [6] identify three different roles in the modeling process: the *thematician*, the *modeler*, and the *computer scientist*. This classification has been expanded by Galán et al. [7] using the framework proposed by Edmonds [5], including an additional role, the *programmer*.

The role of the thematician, which ideally would be performed by an expert in the domain, aims at producing the first conceptualization of the target system. This task entails: defining the objectives and the purpose of the modeling exercise; identifying the relevant components of the system and the relations between them; describing the most important causal dependencies.

The modeler's job is to produce formal requirements for the models starting from the thematician's ideas. These requirements allow the computer scientist to formulate a feasible model that can run in a computer. However, not all the formal specifications can be directly implemented in a computer. The computer scientist role finds a suitable approximation to the modeler's formal model that can be executed in a computational system with the available technology. Finally, the programmer's role is to implement the computer scientist's model to a target simulation platform.

In practical terms, modeling in social sciences faces two problems. In individual developments, it is difficult that one person has all the expertise required; in teams, there are communication problems.

The first problem appears when the same person plays all the roles of the process [6-8]. Minar et al. [9] explain its negative consequences as follows: "*Unfortunately, computer modeling frequently turns good scientists into bad programmers. Most scientists are not trained as software engineers. As a consequence, many home-grown computational experimental tools are (from a software engineering perspective) poorly designed*". Besides, scientists may find difficult understanding the detailed behavior of the underlying software, since doing it would imply a full understanding of its implementation.

On the other hand, many problems require a multidisciplinary perspective, involving members with specialized roles. The second problem arises because effective communication between experts of fundamentally different domains (e.g. sociology and computer science) is not trivial. In most cases, it is difficult to grasp how the social features have been mapped to program constructions. Thus, there are difficulties to assure that the program really implements its conceptual model.

To address these problems, our research promotes the creation of a set of high-level tools, methods and languages, to assist the transfer of models between different roles in the modeling process. These tools should work with modeling languages that include, ideally, concepts close to the thematician's background, but at the same time representing ideas from a software engineering point of view. We must take into

account that any mismatch between the specifications and the actual model passed to the next stage, will end up producing an error.

Moreover, a high-level communication tool may also help in the validation. Model validation is the process of determining that the model behavior represents the real system to satisfactory levels of confidence and accuracy, which are determined by the intended model application and its application domain. When dealing with complex systems, as it is frequent on ABM, the traditional methods used in model validation are not widely accepted [10]. In such cases, a good option for the validation of the conceptual model is to check whether the theoretical foundations and assumptions are reasonable within the context of the objectives of the simulation. This structural validation is sometimes performed on the basis of participatory methods with experts in the modeled domain and stakeholders [11]. Again, these expert panels do not usually have a software engineering background. Intermediate languages between the different roles endowed with a high-level descriptiveness facilitate the communication, modification and criticism of the models in the validation stages.

The rest of the paper is organized as follows. The high-level communication tool chosen, the metamodels, is introduced in the next section, together with an explanation on how to apply them in agent-based social simulation. Subsequently, the methodological process is illustrated step-by-step by means of a case study about a data driven urban dynamics model. The paper ends with a discussion and concluding remarks concerning the approach.

2   A Middle-layer for Agent-based Modeling: Metamodels

2.1   Introducing Metamodels to Social Scientists

In order to solve the mentioned problems, some authors [12] consider the use of domain-specific languages (DSL) [13] to produce intermediate models between the thematician's abstract non-formal models and the final program, but close to the thematician's view. A DSL explicitly defines its concepts, attributes, relationships, and potential constraints applicable to the models specified with it. These elements are "domain-specific" because they are extracted from the target domain. Thus, researchers working with a DSL specify their models using the language of their discipline. Moreover, since the language elements are clearly defined, mapping them to software constructions is significantly easier and more reproducible than in the case of an arbitrary set of elements, which is the situation with non-formal descriptions.

Metamodels [14] are the main resource to define graph-like DSL. A graph-like DSL has nodes (i.e. entities or concepts) linked by arcs (i.e. relationships or references), and both of them can have properties (i.e. features or attributes). A metamodel for a given DSL defines the types of nodes and arcs that correspond to the domain concepts. It indicates their names and attributes, and the rules and constraints that they satisfy, for instance the concepts a relationship can link or its cardinality.

Metamodels have two key advantages for defining DSL. First, they can be extended to satisfy specific modeling needs: if the current form of the DSL is not

José Manuel Galán 29/6/09 12:45

Comentario: *Aquí se debería definir que es un metamodelo*

enough to model a given problem, new elements can be introduced as extensions or specializations of the existing ones. These new elements provide an accurate representation of the domain notions according to the definition of the thematician. Second, metamodels are broadly used in Model-Driven Engineering (MDE) [15] for software development. Thus, there is a wide range of support tools to work with them. For ABM is particularly relevant the availability of graphical editors to define these metamodels and to generate from them customized editors for DSL. This facilitates providing researchers with specific modeling tools for their domains, which improve their productivity and reduce the probability of mistakes.

Table 1. Main concepts of the INGENIAS modeling language.

Concept	Meaning	Icons
Agent	An active concept with explicit goals that is able to initiate some actions involving other elements of the simulation.	
Role	A role groups related goals and tasks. An agent playing a role acquires the goals and tasks of such role.	
Environment application	An element of the environment. Agents act on the environment using its actions and receive information through its events.	
Goal	An objective of an agent. Agents try to satisfy their goals executing tasks. The satisfaction or failure of a goal depends on the presence or absence of some elements (i.e. frame facts and events) in the society or the environment.	
Task	A capability of an agent. In order to execute a task, certain elements (i.e. frame facts and events) must be available. The execution produces/consumes some elements as result.	
Frame Fact	An element produced by a task, and therefore by the agents.	
Event	An element produced by an environment application.	
Interaction	Any kind of social activity involving several agents.	
Group	A set of agents that share some common goals and the applications they have access to.	
Society	A set of agents, applications and groups, along with some general rules that govern the agent and group behavior.	

2.2 Applying Metamodels to Agent-Based Modeling

The proposed approach for ABM is based on an agent-oriented software engineering methodology, INGENIAS [16, 17]. There are two main reasons for the choice of INGENIAS as starting point for this work. First, its modeling language supports well the specification of organization structure and dynamics, as well as agent intentional behavior, characteristics that are present in social systems. This is a feature that general purpose modeling languages lack. This language is supported by the

INGENIAS Development Kit (IDK) with a graphical editor, which can be extended to introduce new modeling concepts. Second, INGENIAS promotes a model-driven engineering approach [12] that facilitates the independence of the modeling language with respect to the implementation platform. This is especially important here in order to abstract away programming details and concentrate on modeling and analysis of social patterns. With this purpose the IDK supports the definition of transformations between models and code for a range of implementation platforms.. The purpose of this work is to shift this focus to DSL for social sciences researchers, providing a process and tools for the development of domain-specific social simulations.

Table 1 summarizes the main concepts of INGENIAS used in the rest of the paper. Note that this table considers concepts but not their relationships. The relationships used in the case study have self-explanatory names, although the paper explains them when introduced. Consider for instance the relationship "*WFProduces*" from a "*Task*" to a "*Frame Fact*". The first two letters indicate the main type of INGENIAS diagram where the type of relationship appears. In this case, "*WF*" stands for "*WorkFlow*" diagram. The rest of the name provides the meaning of the relationship. "*Produces*" shows that the fact is the result of the execution of the task. Inheritance relationships deserve special attention, since they are the simplest means to add new specialized concepts. The new sub-concept inherits all the features of its super-concept (as in standard object-oriented programming) but it can extend or constrain them.

According to the elements available in the INGENIAS modeling language, a simple process can be proposed to define the metamodel and related models for a given problem domain. Fig. 1 summarizes this process. Each round rectangle represents an activity, and activities between thick horizontal lines are carried out in parallel. The involved activities and decisions are:

1. *Domain analysis*. Thematicians consider the concepts that are required to express their hypotheses and the related information in the group or society.
2. *Determine interactive concepts*. Among the domain concepts, some of them focus on the analysis, and thus they are considered as decision makers that follow certain rationality. Besides, some concepts represent elements that initiate interactions with others, for instance asking for some services. All these concepts are candidate agents in the ABM and therefore they are represented as subclasses of Agent. Note here that, as it will be shown in the case study, an agent is not necessary a living-like being: any element that engages in interactions with other elements of the system should be modeled also as an agent.
3. *Determine non-interactive concepts*. Passive elements that do not take decisions are regarded as part of the environment. In INGENIAS, these elements are modeled as subclasses of the environment application.
4. *Determine specialization hierarchies between concepts*. The elements introduced for a problem usually share some features. In order to highlight the common aspects of elements and encourage their reusability, these new elements are arranged in inheritance hierarchies. A super-concept contains all the elements/attributes and participates in all the relationships common to its sub-concepts. Sub-concepts only modify their own specific features, constraining or adding some features of the super-concept.

5. *Determine groups and societies*. In case that several agents share common global goals or environment applications, they can be gathered in groups. If they also share common rules of behavior, they constitute a society.
6. *Determine interactions*. To carry out the activities of the system, agents act on environment applications, receive information from them, and communicate with other agents exchanging elements of the system. A group of interconnected activities aimed at satisfying a global goal constitutes an interaction.
7. *Assign roles, objectives and capabilities to agents*. INGENIAS refines agent definitions with the tasks they are able to do, which correspond to their capabilities and goals. A goal is linked to the tasks which are able to satisfy it. These tasks produce some elements (i.e. frame facts). The presence or absence of some frame facts and events (produced by environment applications) are evidence of the satisfaction or failure in the achievement of the goal.
8. *Refine interactions*. A refined interaction indicates the agents and environment applications that participate in it, the tasks that agents execute, the goals they pursue with those executions, and the elements produced and consumed in it.
9. *Validate ABM*. These agent-based models are a refinement of the thematician's non-formal models. They are expected to represent it accurately, while providing additional details that facilitate the transition to the running system.
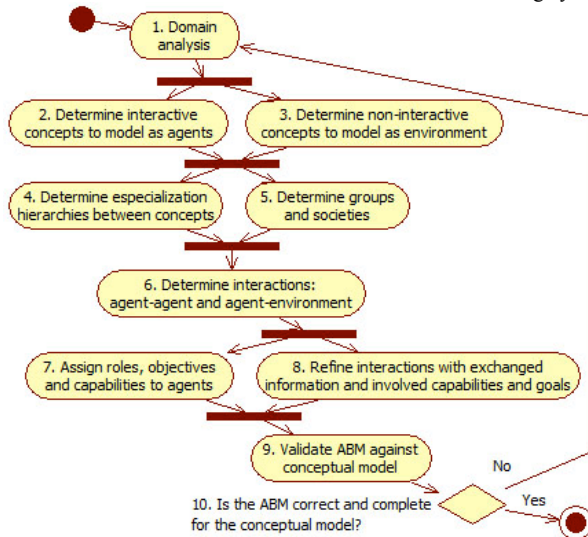

Fig. 1. Simplified process for the definition of metamodels and models for ABM.

Two remarks must be done about this modeling process. First, although it has been described as sequential, researchers do not need to follow the activities in this order. For instance, if there is a precise idea of the existing interactions, modelers can begin with activity 5 and then use this information to discover the agents and environment

applications. Besides, this presentation of the process is necessarily simplified given the space limitations, but more detailed activities are required to provide a full modeling guideline. Second, the previous process can be used to describe models or metamodels. If the concepts have a wide application for a domain, they become part of the metamodel; if they are specific for a problem, they remain at the model level.

This process can be regarded as recursive. That is, a given model can be the abstract model for a new and more platform-oriented model. This allows a transition from the abstract non-formal description to the program in several steps, improving the traceability of the process. This approach is in the basis of MDE, and INGENIAS fully implements it. The advantage of this decomposition is the possibility of using specific guidelines and support tools for each step, which crystallize and automate the expertise of thematicians and modelers and helps novel researchers.

3   Case Study

3.1   Context

The case study selected to illustrate the methodology presented in this paper is a model of urban dynamics. Several complementary theories from fields as Sociology, Geography, Political Science or Economics attempt to explain the complex problem of the dynamic spatial occupation [18]. There are different levels of abstraction where ABM has been used to model urban dynamics [19]. The lowest-level of abstraction is devoted to the application of very descriptive data-driven ABM in the analysis of realistic urban problems, making an intensive use of data (extracted, for instance, from Geographic Information Systems (GIS) and databases).

In order to illustrate the usefulness of metamodeling with INGENIAS in social simulation contexts, a data-driven urban dynamic model has been selected. One of the most descriptive models of urban dynamics applied to real systems is the Yaffo-Tel Aviv model developed by Benenson et al. [20, 21]. It considers the stress-resistance hypothesis [22, 23] as the force that explains intra-population movements in cities. This model has been adapted to the Valladolid metropolitan area (Spain) and, together with other socioeconomic models, is used for exploring the dynamics of urban phenomena [24, 25]. In order to adapt it to this new context and develop the different layers of the model, extensive discussions with domain experts (thematicians) were needed. The description presented in this section is a result of the intensive communication between modelers and those thematicians.

The model comprises two different layers. The first layer is retrieved from a vectorial GIS. This GIS explicitly represents every block with households in the studied geographical region and characterizes them by their spatial and socioeconomic characteristics. In the second layer, the computational agents representing the families that live in the area are spatially situated.

The main assumption of the model is that agent's selection of residence is influenced by intrinsic features of the candidate households and by the similarity of

the agents' socioeconomic factors with those of their neighborhoods. Thus, some of the rules of behavior of the agents in this model are based on the concept of neighborhood, defined taking into account the centroids of their blocks and the Voronoi tessellation. The variable "residential dissonance" quantifies the dissimilarity between an agent, its neighborhood and its household. The probability that an agent leaves a residence is considered proportional to such residential dissonance. This variable may be influenced by differences in terms of nationality or education level, or by imbalances between an agent's wealth and the value of the house where it lives.

Once each agent has calculated the dissonance, the opportunity to change its current residence is modeled through a stochastic process that transforms dissonance into probability of change. Those selected agents are included in a set M of potential internal migrants. If external immigration is enabled, immigrant agents are also included into the set M. Subsequently, each agent A in M estimates the attractiveness (one minus the dissonance) of a number of candidate empty households H_A.

The last step of the algorithm entails the assignment of agents in set M to empty households. Each potential migrant chooses the household found with highest attractiveness. If the dwelling is already empty, the agent occupies it with a probability depending on its attractiveness; if it is not, the agent removes the household from its H_A list. The process is iterative until H_A is empty for all the agents in M. In each iteration, the order of the agents is randomly selected to avoid bias in the agent selection [26]. The agents that have not been able to find a suitable house leave the city with probability L_A and remain in its current house with probability $1-L_A$. The immigrant population without household leaves the city as well.

Another submodel dynamically updates the prices of the households [27, 28]. The value of a household depends on the wealth of its family and neighbor families, and the value of the surrounding empty houses. The value of the empty households decreases at a constant rate considered as an exogenous parameter of the model.

3.2   Following the Steps

The development of the model for the case study following the previous methodology is as follows. Activity 1 corresponds to the descriptive analysis realized by the thematician before the discussions which resulted on the previous subsection, already from the modeler's point of view.

Activity 2 and 3 are made in parallel to identify the entities in the system. In this case, there are at least two active elements, the families and the households. Families initiate interactions to keep low their dissonance level; households perform an interaction with other households to update their value. Thus, both elements are modeled as two different classes of agents.

According to the specification of the problem, when a family wants to migrate, it receives a list of potential candidate households for migration, chosen randomly all over the city. However, communication constraints only allow a family to communicate directly with their neighbors, not all over the city. In order to allow families to get this information about other parts of the city, we introduce a *map* environment application, which is a non-interactive concept as defined in Activity 3.

The map is part of the environment and it provides methods to get the list of empty households and retrieve the house data (such as if it is empty).
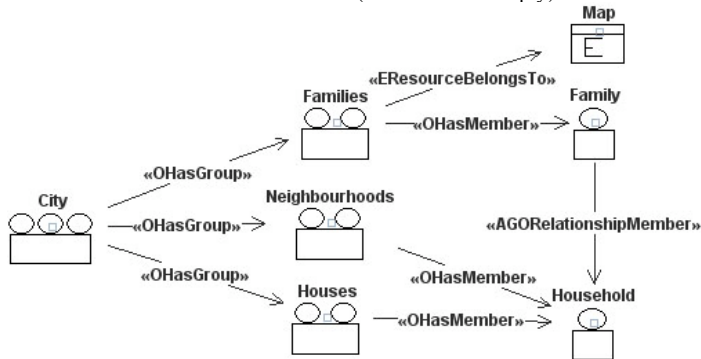

Fig. 2. The *city* society and its components. For the legend see Table 1.

Activities 4 and 5 are intended to create hierarchies and groups of elements that share features. At this step, there are only two types of agents and one environment application, so the hierarchies are flat. In order to represent the organization of the city, a society is introduced, together with three groups that can be seen in **¡Error!No se encuentra el origen de la referencia.**. The *city* society comprehends all the participants in the problem. They share rules of collaborative (e.g. all the agents are willing to provide the requested information) and non-violent behavior (e.g. no family can occupy the household of other family, and no household can eject its family). The three groups in this society are *families*, *households*, and *neighborhoods*. The neighborhoods include the households and the families that are able to communicate among them. This information is used to make explicit the influence of the neighborhood in the calculation of the dissonance level and the price of households. Note also that this information is useful in the implementation of the model. As Repast [29] was chosen as target platform, an agent can only communicate with those placed in adjacent cells. Thus, the neighborhoods tell us which agents must be in which cells. This is a typical example of information with social basis refined to guide the implementation.

Activity 6 identifies the relevant interactions for the problem: the calculation of the household prices and the family dissonance levels. Due to space restrictions, the following diagrams focused on the second interaction.

Activities 7 and 8 identify the tasks and goals of agents, the elements exchanged and their relations to specify the interactions. Fig. 3 and Fig. 4 include part of the results of these activities. They focus on the workflow the family agent performs after it finds out that its dissonance level is too high with respect to its neighborhood.

Fig. 3 shows the part of the workflow where the family agent looks for a suitable and unoccupied household. It begins asking for H unoccupied households using the map. Afterwards, it filters the list to get only the households whose dissonance level is below its threshold. Then, the family uses the map to try to get one of the suitable households. Note that according to the specification of the non-formal description, all

the families that are uncomfortable with their households try to make these tasks at the same time. First, all of them get the list, then they filter it, and afterwards each of them tries to get into a household. Given this order, it is possible that when a family tries to occupy an initially unoccupied household, another discontent family that chose first has already occupied it. For this reason, the task *check household* can fail.

Fig. 4 shows the interaction where the family agent asks the target household agent to accept it. All the involved tasks need to use the map for communication, as it appears in Fig. 3.
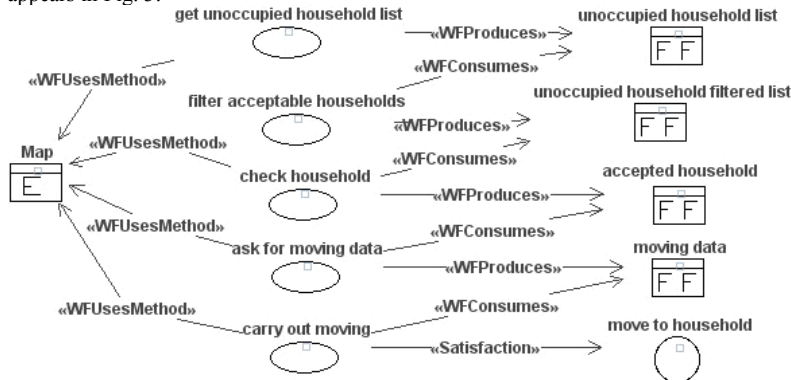

Fig. 3. *Migrate in the city* workflow for the *family* agent. For the legend see Table 1.
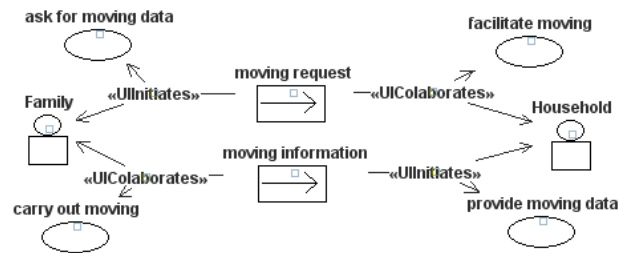

Fig. 4. End of the *migrate in the city* workflow with the interaction between the *family* agent and the target *household* agent. For the legend see Table 1.

4   Discussion and Concluding Remarks

The introduction of a high-level language, which is founded on well established agent concepts and close to the domain-expert, in the form of understandable diagrams, facilitates communication, specification, implementation and validation of agent-based models for the simulation of social systems. This has been validated by providing guidelines for agent-based modeling using the support of the default

modeling tools of a specific agent-oriented modeling language, INGENIAS. A case study on urban dynamics has been used to describe the process step-by-step.

This framework will allow the specification of social systems with a graphical modeling language, the simulation of the models of these systems by exploiting the capabilities of existing agent-based simulation tools/platforms, and the identification and analysis of social patterns (at a macroscopic, or aggregate, level) in terms of the atomic elements of the social system specification (at a microscopic, or individual/interaction, level). The advantages go further than usability. As it has been discussed in [12], this solution facilitates the replication of an experiment on different simulation engines, in order to contrast results. The availability of a graphical view of the system facilitates its understanding too and improves the identification of patterns in the system.

It has still to be evaluated the effort of learning a new language, but, in principle, a visual modeling language should be easier to use than a typical programming language. The main issue, however, is the effort that is required to adapt existing agent metamodels for creating a domain specific language. However, in the case of INGENIAS, this adaptation is feasible as both the language and the tool easily allow extensions to introduce new concepts and relations, together with graphical icons for them. A possible extension could be to differentiate the neighborhood from a standard agents group (such as "Families"), as it should be related with the space in some way.

Despite of these issues, this approach is considered a step forward in the search of more reliable and transparent agent-based models. The increase of formalization associated, together with the facilitation of replication, would restrain the typical criticism of complex models as obscure black-boxes.

References

1. Tesfatsion, L., Judd, K.L. Handbook of computational economics, Vol. 2: Agent-Based Computational Economics. Elsevier, Amsterdam (2006).
2. Macy, M.W., Willer, R.: From factors to actors: Computational sociology and agent-based modeling. Annual Review of Sociology. 28 (2002) 143-66.
3. Gotts, N.M., Polhill, J.G., Law, A.N.R.: Agent-based simulation in the study of social dilemmas. Artificial Intelligence Review. 19 (2003) 3-92.
4. Gilbert, N. Agent based models. Sage, London (2007).
5. Edmonds, B.: The Use of Models - making MABS actually work. In: Moss, S., Davidsson, P. (eds). Multi-Agent-Based Simulation, LNAI Vol. 1979. Springer-Verlag (2001) 15-32.
6. Drogoul, A., Vanbergue, D., Meurisse, T.: Multi-Agent Based Simulation: Where are the Agents? In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds). Proceedings of MABS 2002 Multi-Agent-Based Simulation, LNCS Vol. 2581. Springer-Verlag, Bologna (2003) 1-15.
7. Galán, J.M., Izquierdo, L.R., Izquierdo, S.S., et al.: Errors and artefacts in agent-based modelling. Journal of Artificial Societies and Social Simulation. 12(1) (2009) 1. http://jasss.soc.surrey.ac.uk/12/1/1.html

8. Gilbert, N., Terna, P.: How to build and use agent-based models in social science. Mind and Society. 1 (2000) 57-72.
9. Minar, N., Burkhart, R., Langton, C., et al.: The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations. Santa Fe Institute Working Paper. 96-06-042 (1996).
10. Brown, T.N., Kulasiri, D.: Validating models of complex, stochastic, biological systems. Ecological Modelling. 86 (1996) 129-34.
11. López-Paredes, A., Saurí, D., Galán, J.M.: Urban water management with artificial societies of agents: The FIRMABAR simulator. Simulation. 81 (2005) 189-99.
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. Springer, Berlin Heidelberg (2005) 244-53.
13. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys. 37 (2005) 316-44.
14. OMG: Meta Object Facility (MOF) Core Specification, Version 2.0. http://www.omg.org.
15. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer. 39 (2006) 25-31.
16. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. In: Marik, V., Müller, J., Pechoucek, M. (eds). Multi-Agent Systems and Applications III. LNAI Vol. 2691. Springer-Verlag, Berlin Heidelberg (2003) 394-403.
17. Pavón, J., Gómez-Sanz, J., Fuentes, R.: Model driven development of multi-agent systems. In: Rensink, A., Warmer, J. (eds). Model Driven Architecture – Foundations and Applications. LNCS Vol. 4066, Berlin Heidelberg (2006) 284-98.
18. Aguilera, A., Ugalde, E.: A Spatially Extended Model for Residential Segregation. Discrete Dynamics in Nature and Society. 1 (2007) 48589.
19. Benenson, I., Torrens, P.M. Geosimulation: automata-based modeling of urban phenomena. John Wiley and Sons, Chichester, UK (2004).
20. Benenson, I., Omer, I., Hatna, E.: Entity-based modeling of urban residential dynamics: the case of Yaffo, Tel Aviv. Environment & Planning B. 29 (2002) 491-512.
21. Benenson, I.: Agent-Based Modeling: From Individual Residential Choice to Urban Residential Dynamics. In: Goodchild, M.F., Janelle, D.G. (eds). Spatially Integrated Social Science: Examples in Best Practice. Oxford University Press, Oxford, UK (2004) 67-95.
22. Wolpert, J.: Behavioral Aspects of the Decision to Migrate. Papers and Proceedings of the Regional Science Association. 15 (1965) 159-69.
23. Speare, A.: Residential satisfaction as an intervening variable in residential mobility. Demography. 11 (1974) 173-88.
24. Galán, J.M., del Olmo, R., López-Paredes, A.: Diffusion of Domestic Water Conservation Technologies in an ABM-GIS Integrated Model. In: Corchado, E., Abraham, A., Pedrycz, W. (eds). HAIS '08: Proceedings of the 3rd international workshop on Hybrid Artificial Intelligence Systems. LNAI Vol. 5271. Springer, Berlin Heidelberg (2008) 567-74.
25. Galán, J.M., López-Paredes, A., del Olmo, R.: An agent based model for domestic water management in Valladolid metropolitan area. Water Resources Research. 45 W05401 (2009) doi:10.1029/2007WR006536.
26. Kubera, Y., Mathieu, P., Picault, S.: How to Avoid Biases in Reactive Simulations. In: Demazeau, Y., Pavón, J., Corchado, J.M., et al. (eds). Advances in Intelligent and Soft Computing. Vol. 55. Springer, Berlin / Heidelberg (2009).
27. Benenson, I.: Multi-Agent Simulations of Residential Dynamics in the City. Computing, Environment and Urban Systems. 22 (1998) 25-42.
28. Benenson, I.: Modeling population dynamics in the city: from a regional to a multi-agent approach. Discrete Dynamics in Nature and Society. 3 (1999) 149-70.
29. North, M.J., Collier, N.T., Vos, J.R.: Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit. ACM Transactions on Modeling and Computer Simulation. 16 (2006) 1-25.