A Method for Developing Agent-based Models of Socio-technical Systems

Igor Nikolic and Amineh Ghorbani

***Abstract*— Agent-based modeling is one of the popular tools for analyzing complex socio-technical systems. Because of the complex nature of such systems a systematic methodology is required to guide the modeling process. By studying the existing methodologies in MAS we distinguished four major differences between MAS and ABM regarding goals, system scale and diversity, level of system understanding and verification and validation concerns. In this paper we take these differences into account and based on more than 25 case studies, we present a methodological framework for developing agent-based models that consists of five general iterative phases. These phases namely: system analysis, model design, detailed design, implementation and evaluation further consists of smaller step that are also addressed in this paper. This methodology provides a tool independent template while respecting the specific requirements for ABM.**

I. Introduction

A. Large Scale Socio-Technical systems as Complex Adaptive Systems

Large Scale Socio-Technical Systems [1] (LSSTS) are class of systems that span technical artifacts embedded in a social network. LSSTS include social elements such as operating companies, investors, local and national governments, regional development agencies, non-governmental organizations, customers and institutions. These develop around, sustain and depend on particular technical systems, be it a single plant, industrial complex or set of interconnected supply-chains. Examples of LSSTS are regional industrial clusters, power grids, multimodal transport networks and telecommunication networks [2].

LSTSS can be considered as Complex Adaptive Systems (CAS) [3]. In complex system theory [4], the central principle is that all phenomena can be described by interconnected networks of simple units. This approach, also called Generative Science [5], is an agent-based approach, that describes complex behaviors as a generative process from the bottom-up. In this approach, deterministic, finite rules and parameters interact to generate indeterministic and infinite system behavior. In generative science, the agent or individual component, is the theory, as it is often said. By having an explicit behavioral model of an individual, or the smallest level component, we can build/generate explanations of the observed phenomena. If there is to be any coherent behavior in the system, it has to arise from competition and cooperation among the agents. The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents [6].

Agent-Based Models (ABM) are constructed to discover possible emergent properties from a bottom-up perspective. ABM acknowledges that reality consists of many components acting in parallel. ABM describes these entities and lets them interact in parallel, observing all possible interaction modes. There is no desired state or task that needs to be achieved, only an exploration of the system's possible states. Some typical examples of ABMs are: a model of farmers' adaptations to climate change by [7], a model of the co-evolution of auto-catalytic economic production and economic firms by [8], a model of an abstract economy by [9] and a model of investments in power generation under different CO_2 policies [10].

B. Modeling LSSTS with ABM

Creating models of CAS such as LSSTS is not trivial. Developing ABM is not just writing computer code, but a complex, evolving socio-technical process which needs to be systematically approached. Most previous works on agent-based methodologies mainly focus on the evolving nature of the modeling process [3] or the social process of knowledge acquisition [11], [12]. Such methods usually give general instructions and do not explicitly define the steps that need to be taken throughout the modeling process [13]. On the other hand, some of the more formal approaches include methods for social simulation which do not give guidelines for agent development [14].

All the approaches mentioned above can be sufficient as long as the complexity of the agent-based models and associated computer code generated are relatively low. With increasing complexity of the models however, comes a rapid increase in the complexity of the software which requires systematic software methodologies. A survey on agent-based modeling practices also shows the lack of an agent-based methodological framework that can be used as a tool independent of software to guide social scientists in their modeling practices [15].

In order to find suitable methodologies for developing agent-based models, the multi-agent systems(MAS) and software engineering literature are the most relevant fields to explore. MAS is a field that acknowledges a given control problem (e.g. traffic control, agenda synchronization, etc.) which must be solved using discrete, parallel, autonomous components, namely agents. This is in contrast to agent-based modeling whose goal is to explore the emergent property of systems. Typical examples of MAS in the litera-

Both authors are with the Faculty of Technology, Policy and Management, Section Energy and Industry, TU Delft, Postbus 5015, 2600 GA, Delft i.nikolic@tudelft.nl

ture are: e-commerce agents [16], predictive control system for transportation networks [17] and cooperative agents in medical systems [18].

While MAS methodologies [19], [20], [21] are highly developed they are not always suitable for ABM purposes. Four main reasons are identified for this inadequacy namely: difference in goal, system scale and diversity, level of system understanding, and verification and validation issues.

a) difference in goal: The goal of ABM is to explore the emergent behavior arising from agent interactions, whereas the goal of MAS is to create an agent-based software system that would be used in different domains from the daily life of people to particular control problems. This has several important consequences. Because of its goals, MAS is concerned about the efficiency of the software and therefore about the efficiency of the agents and their intelligence. ABM on the other hand, is aimed at representing real world entities with all of their potential faults and irrational reasoning.

b) system scale and agent diversity: The scale and diversity of a MAS is defined through its distributed nature of elements. The definition of the system in MAS is more limited given its purposes. For example, even though there might be millions of agents interacting in an on-line market, the MAS system definition is only a market. While in the case of ABM, markets, factories and government may all be part of the system and need to be represented, giving a more complex and diverse system definition.

c) level of system understanding: The exploratory nature of ABM and the scale and diversity of the system under study necessarily means that we do not understand the system well. This lack of understanding requires a formal and structured approach to knowledge extraction from stakeholders. This process is crucial for the model creation. However for a MAS, given the goals and system scales it usually deals with, this process is not as difficult.

d) verification and validation issues: Both ABM and MAS approaches require thorough verification and validation. However given the difference in goals and scale of systems studied, the method of validation and verification are largely different. In MAS, aspects such as robustness and time efficiency may be crucial while in ABM closeness to reality whether quantitative and qualitative are essential.

In conclusion, having identified the differences above, even though MAS and software methodologies answer many of the issues for developing ABM they are not sufficient. In this paper, we present an ABM development methodology that builds upon the existing MAS and software engineering methodologies and extends them for ABM purposes. This extension is based on developing more than 25 different models of LSSTS ranging from a model of a multi-modal freight transport system [22], to a bio-electricity investment model [23] to a model of transitions in global LNG markets [24] [1].

[1]For more information about the 25 agent-based models please contact the authors.

II. Agent-based modeling methodology

The methodological framework presented in this paper consists of five iterative steps namely: system analysis, model design, detailed model design, software implementation and model evaluation. These steps are common to many software engineering methodologies. Nonetheless, each of these steps have several iterative sub-steps that are specific to ABM. Figure 1 schematically presents the overall framework.

A. System analysis

Before starting to build a model, it should be stressed that modeling is a means to an end and not a goal in itself. In the system analysis phase of ABM, the system being addressed in studied independent of software modeling and agent thinking.

1) Problem and Problem Owner Identification: The model is expected to increase the understanding of how the system functions, and give insight into how certain actions will influence the systems behavior.

To address this issue, the problem has to be well-formulated and the right questions have to be asked, in close contact with the problem owner(s)[2]. The problem formulation step should therefore address the following questions:

- What is the problem being addressed? What is the emergent pattern that is of interest to us?
- Why is this a problem? Is this pattern an existing or a desired one?
- Whose problem are we addressing?
- What is the intended model use? Before a working model is ready, it should be decided how the model will be used. Are the stakeholders interpreting the model outcomes themselves or are the modelers involved in the usage? To what end are the model outcomes going to be used (e.g. are we making policy, are we analyzing existing policy or are we forming hypothesis about the functioning of the system?)

2) System Identification: The next step towards building the model of a system, after the problem has been formulated, is deciding what the system entails, what its boundaries are and what it is composed of. This is inherently a social process as these systems are usually very large consisting of many social and technological components. The involved stakeholders have biased and partial perspective of the system and none of them have the complete image of the system. The challenge posed to the social process is to combine this partial and inherently biased stakeholder knowledge into one coherent system definition.

The goal of this step is to identify the internal structure of the system under analysis in such a manner that complex analysis of the system becomes possible. This means that the system is not only considered as a collection of actors and interactions that exist in the current situation of the system,

[2]Problem owner is the actor (i.e. a person or an organization) who is interested and has means and power to solve a specific problem. The problem owner can be considered as the party giving the assignment to a team of modelers.

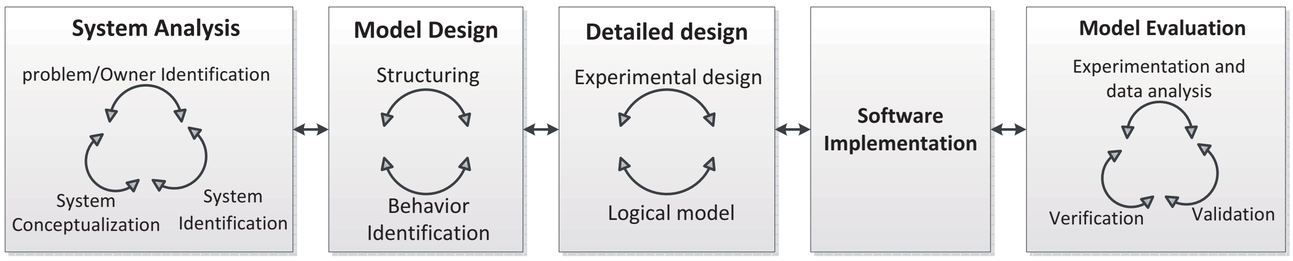

Fig. 1: A methodological framework for agent-based modeling

but that we take into account which elements might change over time and at what speed.

The output of this approach will be as follows:

1) List of actors and their behavior.
2) List of relationship between the actors.
3) A specification of the actions performed by the actors.
4) A description of the environment in which the actors are performing in.

The system identification phase is an inventory of system components. The following recursive steps are taken to obtain the information:

1) Choose a time frame that is important.
2) Interview and brainstorm with domain experts and stakeholders. Let the experts talk about the system in relation with the problem and make an inventory of all relevant, such as concepts, actors, objects, interactions, states, properties, flows, etc.
3) Perform a literature study on the system under study to spot any omissions.

Another important decision that needs to be taken now is what are meaningful metrics that help answer the problem owners questions. In general there are three types of metrics that need to be considered namely: individual agent properties over time, agent interaction data (over time) and emergent property metrics. These metrics are fully dependent on the intended model use and problem definition.

3) System Conceptualization: As the components of the system are identified, they also need to be decomposed into a structure that is manageable and understandable. In [25], the IAD framework and Williamson's four layer model which are both frameworks for institutional analysis are used to decompose the system into components for modeling agent-based social systems. One may also use UML analysis diagrams such as use case diagrams and interaction diagrams for similar purposes. Currently, we are also developing an agent-based methodology, namely MAIA [26] which is especially designed for the conceptualization phase of agent-based model development.

Ontology Formalization. After system identification and conceptualization have been completed, concepts are in natural language. There may be different terms used by different domain experts for the same concept which causes confusion about the logic of the system. Also, these terms are not suitable for interpretation by a computer. A domain ontology would also give an explicit, formal, and computer understandable definition of terms that can later be used in the detailed design phase. This also makes it possible for the system description to be generalized beyond one specific domain.

B. Model Design

Up to now, the focus of the modeling process was on the understanding of the system to be modeled. In the design phase however, it is now time to think in terms of a software model and identify the artificial agents and their interactions. This step is very similar to system identification in the analysis phase as it gives us the blue print of 'what' we will be modeling but not 'how' to implement the model.

1) Structuring: Entities including the actors identified in the system identification phase are classified as different types of agents. Agents are the basic units of the model. They are recognized by their boundaries, behavior (active, proactive, reactive) and ability to interact. The previously identified interactions are used to link different agents.

Note which agents and interactions are dynamic, and which are static. Add hierarchy to the agent components by ordering them in a hierarchical, nested way, as a box within a box. Also, add hierarchy to the interaction components by classifying them from abstract to concrete.

In this step the world outside the agents is determined by grouping all the system components that cannot be influenced by the other subcomponents. This might result in a different environment than the one that was observed in system analysis. This is because we are implementing the model based on a certain problem which results in more focus on one entity (and/or timing frame) and less on the other. They form the external world or the *environment*. The following rules of thumb can be used to determine what system elements are to be considered as the environment:

1) Things that are not influenced by components within the system are part of the environment.
2) Extremely slow processes (relative to the chosen time frame) should be considered to be a part of the environment.

2) Behavior Identification: Once the structure of the system is available, the behavior of the artificial entities as well as the environment need to be documented. This behavior identification is partly from the conceptualization phase and partly the result of the artificial skeleton we have designed in the structuring phase.

C. Detailed Model Design

On the analysis phase of the framework, the problem and the system that the problem holds in, are conceptualized independent of agent-based modeling. In the model design phase, our knowledge of the real world system is mapped into artificial agent-based model with an abstract representation. In the detailed design phase, the details required to program an actual model are specified.

1) Logical model formalization: The main questions here is, how do we make sure that the identified concepts can be understood by a computer, while retaining their originally intended meaning. Most computer languages can deal with a fairly limited set of basic concepts, from which all others are derived from (e.g. objects, classes, string, boolean, lists etc.). Essentially, any concept that must be formalized, as a (nested) combination of these basic elements. This is especially challenging with relatively ill-defined concepts such as trust or risk. Is trust a number from 0 to 100, or is it an elaborate hierarchically structured construct consisting of many other objects and their relationships? These design choices will be based on the detailed understanding we have gained up to now in our modeling process.

Most of this programming details can be added to the formal ontology developed in the analysis phase, the following two steps have to be taken as part of the formalization to use and refine the ontology:

1) Refine the ontology (which could also be an existing generic ontology) by creating new abstract classes applicable for this case and by adding properties to already existing classes.
2) Make the model specification by creating concrete instances of the abstract classes from the ontology.

2) Experimental design: The main question we need to answer is *What is it that we want to measure?* Following the problem definition, we define the desired outcomes of the model. Based on this question, we can form our hypothesis as : *Does (and under which conditions) the designed ABM emerge the macroscopic regularity that is of interest ?* We can also ask the questions *Given the ABM, what is the range of behaviors that it is capable of?*

Some of the most used experimental options available are:

a) Runs over time.: In some cases, we have exact parameter values, observed in reality. We may wish to explore the models behavior at that point, and watch what happens over time. Two main questions need to be answered in this case.

1) *How long to run*. The length of the run is in some cases dictated by the model. However, in models where the state does not clearly converge to a stable attractor, we need to experiment with long runs to get a sense of what is going on.
2) *How many times is a single run repeated?* And how representative is a single mode run? Given the randomized order of agents actions, a single run should never be considered as a result, other than serving as a illustrative example and many repetitions should be performed at every parameter point examined. Using the standard statistical methods, we can estimate the necessary number of repetitions.

b) Scenario design: Scenario is in essence nothing more than a single or a group of time runs under certain parameter values. It is a narrative of what can/should happen in the system that is being modeled. Scenarios may also contain dynamic profiles for some parameter values over time. They should be used to identify the scope of the possible parameter space, and aid the creation of a parameter sweep experiment, rather than the sole experiments to perform with the model.

c) Parameter Sweeps: Experiments are points in the parameter space. We have to decide how big the space is and how granular the analysis needs to be ? Approaches for defining parameter sweeps include: full factorial experiments where all dimensions and all values are considered, thus making it only applicable to small models; Latin hypercube sampling, which is a way to deal with very large parameter spaces and still have a good granularity and Monte Carlo analysis.

D. Software Implementation

Once the details of the model are designed, the program code needs to be implemented. Because of the iterative process of building a model just like any other software system, concepts and design may change according to the outcomes of the implementation or the issues that rise in this phase.

E. Model Evaluation

Evaluation is not a process that starts after the model is implemented but is a continuous process throughout modeling. Different types of evaluation are briefly introduced in this section.

1) Verification: Verification is about answering the question " Does the model do what I wanted it to do?" During the verification the model is checked against its conceptual design to see if all relevant entities and relationships from the conceptual model were correctly translated into the computational model. Verification in software development is always a difficult task, but particularly so when dealing with the complex software developed for agent-based modeling because of the high number of elements, interactions, and possibly surprising behavior.

There are four classes of inputs to the modeling process that need to be verified. These are presented in Table I.

TABLE I: Types of inputs and ease of verification

	Physical	Social
Facts	Easy. Objective, Measurable	Medium. Subjective, Measurable
Knowledge	Medium. Objective, Formalized	Hard. Subjective, Unformalized

We can make a distinction between facts and knowledge and between physical and social domains. Knowledge is defined in this case as the codified experience of its users [27]. Physical facts are easy to verify. One can objectively measure them. Physical knowledge is more difficult, since it represents an encoding of physical laws of nature into a knowledge structure of the domain experts. Knowledge needs to be formalized in order to be used in the modeling method. However, since it is based on physical reality, it is relatively easy to verify the encoding. Social facts are difficult to verify, as they are not objectively defined; they are dependent on the social context from which they are taken. Since they are measurable, their verifiability is medium. The most difficult aspect to verify is social knowledge. Not only does this knowledge represent a codification inside the domain expert's mind, but it is also dependent on a subjective experience of social reality.

Verification needs to be performed at two levels. At the multi formal knowledge level, it is important to verify that the knowledge encoded is what the domain experts have contributed. At the simulation level, it needs to be verified that the simulation code corresponds with the knowledge collected.For surprising behavior one always has to wonder if this is really something new they are learning from the model about the system, or the result caused by an error in the code they wrote for the model. Three phases are considered for verification of agents:

- *Single-agent testing* in which the behavior of one agent is verified includes: theoretical prediction or sanity checks, extreme value testing and dynamic signal testing. The first test is examining a single agents outputs to normal operating inputs. In the second test, parameters are given extreme values (i.e. very large (+infinity), very small (-infinity), 0, and on the edges of the parameter space the agent will experience). If the agents respond with a logical response to these extremes, we have further verified the agents. In the final set of tests, in cases where agents act on time series or have a notion of memory, the agent should be tested with different(extreme) time signals. Examples include random signals, signals with continuous in/decreasing values, signals with step functions and power law distribution of values.
- *Interaction testing, limited to minimal model*. In this phase, the interactions between agents are tested. The model is set up with the minimal number of agents necessary for the model to run, and we examine whether the basic agent interactions happens correctly.
- *Multi-agent testing*. In which the emergent behavior of multiple agents is studied. This includes variability testing and time-line sanity checks. Variability testing is exploring the variability of the outcomes at a certain point in the parameter space which may uncover rare interaction order artifacts. This is done by performing many (100-1000) repetitions of the model and examining the statistics of the outcomes, across a number of output variables. In the cases where outcomes display severe skewness and high positive kurtosis may suggest rare but large outliers. The time-line of the model run should also be examined at several representative parameter settings.

2) Validation: Traditionally, the validation of models is done by performing experiments. If the modeled outcomes correspond with observed reality, the model is validated. In the case of making some prediction about the future state of the world after some change has been implemented, the validating experiment would consist of making such change in the real world, and observing the effects. However, running such an experiment at the scale of LSSTS is clearly impossible. That means that validation is impossible in the traditional sense.

However, there are three main paths for validation of ABM outcomes: historic replays of system development, scenario testing through expert consultations and modeling method validation. In the first path, we attempt to recreate a current observed pattern by taking a (somewhat) known starting point in the past, together with the change of environmental conditions over time, in order to come to a state of a LSSTS observed today. In the second path we explore interesting development scenarios for the LSSTS evolution and discuss these with experts/stakeholders.

Finally, the method of creating models of LSSTS can be validated. It can be repeated with different stakeholders and domain experts, and the resulting models can be compared. Since the goal of social validation is to provide insight and increased knowledge, the approach is validated by the stakeholders themselves. The approach is therefore validated at the moment that stakeholders believe that they have increased their knowledge about the evolutionary processes and the possible states of the system. Traditional social science tools, such as expert consultations and interviews, can be used to judge this validity. It is also worth pointing out that more and more statistical methods are being used to validate ABM which cannot be discussed due to space limitations.

3) Experimentation and data analysis: Given the exploratory nature of ABM and complexity of LSSTS the parameter space is normally extremely large. As discussed in the experimental design section some of this space can be reduced but there is still need for potentially large number of experiments (i.e. model runs) that need to be executed. This can pause a potentially big computational challenge with experimental runs taking days to weeks to complete. One should then explore options for distributed model runs which brings with itself an entirely new set of challenges.

The consequence of having large number of experiments results in extremely large datasets. This presents both a practical problem of handling this amount of data and also a fundamental problem of how to find relevant outcomes. It is worth mentioning that even if we have all the available metrics, because of the high dimensionality of the parameter space it can be very challenging to represent outcomes in an insightful way.

III. Conclusions and discussion

We started the discussion in this work by stating that LSSTS are complex adaptive system, whose emergent properties need to be studied using agent-based modeling. We identified a need for a systematic methodology for model development. By studying the literature we concluded that existing ABM methodologies are inadequate, and subsequently turned to agent-based software development methodologies. While closely related, MAS approaches differ in four significant ways from ABM, namely they differ in goals, system scale and agent diversity, the level of system understanding and in verification & validation requirements.

Using our extensive empirical experience with developing ABM, we extended software methodologies to suit ABMs particular requirements. We presented a 5 phase modeling method, consisting of system analysis, model design, detailed model design, software implementation and model evaluation. These phases individually cover recursive steps that are specific to agent-based modeling.

This modeling methodology is a blue print which is sufficiently generic to allow any type of agent-based social simulation. It is also independent of tool which was also pointed out by [15] as a gap for agent-based methods. The methodology is the result of our extensive empirical use which shows that the method is highly applicable to many different cases.

Due to space limitations, we were not able to discuss the practical details of the steps. As with any other methodology, the presented work cannot guarantee high quality model since the outcome depends on the initial process of collective knowledge.

This work is a part of the ongoing process of standardizing our modeling practices. We are further elaborating the conceptualization and design phase by explicitly incorporating the institutional aspects which are inherently present in all LSSTS.

References

[1] W. Bijker, T. Hughes, and T. Pinch, *The Social construction of technological systems : new directions in the sociology and history of technology*. MIT Press. Cambridge, Mass, 1987.

[2] G. P. J. Dijkema and L. Basson, “Complexity and industrial ecology. foundations for a transformation from analysis to action,” *Journal of Industrial Ecology*, vol. 13, no. 2, pp. 157–164, 2009.

[3] I. Nikolic, “Co-evolutionary method for modelling large scale socio-technical systems evolution,” Ph.D. dissertation, Delft University of Technology, 2009.

[4] S. Kaufman, *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. New York: Oxford University Press, 1995.

[5] J. Epstein, “Agent-based computational models and generative social science,” *Complexity*, vol. 4, no. 5, pp. 41–60, 1999.

[6] M. Waldorp, *Complexity: The emerging science at the edge of order and chaos*. Simon and Schuster, New York, 1992.

[7] S. Schneider, W. Easterling, and L. Mearns, “Adaptation: Sensitivity to natural variability, agent assumptions and dynamic climate changes,” *Clim. Change*, vol. 45, no. 1, pp. 203–221, 2000.

[8] J. Padgett, D. Lee, and N. Collier, “Economic production as chemistry,” *Ind. Corp. Change*, vol. 12, no. 4, pp. 843–877, 2003.

[9] S. Kauffman, “Reinventing the Sacred: A New View of Science, Reason and Religion,” 2008.

[10] E. Chappin and G. Dijkema, “On the impact of CO_2 emission-trading on power generation emissions,” *Technological Forecasting & Social Change*, vol. 76, no. 3, pp. 358–370, March 2009. [Online]. Available: http://dx.doi.org/10.1016/j.techfore.2008.08.004

[11] I. Nikolic, P. Beers, and G. Dijkema, “Facilitating interdisciplinary modelling of complex problems,” in *Proceedings of HICSS-40*, ser. Hawaii Internatinal Conference On Systems Science. University of Hawaii at Manoa, January 3-6 2007.

[12] A. Ramanath and N. Gilbert, “The design of participatory agent-based social simulations,” *Journal of Artificial Societies and Social Simulation*, vol. 7, no. 4, p. 1, 2004.

[13] D. Phan and F. Varenne, “Agent-based models and simulations in economics and social sciences: from conceptual exploration to distinct ways of experimenting,” *Journal of Artificial Societies and Social Simulation*, vol. 13, no. 1, p. 5, 2010.

[14] S. Rossiter, J. Noble, and K. R. Bell, “Social simulations: Improving interdisciplinary understanding of scientific positioning and validity,” *Journal of Artificial Societies and Social Simulation*, vol. 13, no. 1, p. 10, 2010. [Online]. Available: http://jasss.soc.surrey.ac.uk/13/1/10.html

[15] B. Heath, R. Hill, and F. Ciarallo, “A survey of agent-based modeling practices (january 1998 to july 2008),” *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 4, p. 9, 2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/4/9.html

[16] R. Glushko, J. Tenenbaum, and B. Meltzer, “An xml framework for agent-based e-commerce,” *Communications of the ACM*, vol. 42, no. 3, p. 106, 1999.

[17] R. Negenborn, B. De Schutter, and H. Hellendoorn, “Multi-agent model predictive control of transportation networks,” in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC 2006)*, 2006, pp. 296–301.

[18] G. Lanzola, L. Gatti, S. Falasconi, and M. Stefanelli, “A framework for building cooperative software agents in medical applications,” *Artif. Intell. Med.*, vol. 16, no. 3, pp. 223–249, 1999.

[19] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, “Tropos: An agent-oriented software development methodology,” *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[20] V. Dignum, “A model for organizational interaction: based on agents, founded in logic,” Ph.D. dissertation, 2004.

[21] M. Wooldridge, N. Jennings, and D. Kinny, “The gaia methodology for agent-oriented analysis and design,” *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.

[22] A. Sirikijpanichkul, D. VAN, *et al.*, “Optimizing the location of intermodal freight hubs: an overview of the agent based modelling approach,” *Journal of Transportation Systems Engineering and Information Technology*, vol. 7, no. 4, pp. 71–81, 2007.

[23] C. Davis, I. Nikolic, and G. Dijkema, “Integration of Life Cycle Assessment Into Agent-Based Modeling. Toward Informed Decisions on Evolving Infrastructure Systems,” *Journal of Industrial Ecology*, vol. 13, no. 2, pp. 306–325, 2009.

[24] E. J. L. Chappin, R. Praet, and G. P. J. Dijkema, “Innovation and transition of global lng markets,” in *11th International Conference on Technology Policy and Innovation - ICTPI ’08*, New Delhi, India, 2009.

[25] A. Ghorbani, A. Ligtvoet, I. Nikolic, and G. Dijkema, “Using institutional frameworks to conceptualize agent-based models of socio-technical systems,” *Proceeding of the 2010 workshop on complex system modeling and simulation*, vol. 3, pp. 33–41, 2010.

[26] A. Ghorbani, V. Dignum, and D. Dijkema, “Maia: An analysis and design framework for building artificial societies,” in *Submitted :Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*.

[27] M. Stefik, *Introduction to Knowledge Systems*. Morgan Kaufmann, 1995.