Chapter 6
Checking Simulations: Detecting and Avoiding Errors and Artefacts

José M. Galán, Luis R. Izquierdo, Segismundo S. Izquierdo, José I. Santos, Ricardo del Olmo, and Adolfo López-Paredes

Why Read This Chapter? Given the complex and exploratory nature of many agent-based models, checking that the model performs in the manner intended by its designers is a very challenging task. This chapter helps the reader to identify some of the possible types of error and artefact that may appear in the different stages of the modelling process. It will also suggest some activities that can be conducted to detect, and hence avoid, each type.

Abstract The aim of this chapter is to provide the reader with a set of concepts and a range of suggested activities that will enhance his or her ability to understand agent-based simulations. To do this in a structured way, we review the main concepts of the methodology (e.g. we provide precise definitions for the terms 'error' and 'artefact') and establish a general framework that summarises the process of designing, implementing, and using agent-based models. Within this framework we identify the various stages where different types of assumptions are usually made and, consequently, where different types of errors and artefacts may appear. We then propose several activities that can be conducted to detect each type of error and artefact.

J.M. Galán (✉) • L.R. Izquierdo • J.I. Santos • R. del Olmo
Departamento de Ingeniería Civil, Universidad de Burgos, Burgos E-09001, Spain
e-mail: jmgalan@ubu.es; luis@izquierdo.name; jisantos@ubu.es; rdelolmo@ubu.es

S.S. Izquierdo • A. López-Paredes
Departamento de Ingeniería Civil, Universidad de Burgos, Valladolid E-47011, Spain
e-mail: segis@eis.uva.es; adolfo@insisoc.org

6.1 Introduction

Agent-based modelling is one of multiple techniques that can be used to conceptualise social systems. What distinguishes this methodology from others is the use of a more direct correspondence between the entities in the system to be modelled and the agents that represent such entities in the model (Edmonds 2001). This approach offers the potential to enhance the transparency, soundness, descriptive accuracy, and rigour of the modelling process, but it can also create difficulties: agent-based models are generally complex and mathematically intractable, so their exploration and analysis often require computer simulation.

The problem with computer simulations is that understanding them in reasonable detail is not as straightforward an exercise as one could think (this also applies to one's own simulations). A computer simulation can be seen as the process of applying a certain function to a set of inputs to obtain some results. This function is usually so complicated and cumbersome that the computer code itself is often not that far from being one of the best descriptions of the function that can be provided. Following this view, understanding a simulation would basically consist in identifying the parts of the mentioned function that are responsible for generating particular (sub)sets of results.

Thus, it becomes apparent that a prerequisite to understand a simulation is to make sure that there is no significant disparity between what we think the computer code is doing and what is actually doing. One could be tempted to think that, given that the code has been programmed by someone, surely there is always at least one person – the programmer – who knows precisely what the code does. Unfortunately, the truth tends to be quite different, as the leading figures in the field report:

> You should assume that, no matter how carefully you have designed and built your simulation, it will contain bugs (code that does something different to what you wanted and expected). (Gilbert 2007)
>
> An unreplicated simulation is an untrustworthy simulation – do not rely on their results, they are almost certainly wrong. ('Wrong' in the sense that, at least in some detail or other, the implementation differs from what was intended or assumed by the modeller). (Edmonds and Hales 2003)
>
> Achieving internal validity is harder than it might seem. The problem is knowing whether an unexpected result is a reflection of a mistake in the programming, or a surprising consequence of the model itself. [...] As is often the case, confirming that the model was correctly programmed was substantially more work than programming the model in the first place. (Axelrod 1997a)

In the particular context of *agent-based* simulation, the problem tends to be exacerbated. The complex and exploratory nature of most agent-based models implies that, before running a model, there is almost always some uncertainty about what the model will produce. Not knowing a priori what to expect makes it difficult to discern whether an unexpected outcome has been generated as a legitimate result of the assumptions embedded in the model or, on the contrary, it is due to an error or an artefact created in its design, its implementation, or in the running process (Axtell and Epstein 1994: 31; Gilbert and Terna 2000).

Moreover, the challenge of understanding a computer simulation does not end when one is confident that the code is free from errors; the complex issue of identifying what parts of the code are generating a particular set of outputs remains to be solved. Stated differently, this is the challenge of discovering what assumptions in the model are causing the results we consider significant. Thus, a substantial part of this non-trivial task consists in detecting and avoiding artefacts: significant phenomena caused by accessory assumptions in the model that are (mistakenly) deemed irrelevant. We explain this in detail in subsequent sections.

The aim of this chapter is to provide the reader with a set of concepts and a range of suggested activities that will enhance his ability to understand simulations. As mentioned before, simulation models can be seen as functions operating on their inputs to produce the outputs. These functions are created by putting together a range of different assumptions of very diverse nature. Some assumptions are made because they are considered to be an essential feature of the system to be modelled; others are included in a somewhat arbitrary fashion to achieve completeness – i.e. to make the computer model run –, and they may not have a clear referent in the target system. There are also assumptions – e.g. the selection of the compiler and the particular pseudo-random number generator to be employed – that are often made, consciously or not, without fully understanding in detail how they work, but *trusting* that they operate in the way we think they do. Finally, there may also be some assumptions in a computer model that not even its own developer is aware of, e.g. the use of floating-point arithmetic, rather than real arithmetic.

Thus, in broad terms, understanding simulations requires identifying what assumptions are being made, and assessing their impact on the results. To achieve this, we believe that it is useful to characterise the process by which assumptions accumulate to end up forming a complete model. We do this in a structured way by presenting a general framework that summarises the process of creating and using agent-based models through various stages; then, within this framework, we characterise the different types of assumptions that are made in each of the stages of the modelling process, and we identify the sort of errors and artefacts that may occur; we also propose activities that can be conducted to avoid each type of error or artefact.

The chapter is structured as follows: the following section is devoted to explaining what we understand by modelling, and to argue that computer simulation is a useful tool to explore formal models, rather than a distinctively new symbolic system or a uniquely different reasoning process, as it has been suggested in the literature. In Sect. 6.3 we explain what the essence of agent-based modelling is in our view, and we present the general framework that summarises the process of designing, implementing, and using agent-based models. In Sect. 6.4 we define the concepts of error and artefact, and we discuss their relevance for validation and verification. The framework presented in Sect. 6.3 is then used to identify the various stages of the modelling process where different types of assumptions are made and, consequently, where different types of errors and artefacts may appear. We then propose various activities aimed at avoiding the types of errors and artefacts previously described, and we conclude with a brief summary of the chapter.

6.2 Three Symbolic Systems Used to Model Social Processes

Modelling is the art of building models. In broad terms, a model can be defined as an abstraction of an observed system that enables us to establish some kind of inference process about how the system works, or about how certain aspects of the system operate.

Modelling is an activity inherent to every human being: people constantly develop mental models, more or less explicit, about various aspects of their daily life. Within science in particular, models are ubiquitous. Many models in the "hard" sciences are formulated using mathematics (e.g. differential equation models and statistical regressions), and they are therefore formal, but it is also perfectly feasible – and acceptable – to build non-formal models within academia; this is often the case in disciplines like history or sociology – consider e.g. a model written in natural language that tries to explain the expansion of the Spanish Empire in the sixteenth century, or the formation of urban "tribes" in large cities.

We value a model to the extent that it is useful – i.e. in our opinion, what makes a model good is its fitness for purpose. Thus, the assessment of any model can only be conducted relative to a predefined purpose. Having said that, there is a basic set of general features that are widely accepted to be desirable in any model, e.g. accuracy, precision, generality, and simplicity (see Fig. 6.1). Frequently some of these features are inversely related; in such cases the modeller is bound to compromise to find a suitable trade-off, considering the perceived relative importance of each of these desirable features for the purpose of the model (Edmonds 2005).

Some authors (Gilbert 1999; Holland and Miller 1991; Ostrom 1988) classify the range of available techniques for modelling phenomena in which the social dimension is influential according to three symbolic systems.

One possible way of representing and studying social phenomena is through verbal argumentation in natural language. This is the symbolic system traditionally used in historical analyses, which, after a process of abstraction and simplification, describe past events emphasising certain facts, processes, and relations at the expense of others. The main problem with this type of representation is its intrinsic lack of precision (due to the ambiguity of natural language) and the associated difficulty of uncovering the exact implications of the ideas put forward in this way. In particular, using this symbolic system it is often very difficult to determine the whole range of inferences that can be obtained from the assumptions embedded in the model in reasonable detail; therefore it is often impossible to assess its logical consistency, its scope, and its potential for generalisation in a formal way.

A second symbolic system that is sometimes used in the Social Sciences, particularly in Economics, is the set of formal languages (e.g. leading to models expressed as mathematical equations). The main advantage of this symbolic system derives from the possibility of using formal deductive reasoning to infer new facts from a set of clearly specified assumptions; formal deductive reasoning guarantees that the obtained inferences follow from the axioms with logical consistency. Formal languages also facilitate the process of assessing the generality of a

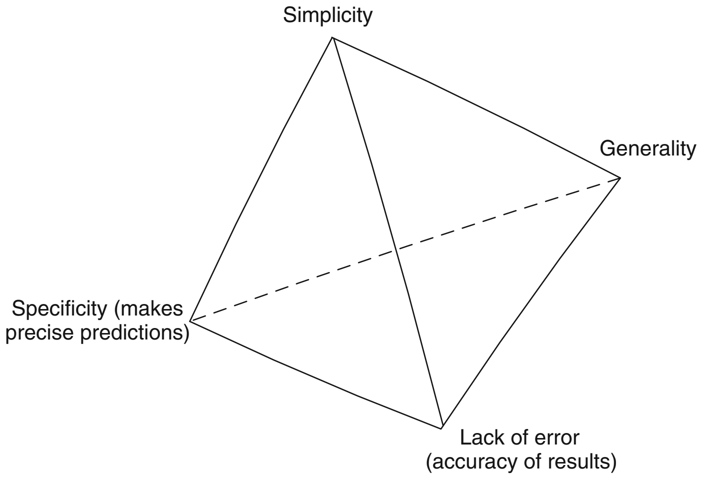
Simplicity
Generality
Specificity (makes precise predictions)
Lack of error (accuracy of results)

Fig. 6.1 The trade-off between various desirable features depends on the specific case and model. There are not general rules that relate, not even in a qualitative fashion, all these features. The figure shows a particular example from Edmonds (2005) that represents the possible equilibrium relationships between some features in a particular model

model and its sensitivity to assumptions that are allowed to change within the boundaries of the model (i.e. parameter values and non-structural assumptions).

However, the process of reducing social reality to formal models is not exempt from disadvantages. Social systems can be tremendously complex, so if such systems are to be abstracted using a formal language (e.g. mathematical equations), we run the risk of losing too much in descriptiveness. To make things worse, in those cases where it appears possible to produce a satisfactory formal model of the social system under investigation, the resulting equations may be so complex that the formal model becomes mathematically intractable, thus failing to provide most of the benefits that motivated the process of formalisation in the first place. This is particularly relevant in the domain of the Social Sciences, where the systems under investigation often include non-linear relations (Axtell 2000). The usual approach then is to keep on adding simplifying hypotheses to the model – thus making it increasingly restrictive and unrealistic – until we obtain a tractable model that can be formally analysed with the available tools. We can find many examples of such assumptions in Economics: instrumental rationality, perfect information, representative agents, etc. Most often these concepts are not included because economists think that the real world works in this way, but to make the models tractable (see for instance Conlisk (1996), Axelrod (1997a), Hernández (2004), Moss (2001, 2002)). It seems that, in many cases, the use of formal symbolic systems tends to increase the danger of letting the pursuit for tractability be the driver of the modelling process.

But then, knowing that many of the hypotheses that researchers are obliged to assume may not hold in the real world, and could therefore lead to deceptive conclusions and theories, does this type of modelling representation preserve its advantages? Quoting G.F. Shove, it could be the case that sometimes "*it is better to be vaguely right than precisely wrong*".

The third symbolic system, computer modelling, opens up the possibility of building models that somewhat lie in between the descriptive richness of natural language and the analytical power of traditional formal approaches. This third type of representation is characterized by representing a model as a computer program (Gilbert and Troitzsch 1999). Using computer simulation we have the potential to build and study models that to some extent combine the intuitive appeal of verbal theories with the rigour of analytically tractable formal modelling.

In Axelrod's (1997a) opinion, computational simulation is the third way of doing science, which complements induction –the search for patterns in data– and deduction –the proof of theorems from a set of fixed axioms. In his opinion, simulation, like deduction, starts from an explicit set of hypotheses but, rather than generating theorems, it generates data that can be inductively analysed.

While the division of modelling techniques presented above seems to be reasonably well accepted in the social simulation community –and we certainly find it useful–, we do not fully endorse it. In our view, computer simulation does not constitute a distinctively new symbolic system or a uniquely different reasoning process by itself, but rather a (very useful) tool for exploring and analysing formal systems. We see computers as inference engines that are able to conduct algorithmic processes at a speed that the human brain cannot achieve. The inference derived from running a computer model is constructed by example and, in the general case, reads: *the results obtained from running the computer simulation follow (with logical consistency) from applying the algorithmic rules that define the model on the input parameters*[1] *used*.

In this way, simulations allow us to explore the properties of certain formal models that are intractable using traditional formal analyses (e.g. mathematical analyses), and they can also provide fundamentally new insights even when such analyses are possible. Like Gotts et al. (2003), we also believe that mathematical analysis and simulation studies should not be regarded as alternative and even opposed approaches to the formal study of social systems, but as complementary. They are both extremely useful tools to analyse formal models, and they are complementary in the sense that they can provide fundamentally different insights on one same model.

To summarise, a computer program is a formal model (which can therefore be expressed in mathematical language, e.g. as a set of stochastic or deterministic equations), and computer simulation is a tool that enables us to study it in ways that go beyond mathematical tractability. Thus, the final result is a potentially more realistic – and still formal – study of a social system.

[1] By *input parameters* in this statement we mean "everything that may affect the output of the model", e.g. the random seed, the pseudo-random number generator employed and, potentially, information about the microprocessor and operating system on which the simulation was run, if these could make a difference.

6.3 Agent Based Modelling

6.3.1 Concept

As stated before, modelling is the process of building an abstraction of a system for a specific purpose (see Edmonds (2001) and Epstein (2008) for a list of potential applications). Thus, in essence, what distinguishes one modelling paradigm from another is precisely the way we construct that abstraction from the observed system.

In our view, agent-based modelling is a modelling paradigm with the defining characteristic that entities within the target system to be modelled – and the interactions between them – are explicitly and individually represented in the model (see Fig. 6.2). This is in contrast to other models where some entities are represented via average properties or via single representative agents. In many other models, entities are not represented at all, and it is only processes that are studied (e.g. a model of temperature variation as a function of pressure), and it is worth noting that such processes may well be already abstractions of the system.[2] The specific process of abstraction employed to build one particular model does not necessarily make it better or worse, only more or less useful for one purpose or another.

The specific way in which the process of abstraction is conducted in agent-based modelling is attractive for various reasons: it leads to (potentially) formal yet more natural and transparent descriptions of the target system, provides the possibility to model heterogeneity almost by definition, facilitates an explicit representation of the environment and the way other entities interact with it, allows for the study of the bidirectional relations between individuals and groups, and it can also capture emergent behaviour (see Epstein 1999; Axtell 2000; Bonabeau 2002). Unfortunately, as one would expect, all these benefits often come at a price: most of the models built in this way are mathematically intractable. A common approach to study the behaviour of mathematically intractable formal models is to use computer simulation. It is for this reason that we often find the terms "agent-based modelling" and "agent-based simulation" used as synonyms in the scientific literature (Hare and Deadman 2004).

Thus, to summarise our thoughts in the context of the classification of modelling approaches in the social sciences, we understand that the essence of agent-based modelling is the individual and explicit representation of the entities and their interactions in the model, whereas computer simulation is a useful tool for studying the implications of formal models. This tool happens to be particularly well suited to explore and analyse agent-based models for the reasons explained above. Running an agent-based model in a computer provides a formal proof that a particular micro-specification is *sufficient* to generate the global behaviour that is observed

[2] The reader can see an interesting comparative analysis between agent-based and equation-based modelling in (Parunak et al. 1998).

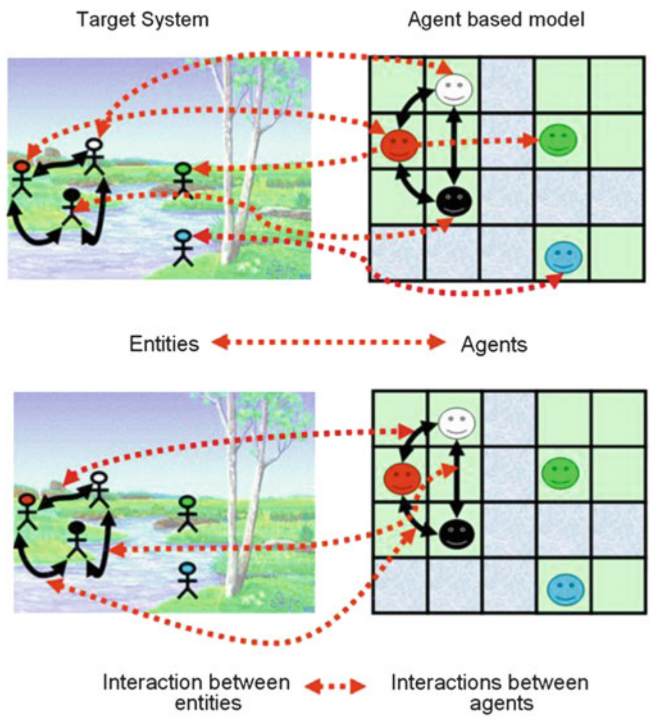

Fig. 6.2 In agent based modelling the entities of the system are represented explicitly and individually in the model. The limits of the entities in the target system correspond to the limits of the agents in the model, and the interactions between entities correspond to the interactions of the agents in the model (Edmonds 2001)

during the simulation. If a model can be run in a computer, then it is in principle possible to express it in many different formalisms, e.g. as a set of mathematical equations. Such equations may be very complex, difficult to interpret and impossible to solve, thus making the whole exercise of changing formalism frequently pointless, but what we find indeed useful is *the thought* that such an exercise *could* be undertaken, i.e. an agent-based model that can be run in a computer is not that different from the typical mathematical model. As a matter of fact, it is not difficult to formally characterise most agent-based models in a general way (Leombruni and Richiardi 2005).

6.3.2 *Design, Implementation, and Use of an Agent-Based Model*

Drogoul et al. (2003) identify three different roles in the design, implementation, and use of a typical agent-based model: the *thematician* (domain expert), the *modeller*, and the *computer scientist*. It is not unusual in the field to observe that

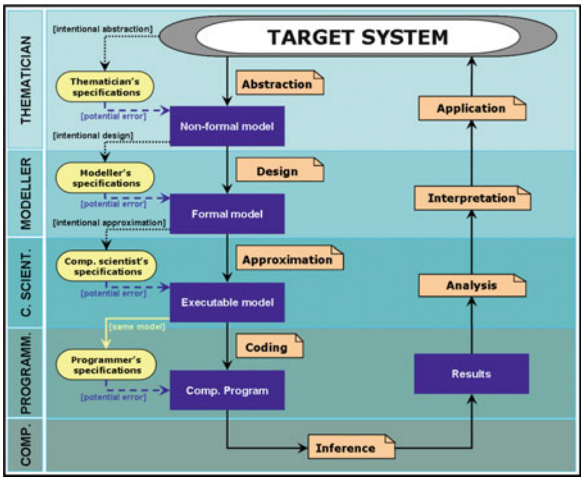

Fig. 6.3 Different stages in the process of designing, implementing and using an agent-based model

one single person undertakes several or even all of these roles. We find that these three roles fit particularly well into the framework put forward by Edmonds (2001) to describe the process of modelling with an intermediate abstraction. Here we marry Drogoul et al.'s and Edmonds' views on modelling by dissecting one of Drogoul et al.'s roles and slightly expanding Edmonds' framework (Fig. 6.3). We then use our extended framework to identify the different types of assumptions that are made in each of the stages of the modelling process, the errors and artefacts that may occur in each of them, and the activities that can be conducted to avoid such errors and artefacts. We start by explaining the three different roles proposed by Drogoul et al. (2003).

The role of the *thematician* is undertaken by experts in the target domain. They are the ones that better understand the target system, and therefore the ones who carry out the abstraction process that is meant to produce the first conceptualisation of the target system. Their job involves defining the objectives and the purpose of the modelling exercise, identifying the critical components of the system and the linkages between them, and also describing the most prominent causal relations. The output of this first stage of the process is most often a non-formal model expressed in natural language, and it may also include simple conceptual diagrams, e.g. block diagrams. The non-formal model produced may describe the system using potentially ambiguous terms (such as e.g. learning or imitation, without fully specifying how these processes actually take place).

The next stage in the modelling process is carried out by the role of the *modeller*. The modeller's task is to transform the non-formal model that the *thematician* aims to explore into the (formal) requirement specifications that the *computer scientist* – the third role – needs to formulate the (formal) executable model. This job involves (at least) three major challenges. The first one consists in acting as a mediator between two domains that are very frequently fundamentally different (e.g. sociology and computer science). The second challenge derives from the fact that in most cases the *thematician*'s model is not fully specified, i.e. there are many formal models that would conform to it.[3] In other words, the formal model created by the *modeller* is most often just one of many possible particularisations of the *thematician*'s (more general) model. Lastly, the third challenge appears when the *thematician*'s model is not consistent, which may perfectly be the case since his model is often formulated using natural language. Discovering inconsistencies in natural language models is in general a non-trivial task. Several authors (e.g. Christley et al. (2004), Pignotti et al. (2005), and Polhill and Gotts (2006)) have identified ontologies to be particularly useful for this purpose, especially in the domain of agent-based social simulation. Polhill and Gotts (2006) write:

> An ontology is defined by Gruber (1993) as "a formal, explicit specification of a shared conceptualisation". Fensel (2001) elaborates: ontologies are formal in that they are machine readable; explicit in that all required concepts are described; shared in that they represent an agreement among some community that the definitions contained within the ontology match their own understanding; and conceptualisations in that an ontology is an abstraction of reality. (Polhill and Gotts 2006, p. 51)

Thus, the modeller has the difficult –potentially unfeasible– task of finding a set of (formal and consistent) requirement specifications[4] where each individual requirement specification of that set is a legitimate particular case of the *thematician*'s model, and the set as a whole is *representative* of the *thematician*'s specifications (i.e. the set is sufficient to fully characterise the *thematician*'s model to a satisfactory extent).

Drogoul et al.'s third role is the *computer scientist*. Here we distinguish between *computer scientist* and *programmer*. It is often the case that the modeller comes up with a formal model that cannot be implemented in a computer. This could be, for example, because the model uses certain concepts that cannot be operated by present-day computers (e.g. real numbers, as opposed to floating-point numbers), or because running the model would demand computational requirements that are not yet available (e.g. in terms of memory and processing capacity). The job of the *computer scientist* consists in finding a suitable (formal) approximation to the *modeller*'s formal model that can be executed in a computer (or in several computers) given the available technology. To achieve this, the *computer scientist* may have to approximate and simplify certain aspects of the *modeller*'s formal model, and it is his job to make sure that these simplifications are not affecting the results significantly. As an example, Cioffi-Revilla (2002) warns about the potentially significant effects of altering system size in agent-based simulations.

The Navier–Stokes equations of fluid dynamics are a paradigmatic case in point. They are a set of non-linear differential equations that describe the motion of a fluid. Although these equations are considered a very good (formal and fully specified)

[3] Note that the *thematician* faces a similar problem when building his non-formal model. There are potentially an infinite number of models for one single target system.

[4] Each individual member of this set can be understood as a different model or, alternatively, as a different parameterisation of one single –more general– model that would itself define the whole set.

model, their complexity is such that analytical closed-form solutions are available only for the simplest cases. For more complex situations, solutions of the Navier–Stokes equations must be estimated using approximations and numerical computation (Heywood et al. 1990; Salvi 2002). Deriving such approximations would be the task of the *computer scientist*'s role, as defined here.

One of the main motivations to distinguish between the *modeller*'s role and the *computer scientist*'s role is that, in the domain of agent-based social simulation, it is the description of the *modeller*'s formal model what is usually found in academic papers, even though the *computer scientist*'s model was used by the authors to produce the results in the paper. Most often the *modeller*'s model (i.e. the one described in the paper) simply *cannot* be run in a computer; it is the (potentially faulty) implementation of the *computer scientist*'s approximation to such a model what is really run by the computer. As an example, note that computer models described in scientific papers are most often expressed using equations in real arithmetic, whereas the models that actually run in computers almost invariably use floating-point arithmetic.

Finally, the role of the *programmer* is to implement the *computer scientist*'s executable model. In our framework, by definition of the role *computer scientist*, the model he produces must be executable and fully specified, i.e. it must include all the necessary information so given a certain input the model always produces the same output. Thus, the executable model will have to specify in its definition everything that could make a difference, e.g. the operating system and the specific pseudo-random number generator to be used. This is a subtle but important point, since it implies that the *programmer*'s job does not involve any process of abstraction or simplification; i.e. the executable model and the *programmer*'s specifications are by definition *the same* (see Fig. 6.3). (We consider two models to be *the same* if and only if they produce the same outputs when given the same inputs.) The *programmer*'s job consists "only" in writing the executable model in a programming language.[5] If the *programmer* does not make any mistakes, then the implemented model (e.g. the code) and the executable model will be the same.

Any mismatch between someone's specifications and the actual model he passes to the next stage is considered here an error (see Fig. 6.3). As an example, if the code implemented by the programmer is not the same model as his specifications, then there has been an implementation error. Similarly, if the *computer scientist*'s specifications are not complete (i.e. they do not define a unique model that produces a precise set of outputs for each given set of inputs) we say that he has made an error since the model he is producing is necessarily fully specified (by definition of the role). This opens up the question of how the executable model is defined: the executable model is *the same model* as the code if the *programmer* does not

[5] There are some interesting attempts with INGENIAS (Pavón and Gómez-Sanz 2003) to use modelling and visual languages as programming languages rather than merely as design languages (Sansores and Pavón 2005; Sansores et al. 2006. These efforts are aimed at automatically generating several implementations of one single executable model (in various different simulation platforms).

make any mistakes. So, to be clear, the distinction between the role of *computer scientist* and *programmer* is made here to distinguish (a) errors in the implementation of a fully specified model (which are made by the *programmer*) from (b) errors derived from an incomplete understanding of how a computer program works (which are made by the *computer scientist*). An example of the latter would be one where the *computer scientist*'s specifications stipulate the use of real arithmetic, but the executable model uses floating-point arithmetic.

It is worth noting that in an ideal world the specifications created by each role would be written down. Unfortunately the world is far from ideal, and it is often the case that the mentioned specifications stay in the realm of mental models, and never reach materialisation.

The reason for which the last two roles in the process are called 'the *computer scientist*' and the '*programmer*' is because, as mentioned before, most agent-based models are implemented as computer programs, and then explored through simulation (for tractability reasons). However, one could also think of e.g. a mathematician conducting these two roles, especially if the formal model provided by the *modeller* can be solved analytically. For the sake of clarity, and without great loss of generality, we assume here that the model is implemented as a computer program and its behaviour is explored through computer simulation.

Once the computer model is implemented, it is run, and the generated results are analysed. The analysis of the results of the computer model leads to conclusions on the behaviour of the *computer scientist*'s model and, to the extent that the *computer scientist*'s model is a valid approximation of the *modeller*'s formal model, these conclusions also apply to the *modeller*'s formal model. Again, to the extent that the formal model is a legitimate particularisation of the non-formal model created by the *thematician*, the conclusions obtained for the *modeller*'s formal model can be interpreted in the terms used by the non-formal model. Furthermore, if the *modeller*'s formal model is representative of the *thematician*'s model, then there is scope for making general statements on the behaviour of the *thematician*'s model. Finally, if the *thematician*'s model is satisfactorily capturing social reality, then the knowledge inferred in the whole process can be meaningfully applied to the target system.

In the following section we use our extended framework to identify the different errors and artefacts that may occur in each of the stages of the modelling process and the activities that can be conducted to avoid such errors and artefacts.

6.4 Errors and Artefacts

6.4.1 Definition of Error and Artefact and Their Relevance for Validation and Verification

Since the meanings of the terms validation, verification, error, and artefact are not uncontested in the literature, we start by stating the meaning that *we* attribute to each of them. For us, validation is the process of assessing how useful a model is for a certain purpose. A model is valid to the extent that it provides a satisfactory range of accuracy consistent with the intended application of the model (Kleijnen 1995; Sargent 2003).[6] Thus, if the objective is to accurately represent social reality, then validation is about assessing how well the model is capturing the essence of its empirical referent. This could be measured in terms of goodness of fit to the characteristics of the model's referent Moss et al. (1997).

Verification – sometimes called "internal validation", e.g. by Taylor (1983), Drogoul et al. (2003), Sansores and Pavón (2005), or "internal validity", e.g. by Axelrod (1997a) – is the process of ensuring that the model performs in the manner intended by its designers and implementers (Moss et al. 1997). Let us say that a model is *correct* if and only if it would pass a verification exercise. Using our previous terminology, an expression of a model in a language is correct if and only if it is *the same* model as the developer's specifications. Thus, it could well be the case that a correct model is not valid (for a certain purpose). Conversely, it is also possible that a model that is not correct is actually valid for some purposes. Having said that, one would think that the chances of a model being valid are higher if it performs in the manner intended by its designer. To be sure, according to our definition of validation, what we want is a valid model, and we are interested in its correctness only to the extent that correctness contributes to make the model valid.

We also distinguish between errors and artefacts (Galán et al. 2009). *Errors* appear when a model does not comply with the requirement specifications self-imposed by its own developer. In simple words, an error is a mismatch between what the developer thinks the model is, and what it actually is. It is then clear that there is an error in the model if and only if the model is not correct. Thus, verification is the process of looking for errors. An example of an implementation error would be the situation where the *programmer* intends to loop through the whole list of agents in the program, but he mistakenly writes the code so it only runs through a subset of them. A less trivial example of an error would be the situation where it is believed that a program is running according to the rules of real arithmetic, while the program is actually using floating-point arithmetic (Izquierdo and Polhill 2006; Polhill and Izquierdo 2005; Polhill et al. 2005, 2006).

In contrast to errors, *artefacts* relate to situations where there is no mismatch between what the developer thinks a model is and what it actually is. Here the mismatch is between the set of assumptions in the model that the developer thinks are producing a certain phenomenon, and the assumptions that are the actual cause of such phenomenon. We explain this in detail. We distinguish between *core* and *accessory* assumptions in a model. *Core* assumptions are those whose presence is believed to be important for the purpose of the model. Ideally these would be the only assumptions present in the model. However, when producing a formal model it is often the case that the developer is bound to include some additional assumptions for the only purpose of making the model complete. We call these *accessory*

[6] See a complete epistemic review of the validation problem in Kleindorfer et al. (1998).

assumptions. Accessory assumptions are not considered a crucial part of the model; they are included to *make the model work*. We also distinguish between *significant* and *non-significant* assumptions. A *significant* assumption is an assumption that is the cause of some significant result obtained when running the model. Using this terminology, we define *artefacts* as significant phenomena caused by *accessory* assumptions in the model that are (mistakenly) deemed *non-significant*. In other words, an artefact appears when an accessory assumption that is considered non-significant by the developer is actually significant. An example of an artefact would be the situation where the topology of the grid in a model is accessory, it is believed that some significant result obtained when running the model is independent of the particular topology used (say, e.g. a grid of square cells), but it turns out that if an alternative topology is chosen (say, e.g. hexagonal cells) then the significant result is not observed.

The relation between artefacts and validation is not as straight-forward as that between errors and verification. For a start, artefacts are relevant for validation only to the extent that identifying and understanding causal links in the model's referent is part of the purpose of the modelling exercise. We assume that this is the case, as indeed it usually is in the field of agent-based social simulation. A clear example is the Schelling-Sakoda model of segregation, which was designed to investigate the causal link between individual preferences and global patterns of segregation (Sakoda 1971; Schelling 1971, 1978). The presence of artefacts in a model implies that the model is not representative of its referent, since one can change some accessory assumption (thus creating an alternative model which still includes all the core assumptions) and obtain significantly different results. When this occurs, we run the risk of interpreting the results obtained with the (non-representative) model beyond its scope (Edmonds and Hales 2005). Thus, to the extent that identifying causal links in the model's referent is part of the purpose of the modelling exercise, the presence of artefacts decreases the validity of the model. In any case, the presence of artefacts denotes a misunderstanding of what assumptions are generating what results.

6.4.2 Appearance of Errors and Artefacts

The dynamics of agent-based models are generally sufficiently complex that model developers themselves do not understand in exhaustive detail how the obtained results have been produced. As a matter of fact, in most cases if the exact results and the processes that generated them were known and fully understood in advance, there would not be much point in running the model in the first place. Not knowing exactly what to expect makes it impossible to tell whether any unanticipated results derive exclusively from what the researcher believes are the core assumptions in the model, or whether they are due to errors or artefacts. The question is of crucial importance since, unfortunately, the truth is that there are many things that can go wrong in modelling.

Errors and artefacts may appear at various stages of the modelling process (Galán and Izquierdo 2005). In this section we use the extended framework explained in the previous section to identify the critical stages of the modelling process where errors and artefacts are most likely to occur.

According to our definition of artefact – i.e. significant phenomena caused by accessory assumptions that are not considered relevant –, artefacts *cannot* appear in the process of abstraction conducted by the *thematician*, since this stage consists precisely in distilling the *core* features of the target system. Thus, there should not be accessory assumptions in the *thematician*'s model. Nevertheless, there could still be issues with validation if, for instance, the *thematician*'s model is not capturing social reality to a satisfactory extent. Errors could appear in this stage because the *thematician*'s specifications are usually expressed in natural language, and rather than being written down, they are often transmitted orally to the modeller. Thus, an error (i.e. a mismatch between the *thematician*'s specifications and the non-formal model received by the *modeller*) could appear here if the *modeller* misunderstands some of the concepts put forward by the *thematician.*

The *modeller* is the role that may introduce the first artefacts in the modelling process. When formalising the *thematician*'s model, the *modeller* will often have to make a number of additional assumptions so the produced formal model is fully specified. By our definition of the two roles, these additional assumptions are not crucial features of the target system. If such accessory assumptions have a significant impact on the behaviour of the model and the *modeller* is not aware of it, then an artefact has been created. This would occur if, for instance, (a) the *thematician* did not specify any particular neighbourhood function, (b) different neighbourhood functions lead to different results, and (c) the *modeller* is using only one of them and believes that they all produce essentially the same results.

Errors could also appear at this stage, although it is not very likely. This is so because the specifications that the *modeller* produces must be formal, and they are therefore most often written down in a formal language. When this is the case, there is little room for misunderstanding between the *modeller* and the computer scientist, i.e. the *modeller*'s specifications and the formal model received by the *computer scientist* would be the same, and thus there would be no error at this stage.

The role of the *computer scientist* could introduce artefacts in the process. This would be the case if, for instance, his specifications require the use of a particular pseudo-random number generator, he believes that this choice will not have any influence in the results obtained, but it turns out that it does. Similar examples could involve the arbitrary selection of an operating system or a specific floating-point arithmetic that had a significant effect on the output of the model.

Errors can quite easily appear in between the role of the *computer scientist* and the role of the programmer. Note that in our framework any mismatch between the *computer scientist*'s specifications and the executable model received by the *programmer* is considered an error. In particular, if the *computer scientist's* specifications are not executable, then there is an error. This could be, for instance, because the *computer scientist's* specifications stipulate requirements that cannot be executed with present-day computers (e.g. real arithmetic), or because it does not specify all the necessary information to be run in a computer in an unequivocal way (e.g. it does not specify a particular pseudo-random number generator). The error then may affect the validity of the model significantly, or may not.

Note from the previous examples that if the *computer scientist* does not provide a fully executable set of requirement specifications, then he is introducing an error, since in that case the computer program (which is executable) would be necessarily different from his specifications. On the other hand, if he does provide an executable model but in doing so he makes an arbitrary accessory assumption that turns out to be significant, then he is introducing an artefact.

Finally, the *programmer* cannot introduce artefacts because his specifications are the same as the executable model by definition of the role (i.e. the *programmer* does not have to make any accessory assumptions). However, he may make mistakes when creating the computer program from the executable model.

6.4.3 Activities Aimed at Detecting Errors and Artefacts

In this section we identify various activities that the different roles defined in the previous sections can undertake to detect errors and artefacts. We consider the use of these techniques as a very recommendable and eventually easy to apply practice. In spite of this, we should warn that, very often, these activities may require a considerable human and computational effort.

6.4.3.1 ***Modeller*'s Activities**

- Develop and analyse new formal models by implementing alternative accessory assumptions while keeping the core assumptions identified by the *thematician*. This exercise will help to detect artefacts. Only those conclusions which are not falsified by any of these models will be valid for the *thematician*'s model. As an example, see Galán and Izquierdo (2005), who studied different instantiations of one single conceptual model by implementing different evolutionary selection mechanisms. Takadama et al. (2003) conducted a very similar exercise implementing three different learning algorithms for their agents. In a collection of papers, Klemm et al. (2003a, b, c, 2005) investigate the impact of various accessory assumptions in Axelrod's model for the dissemination of culture (Axelrod 1997b). Another example of studying different formal models that address one single problem is provided by Kluver and Stoica (2003).
- Conduct a more exhaustive exploration of the parameter space within the boundaries of the *thematician*'s specifications. If we obtain essentially the same results using the wider parameter range, then we will have broadened the scope of the model, thus making it more representative of the *thematician*'s model. If, on the other hand, results change significantly, then we will have

identified artefacts. This type of exercise has been conducted by e.g. Castellano et al. (2000) and Galán and Izquierdo (2005).

- Create abstractions of the formal model which are mathematically tractable. An example of one possible abstraction would be to study the *expected* motion of a dynamic system (see the studies conducted by Galán and Izquierdo (2005), Edwards et al. (2003), and Castellano et al. (2000) for illustrations of mean-field approximations). Since these mathematical abstractions do not correspond in a one-to-one way with the specifications of the formal model, any results obtained with them will not be conclusive, but they may suggest parts of the model where there may be errors or artefacts.
- Apply the simulation model to relatively well understood and predictable situations to check that the obtained results are in agreement with the expected behaviour (Gilbert et al. 2000).

6.4.3.2 *Computer Scientist*'s Activities

- Develop mathematically tractable models of certain aspects, or particular cases, of the *modeller*'s formal model. The analytical results derived with these models should match those obtained by simulation; a disparity would be an indication of the presence of errors.
- Develop new executable models from the *modeller*'s formal model using alternative modelling paradigms (e.g. procedural vs. declarative). This activity will help to identify artefacts. As an example, see Edmonds and Hales' (2003) reimplementation of Riolo et al. (2001) model of cooperation among agents using tags. Edmonds reimplemented the model using SDML (declarative), whereas Hales reprogrammed the model in Java (procedural).
- Rerun the same code in different computers, using different operating systems, with different pseudo-random number generators. These are most often accessory assumptions of the executable model that are considered non-significant, so any detected difference will be a sign of an artefact. If no significant differences are detected, then we can be confident that the code comprises all the assumptions that could significantly influence the results. This is a valuable finding that can be exploited by the *programmer* (see next activity). As an example, Polhill et al. (2005) explain that using different compilers can result in the application of different floating-point arithmetic systems to the simulation run.

6.4.3.3 *Programmer*'s Activities

- Re-implement the code in different programming languages. Assuming that the code contains all the assumptions that can influence the results significantly, this activity is equivalent to creating alternative representations of the same executable model. Thus, it can help to detect errors in the implementation. There are several examples of this type of activity in the literature. Bigbee et al. (2007) reimplemented Sugarscape (Epstein and Axtell 1996) using MASON. Xu et al. (2003) implemented one single model in Swarm and Repast. The reimplementation exercise conducted by Edmonds and Hales (2003) applies here too.

- Analyse particular cases of the executable model that are mathematically tractable. Any disparity will be an indication of the presence of errors.
- Apply the simulation model to extreme cases that are perfectly understood (Gilbert et al. 2000). Examples of this type of activity would be to run simulations without agents or with very few agents, explore the behaviour of the model using extreme parameter values, or model very simple environments. This activity is common practice in the field.

6.5 Summary

The dynamics of agent-based models are usually so complex that their own developers do not *fully* understand how they are generated. This makes it difficult, if not impossible, to discern whether observed significant results are legitimate logical implications of the assumptions that the model developer is interested in or whether they are due to errors or artefacts in the design or implementation of the model.

Errors are mismatches between what the developer believes a model is and what the model actually is. Artefacts are significant phenomena caused by accessory assumptions in the model that are (mistakenly) considered non-significant. Errors and artefacts prevent developers from correctly understanding their simulations. Furthermore, both errors and artefacts can significantly decrease the validity of a model, so they are best avoided.

In this chapter we have outlined a general framework that summarises the process of designing, implementing, and using agent-based models. Using this framework we have identified the different type of errors and artefacts that may occur in each of the stages of the modelling process. Finally, we have proposed several activities that can be conducted to avoid each type of error or artefact. Some of these activities include repetition of experiments in different platforms, reimplementation of the code in different programming languages, reformulation of the conceptual model using different modelling paradigms, and mathematical analyses of simplified versions or particular cases of the model. Conducting these activities will surely increase our understanding of a particular simulation model.

Acknowledgements The authors have benefited from the financial support of the Spanish Ministry of Education and Science (projects DPI2004–06590, DPI2005–05676 and TIN2008–06464–C03–02), the Spanish Ministry for Science and Innovation (CSD2010–00034) within the framework of CONSOLIDER-INGENIO 2010 and of the JCyL (projects VA006B09, BU034A08 and GREX251–2009). We are also very grateful to Nick Gotts, Gary Polhill, Bruce Edmonds and Cesáreo Hernández for many discussions on the philosophy of modelling.

Further Reading

Gilbert (2007) provides an excellent basic introduction to agent based modelling. Chapter 3 summarizes the different stages involved in an agent-based modelling project, including verification and validation. The paper entitled “Some myths and common errors in simulation experiments” (Schmeiser 2001) discusses briefly some of the most common errors found in simulation from a probabilistic and statistical perspective. The approach is not focused specifically on agent based modelling but on simulation in general. Yilmaz (2006) presents an analysis of the life cycle of a simulation study and proposes a process-centric perspective for the validation and verification of agent-based computational organization models. Finally, Chap. 8 in this volume (David 2013) discusses validation in detail.

References

Axelrod RM (1997a) Advancing the art of simulation in the social sciences. In: Conte R, Hegselmann R, Terna P (eds) Simulating social phenomena, vol 456, Lecture notes in economics and mathematical systems. Springer, Berlin, pp 21–40

Axelrod RM (1997b) The dissemination of culture: a model with local convergence and global polarization. J Confl Resolut 41(2):203–226

Axtell RL (2000) 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, pp 3–24

Axtell RL, Epstein JM (1994) Agent based modeling: understanding our creations. The Bulletin of the Santa Fe Institute, Winter, pp 28–32

Bigbee T, Cioffi-Revilla C, Luke S (2007) Replication of sugarscape using MASON. In: Terano T, Kita H, Deguchi H, Kijima K (eds) Agent-based approaches in economic and social complex systems IV: post-proceedings of the AESCS international workshop 2005. Springer, Tokyo, pp 183–190

Bonabeau E (2002) Agent-based modeling: methods and techniques for simulating human systems. Proc Natl Acad Sci U S A 99(2):7280–7287

Castellano C, Marsili M, Vespignani A (2000) Nonequilibrium phase transition in a model for social influence. Phys Rev Lett 85(16):3536–3539

Christley S, Xiang X, Madey G (2004) Ontology for agent-based modeling and simulation. In: Macal CM, Sallach D, North MJ (eds) Proceedings of the agent 2004 conference on social dynamics: interaction, reflexivity and emergence. Argonne National laboratory/The University of Chicago, Chicago. http://www.agent2005.anl.gov/Agent2004.pdf

Cioffi-Revilla C (2002) Invariance and universality in social agent-based simulations. Proc Natl Acad Sci U S A 99(3):7314–7316

Conlisk J (1996) Why bounded rationality? J Econ Lit 34(2):669–700

David N (2013) Validating simulations. Chapter 8 in this volume

Drogoul A, Vanbergue D, Meurisse T (2003) Multi-agent based simulation: where are the agents? In: Sichman JS, Bousquet F, Davidsson P (eds) Proceedings of MABS 2002 multi-agent-based simulation, vol 2581, Lecture notes in computer science. Springer, Bologna, pp 1–15

Edmonds B (2001) The use of models: making MABS actually work. In: Moss S, Davidsson P (eds) Multi-agent-based simulation, vol 1979, Lecture notes in artificial intelligence. Springer, Berlin, pp 15–32

Edmonds B (2005) Simulation and complexity: how they can relate. In: Feldmann V, Mühlfeld K (eds) Virtual worlds of precision: computer-based simulations in the sciences and social sciences. Lit-Verlag, Münster, pp 5–32

Edmonds B, Hales D (2003) Replication, replication and replication: some hard lessons from model alignment. J Artif Soc Soc Simulat 6(4). http://jasss.soc.surrey.ac.uk/6/4/11.html

Edmonds B, Hales D (2005) Computational simulation as theoretical experiment. J Math Sociol 29:1–24

Edwards M, Huet S, Goreaud F, Deffuant G (2003) Comparing an individual-based model of behaviour diffusion with its mean field aggregate approximation. J Artif Soc Soc Simulat 6(4). http://jasss.soc.surrey.ac.uk/6/4/9.html

Epstein JM (1999) Agent-based computational models and generative social science. Complexity 4(5):41–60

Epstein JM (2008) Why model? J Artif Soc Soc Simul 11(4). http://jasss.soc.surrey.ac.uk/11/4/12.html

Epstein JM, Axtell RL (1996) Growing artificial societies: social science from the bottom up. Brookings Institution Press/MIT Press, Cambridge, MA

Fensel D (2001) Ontologies: a silver bullet for knowledge management and electronic commerce. Springer, Berlin

Galán JM, Izquierdo LR (2005) Appearances can be deceiving: lessons learned re-implementing Axelrod's 'Evolutionary approach to norms'. J Artif Soc Soc Simulat 8(3). http://jasss.soc.surrey.ac.uk/8/3/2.html

Galán JM et al (2009) Errors and artefacts in agent-based modelling. J Artif Soc Soc Simulat 12(1). http://jasss.soc.surrey.ac.uk/12/1/1.html

Gilbert N (1999) Simulation: a new way of doing social science. Am Behav Sci 42(10):1485–1487

Gilbert N (2007) Agent-based models. Sage, London

Gilbert N, Terna P (2000) How to build and use agent-based models in social science. Mind Soc 1 (1):57–72

Gilbert N, Troitzsch KG (1999) Simulation for the social scientist. Open University Press, Buckingham

Gotts NM, Polhill JG, Adam WJ (2003) Simulation and analysis in agent-based modelling of land use change. In: Online proceedings of the first conference of the European Social Simulation Association, Groningen, 18–21 Sept 2003. http://www.uni-koblenz.de/~essa/ESSA2003/gotts_polhill_adam-rev.pdf

Gruber TR (1993) A translation approach to portable ontology specifications. Knowl Acquis 5 (2):199–220

Hare M, Deadman P (2004) Further towards a taxonomy of agent-based simulation models in environmental management. Math Comput Simulat 64(1):25–40

Hernández C (2004) Herbert A Simon, 1916–2001, y el Futuro de la Ciencia Económica. Revista Europea De Dirección y Economía De La Empresa 13(2):7–23

Heywood JG, Masuda K, Rautmann R, Solonnikov VA (eds) (1990) The Navier–Stokes equations: theory and numerical methods. In: Proceedings of a conference held at Oberwolfach, FRG, 18–24, Sept 1988 (Lecture notes in mathematics), vol 1431. Springer, Berlin

Holland JH, Miller JH (1991) Artificial adaptive agents in economic theory. Am Econ Rev 81 (2):365–370

Izquierdo LR, Polhill JG (2006) Is your model susceptible to floating point errors? J Artif Soc Soc Simulat 9(4). http://jasss.soc.surrey.ac.uk/9/4/4.html

Kleijnen JPC (1995) Verification and validation of simulation models. Eur J Oper Res 82 (1):145–162

Kleindorfer GB, O'Neill L, Ganeshan R (1998) Validation in simulation: various positions in the philosophy of science. Manage Sci 44(8):1087–1099

Klemm K, Eguíluz V, Toral R, San Miguel M (2003a) Role of dimensionality in Axelrod's model for the dissemination of culture. Phys A 327:1–5

Klemm K, Eguíluz V, Toral R, San Miguel M (2003b) Global culture: a noise-induced transition in finite systems. Phys Rev E 67(4):045101

Klemm K, Eguíluz V, Toral R, San Miguel M (2003c) Nonequilibrium transitions in complex networks: a model of social interaction. Phys Rev E 67(2):026120

Klemm K, Eguíluz V, Toral R, San Miguel M (2005) Globalization, polarization and cultural drift. J Econ Dyn Control 29(1–2):321–334

Kluver J, Stoica C (2003) Simulations of group dynamics with different models. J Artif Soc Soc Simulat 6(4). http://jasss.soc.surrey.ac.uk/6/4/8.html

Leombruni R, Richiardi M (2005) Why are economists sceptical about agent-based simulations? Phys A 355:103–109

Moss S (2001) Game theory: limitations and an alternative. J Artif Soc Soc Simulat 4(2). http://jasss.soc.surrey.ac.uk/4/2/2.html

Moss S (2002) Agent based modelling for integrated assessment. Integr Assess 3(1):63–77

Moss S, Edmonds B, Wallis S (1997) Validation and verification of computational models with multiple cognitive agents (Report no. 97–25). Centre for Policy Modelling, Manchester. http://cfpm.org/cpmrep25.html

Ostrom T (1988) Computer simulation: the third symbol system. J Exp Soc Psychol 24(5):381–392

Parunak HVD, Savit R, Riolo RL (1998) Agent-based modeling vs. equation-based modeling: a case study and users' guide. In: Sichman JS, Conte R, Gilbert N (eds) Multi-agent systems and agent-based simulation, vol 1534, Lecture notes in artificial intelligence. Springer, Berlin, pp 10–25

Pavón J, Gómez-Sanz J (2003) Agent oriented software engineering with INGENIAS. In: Marik V, Müller J, Pechoucek M (eds) Multi-agent systems and applications III, 3rd international central and eastern European conference on multi-agent systems, CEEMAS 2003 (Lecture notes in artificial intelligence), vol 2691. Springer, Berlin, pp 394–403

Pignotti E, Edwards P, Preece A, Polhill JG, Gotts NM (2005) Semantic support for computational land-use modelling. In: Proceedings of the 5th international symposium on cluster computing and the grid (CCGRID 2005). IEEE Press, Piscataway, pp 840–847

Polhill JG, Gotts NM (2006) A new approach to modelling frameworks. In: Proceedings of the first world congress on social simulation, vol 1. Kyoto, 21–25 Aug 2006, pp 215–222

Polhill JG, Izquierdo LR (2005) Lessons learned from converting the artificial stock market to interval arithmetic. J Artif Soc Soc Simulat 8(2). http://jasss.soc.surrey.ac.uk/8/2/2.html

Polhill JG, Izquierdo LR, Gotts NM (2005) The ghost in the model (and other effects of floating point arithmetic). J Artif Soc Soc Simulat 8(1). http://jasss.soc.surrey.ac.uk/8/1/5.html

Polhill JG, Izquierdo LR, Gotts NM (2006) What every agent based modeller should know about floating point arithmetic. Environ Model Software 21(3):283–309

Riolo RL, Cohen MD, Axelrod RM (2001) Evolution of cooperation without reciprocity. Nature 411:441–443

Sakoda JM (1971) The checkerboard model of social interaction. J Math Sociol 1(1):119–132

Salvi R (2002) The Navier–Stokes equation: theory and numerical methods, Lecture notes in pure and applied mathematics. Marcel Dekker, New York

Sansores C, Pavón J (2005) Agent-based simulation replication: a model driven architecture approach. In: Gelbukh AF, de Albornoz A, Terashima-Marín H (eds) Proceedings of MICAI 2005: advances in artificial intelligence, 4th Mexican international conference on artificial intelligence, Monterrey, 14–18 Nov 2005. Lecture notes in computer science, vol 3789. Springer, Berlin, pp 244–253

Sansores C, Pavón J, Gómez-Sanz J (2006) Visual modeling for complex agent-based simulation systems. In: Sichman JS, Antunes L (eds) Multi-agent-based simulation VI, international workshop, MABS 2005, Utrecht, 25 July 2005, Revised and invited papers. Lecture notes in computer science, vol 3891. Springer, Berlin, pp 174–189

Sargent RG (2003) Verification and validation of simulation models. In: Chick S, Sánchez PJ, Ferrin D, Morrice DJ (eds) Proceedings of the 2003 winter simulation conference. IEEE, Piscataway, pp 37–48

Schelling TC (1971) Dynamic models of segregation. J Math Sociol 1(2):47–186
Schelling TC (1978) Micromotives and macrobehavior. Norton, New York
Schmeiser BW (2001) Some myths and common errors in simulation experiments. In: Peters BA, Smith JS, Medeiros DJ, Rohrer MW (eds) Proceedings of the winter simulation conference, vol 1. Arlington, 9–12 Dec 2001, pp 39–46
Takadama K, Suematsu YL, Sugimoto N, Nawa NE, Shimohara K (2003) Cross-element validation in multiagent-based simulation: switching learning mechanisms in agents. J Artif Soc Soc Simulat 6(4). http://jasss.soc.surrey.ac.uk/6/4/6.html
Taylor AJ (1983) The verification of dynamic simulation models. J Oper Res Soc 34(3):233–242
Xu J, Gao Y, Madey G (2003) A docking experiment: swarm and repast for social network modeling. In: seventh annual swarm researchers conference (SwarmFest 2003), 13–15 Apr 2003, Notre Dame
Yilmaz L (2006) Validation and verification of social processes within agent-based computational organization models. Comput Math Organ Theory 12(4):283–312