José Manuel Galán, Luis R. Izquierdo, Segismundo S. Izquierdo, José Ignacio Santos, Ricardo del Olmo, Adolfo López-Paredes and Bruce Edmonds (2009)

Errors and Artefacts in Agent-Based Modelling

Journal of Artificial Societies and Social Simulation vol. 12, no. 1 1
<http://jasss.soc.surrey.ac.uk/12/1/1.html>

For information about citing this article, click here

Received: 13-Feb-2008 Accepted: 12-Oct-2008 Published: 31-Jan-2009

Abstract

The objectives of this paper are to define and classify different types of errors and artefacts that can appear in the process of developing an agent-based model, and to propose activities aimed at avoiding them during the model construction and testing phases. To do this in a structured way, we review the main concepts of the process of developing such a model - establishing 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 errors and artefacts may appear. Finally we propose activities that could be used to detect (and hence eliminate) each type of error or artefact.

Keywords:
Verification, Replication, Artefact, Error, Agent-Based Modelling, Modelling Roles

Introduction

1.1

Agent-based simulation is one of the techniques that can be used to model social systems. What distinguishes this approach from others is that it facilitates a more direct correspondence between the entities in the target system and the parts of the model that represent them (i.e. the agents) (Edmonds 2001). This more direct approach has the potential to enhance the transparency, soundness, descriptive accuracy and rigour of the modelling process. However, it can also create difficulties: agent-based models are generally mathematically intractable, so that there is little choice but computer simulation for their exploration and analysis.

1.2

The problem with computer simulations is that they can be very complex, so that understanding them in reasonable detail is not a straightforward exercise (this applies as much to understanding one's own simulations as for understanding those of others). A computer simulation can be seen as a process of applying a certain function or rule to the set of inputs to obtain the 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. In other words, in these cases there is no readily accessible "short-cut" to inferring properties of the results. Following this view, understanding a simulation would consist in identifying the parts of this function that are responsible for generating particular (sub)sets of results or properties of results.

1.3

Thus, it becomes apparent that a prerequisite to understanding 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, including the following:

> "*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, p. 38).
>
> "*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 1997b)

1.4

This problem is particularly acute in the case of agent-based simulation. The complex and exploratory nature of most agent-based models implies that, before running a model, there is 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 the model design, its implementation, or its execution.

> "*Indeed, the 'robustness' of macrostructures to perturbations in individual agent performance* [...] *is often a property of agent-based-models and exacerbates the problem of detecting 'bugs'.* " (Axtell and Epstein 1994, p. 31)

1.5

Moreover, the challenge of understanding a computer simulation does not end when one has eliminated any "errors". The difficult task of identifying what parts of the code are generating a particular set of outputs remains. Stated differently, this is the challenge of discovering which assumptions in the model are responsible for the aspects of 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 to the significant results. We explain this in detail in subsequent sections.

1.6

The aim of this paper is to provide a set of concepts and activities that will help in eliminating errors and artefacts in our simulations. Simulations are created by putting together a range of diverse assumptions. All formal models do this, but the more complex nature of simulations, and *especially* social simulations, means that there are often more assumptions and they are intertwined in more complex ways. Some assumptions are made because they are considered to be an essential feature of the system to be modelled; but some are included for other reasons, often in a somewhat arbitrary fashion to achieve completeness -i.e. to make the computer model run (e.g. modelling real space as a grid of square patches). These assumptions may not have a clear referent in the target system but may be necessary to include in order to get a working simulation (but contingent in the sense that there will be a choice in regard to any particular assumption of this kind). These may originate from non-precise intuitions of the modeller, may be dictated by the traditions of a field, or may be imposed by the modelling platform that is being used, for example. There are also many technical 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 its idealised archetype - real arithmetic.

1.7

Thus, in broad terms, if we are to have some confidence in our simulations we need to understand them somewhat - have a good "theory" of why they produce the results they do. Understanding simulations requires identifying what assumptions are being made, and assessing the impact of each one of them 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 framework that summarises the process of creating and using agent-based models in stages; then, within this framework, we characterise the different types of assumptions that are made at 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.

1.8

Agent-based modeling (ABM) is just one particular paradigm in scientific modelling, and many of the considerations in this paper are actually not specific to ABM, but to the general process of scientific modelling. However, given that the motivation of this paper is the analysis of ABM specifically, our discussion is particularly focused on those features and tools of modelling and analysis, such as simulation, that (whilst being common to many other

modelling approaches) are characteristic of ABM. In particular, we will only consider models that are represented (or, more precisely, approximated) by computer programs.

1.9

Admittedly, some of our colleagues have argued that parts of the modelling framework we present here may be controversial and even look arbitrary. Thus, we would like to start by clearly stating the scope of this paper: we do not intended to state "the scientific way" of creating agent-based models or scientific models in general (an issue extensively discussed in Philosophy of Science, which is a controversial discipline by itself (Chalmers 1999)). The aim of this paper is to assist modellers in the difficult task of avoiding errors and artefacts which may appear in the process of developing an agent-based model. To this end, we present a taxonomy and a decomposition of the modelling process into roles and stages (i.e. a modelling framework) that we have found useful and enlightening. The modelling framework we present here is not innovative, but it is based on previous work (Edmonds 2001; Drogoul, Vanbergue and Meurisse 2003); similarly, we do not claim that it is "the right" framework, but just one that has helped us in many cases to achieve our goal of detecting and avoiding errors and artefacts in agent-based models. Our hope is that the reader will find it useful too. As Box and Draper (1987) put it, "all models are wrong, but some are useful".

1.10

The paper is structured as follows. The following section is devoted to explaining what we understand by modelling; 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 section 3 we define the concepts of error and artefact, and we discuss their relevance for validation and verification. The framework presented in section 2 is then used to identify the various stages of the model building 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 during these stages, and we conclude with a brief summary of the paper.

Agent-based modelling

Computational Modelling

2.1

Modelling is the process of building an abstraction of a system for a specific purpose. A model is an abstraction of what is being modelled: maybe retaining only certain features that are considered relevant; maybe making assumptions about unknown aspects; maybe simplifying aspects. Models may be made for a wide variety of purposes, only some of which aim to produce an essentially "correct" representation of the causes behind observed phenomena or to predict outcomes from given conditions. However, if an abstraction does not *in any way* represent its modelling target, it would be inappropriate to call it a model. Thus here we do assume that the behaviour of a model is somehow comparable to what is being modelled. Of course, in many cases things are not directly modelled but rather an abstraction of the target is modelled (Edmonds 2001).

2.2

In science there is a long tradition of analytic modelling, where formal structures using the language of mathematics are used to represent aspects of natural phenomena and to predict them. Analytic modelling can sometimes allow the general derivation of the outcomes that might be observed in the real system being modelled, but generally requires simple circumstances, or strong assumptions to make this possible. Computational modelling is where there is a formal representation that is animated by the computer to produce the outcomes of the model. This may be in the form of a computer program or an algorithm plus a set of equations. The advantages are that one still has a formal object as the model, which can be replicated, checked and used by other researchers (Edmonds 2000) but is freed from the need to be able to derive general results. Rather it can provide a collection of instances of outcomes automatically and reliably. However a big disadvantage is that the model can itself be complex and hard to completely understand.

2.3

There is a wide range of possible compromises that one may have to make when dealing with models that have some analytic and some computational features. The trade-off between various desirable features (e.g. generality, simplicity, predictive power, etc.) depends on the specific case and model. There are not general rules that relate, not even in a qualitative fashion, all these features (Edmonds 2005).

2.4

In any case, it is important to realise that a computer program is a formal model[1] that can

be expressed in mathematical language (Curry-Howard correspondence: Cutland 1980), e.g. as a set of stochastic or deterministic equations, and computer simulation is an inference tool that enables us to study it in ways that go beyond mathematical tractability[2]. Thus, the final result is a potentially more realistic - and still formal - study of a social system.

2.5

Thus, 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.

Concept of Agent-based Modelling

2.6

As stated before, modelling is the process of building an abstraction of a system for a specific purpose (see Edmonds (2001) 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.

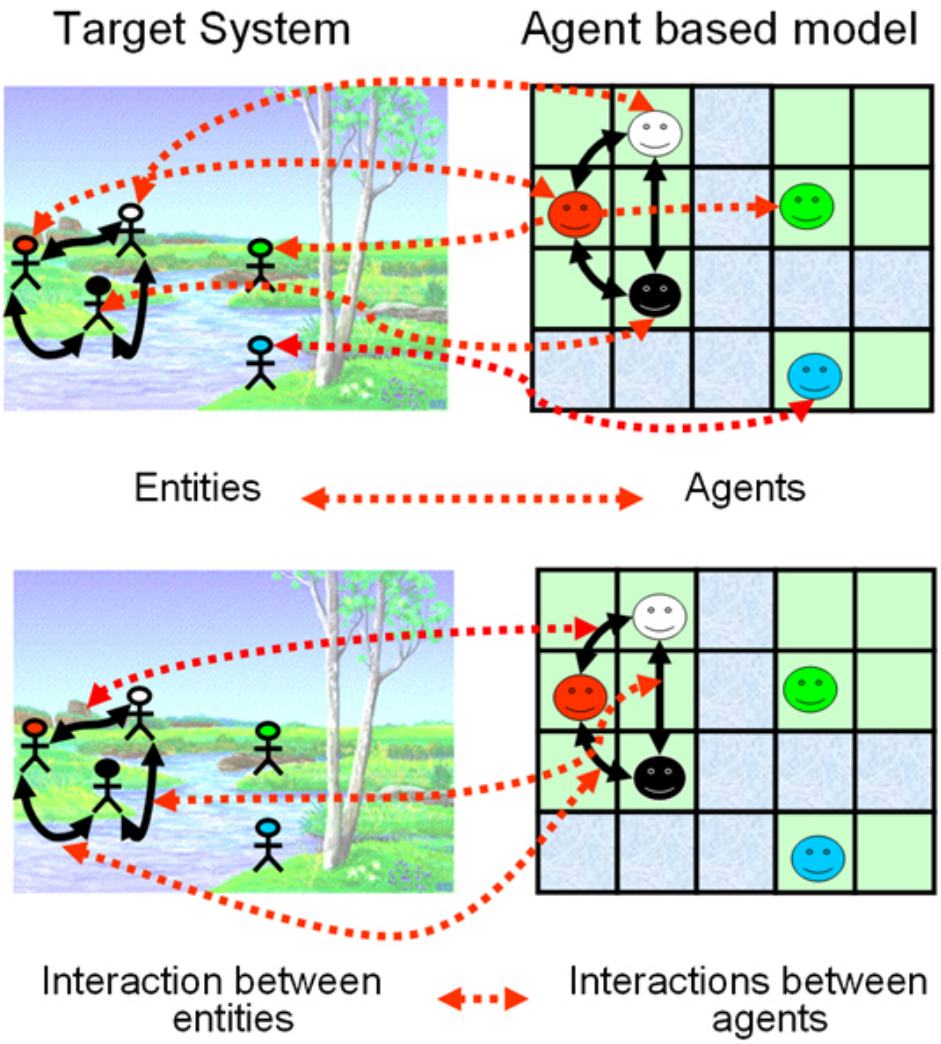

Figure 1. In agent-based modelling the entities of the system are represented explicit and individually in the model. The limits of the entities in the target system correspond to the limits of the agents in the model, and the interactions between entities correspond to the interactions of the agents in the model (Edmonds 2001).

2.7

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 Figure 1). 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[3]. 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.

2.8

The particular way in which the process of abstraction is conducted in agent-based modelling is attractive for various reasons: e.g. 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).

2.9

Thus, to summarise our thoughts in the context 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 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).

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

2.10

This section describes a division of the modelling process into different stages and roles. In practice, different roles may correspond to the same person; the rationale to make this division is that it will help us to detect and classify possible errors and artefacts along the modelling process.

2.11

Drogoul et al. (2003) identify three different roles in the design, implementation, and use of a typical agent-based model: the *thematician*, the *modeller*, and the *computer scientist*. It is not unusual in the field to observe that one single person undertakes several or even all 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 Edmond's views on modelling by dissecting one of Drogoul et al.'s roles into two, and slightly expanding Edmond's framework (Figure 2). In the following section we 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).

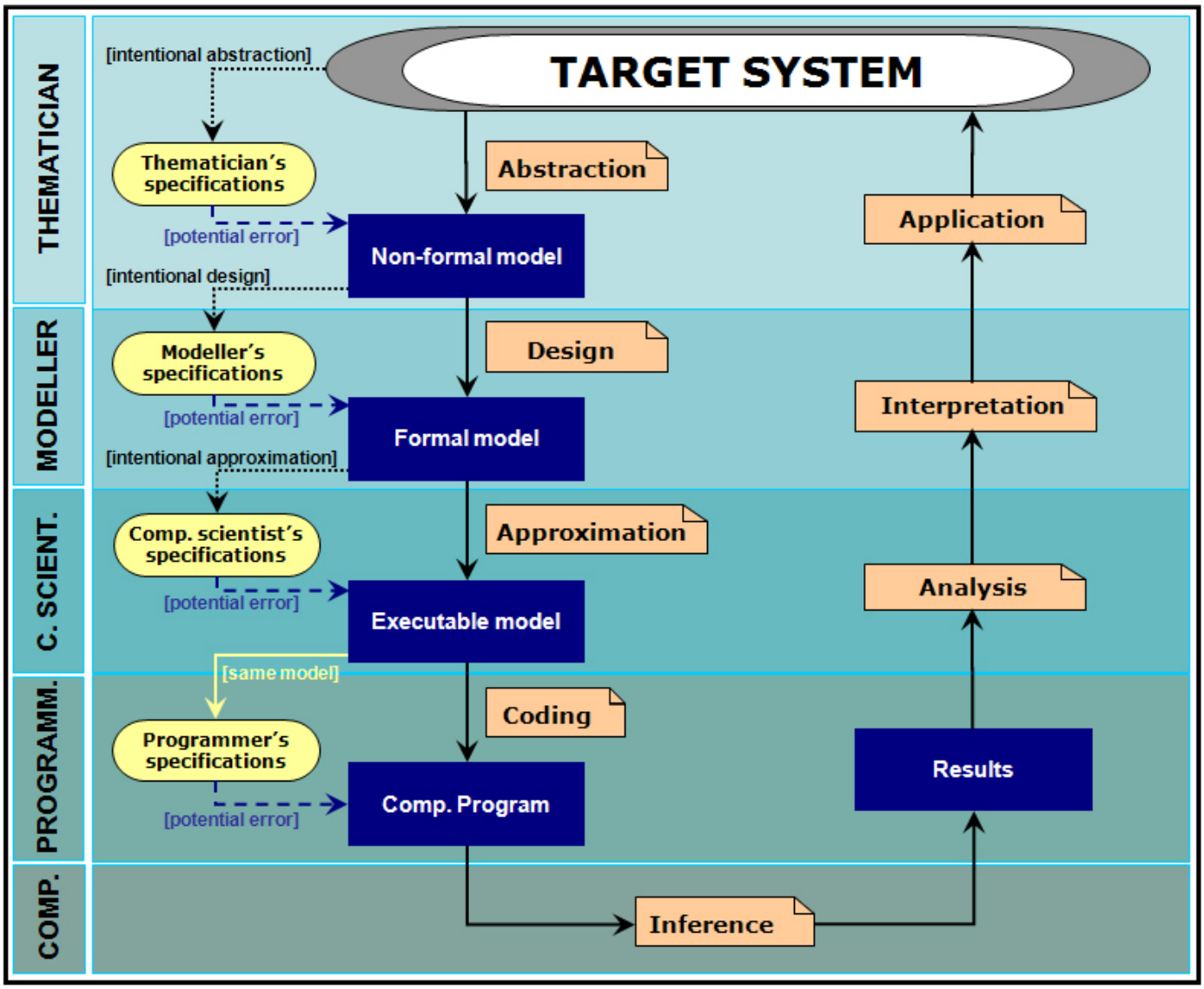

Figure 2. Different stages in the process of designing, implementing and using and agent-based model.

2.12

The role of the *thematician* is meant to produce the first conceptualisation of the target system. This 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).

2.13

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[4]. 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, Polhill and Gotts 2006 and Polhill et al. 2007) have identified ontologies to be particularly promising 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 [...] and conceptualisations in that an ontology is an abstraction of reality." (Polhill and Gotts 2006, p. 51)

2.14

Thus, the modeller has the difficult – potentially unfeasible – task of finding a set of (formal and consistent) requirement specifications[5] 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).

2.15

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.

2.16

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) 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 1990; Salvi 2002). Deriving such approximations would be the task of the *computer scientist*'s role, as defined here.

2.17

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, but *the computer scientist*'s model what 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 which 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. Note also that we consider that the defining feature of a model is the particular input-output relationship it implies. Consequently, two different programs that provide the same input-output relationship would actually be two different representations of the same (executable) model, even though they may be written in different languages and for different operating systems.

2.18

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 Figure 2). (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[6]. If the *programmer* does not make any mistakes, then the implemented model (e.g. the code) and the executable model will be the same.

2.19

Any mismatch between someone's specifications and the actual model he passes to the next

stage is considered here an error (see Figure 2). 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 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.

2.20

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.

2.21

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.

2.22

Once the computer model is implemented, it is run, and the generated results are analysed (see Figure 2). 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.

2.23

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.

Errors and Artefacts

Definition of Error and Artefact, and their Relevance for Validation and Verification

3.1

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)[7]. 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, Edmonds and Wallis 1997).

3.2

Verification (sometimes called "internal validation", e.g. by Taylor (1983), Axelrod (1997b), Drogoul et al. (2003), and Sansores and Pavón (2005), or "program validation", e.g. by Stanislaw (1986) and Richiardi et al (2006)) 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 (i.e. if it is correct). 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.

3.3

We also distinguish between errors and artefacts. *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 (Polhill, Izquierdo and Gotts 2005; Polhill and Izquierdo 2005; Polhill, Izquierdo and Gotts 2006; Izquierdo and Polhill 2006).

3.4

Given our definition of error, it is just impossible to assess whether a certain implementation of a model has errors or not without knowing what the implementer intended to do. A particular implication of this statement is that one cannot look at a piece of code and state that the model has errors (or otherwise) if the programmer's intention is unknown. This is a subtle but important point, so let us use an example to explain it clearly. One may implement a program containing a rule that replaces 2 + 2 with 5. If implementing such a rule was the implementer's intention, then there is no error. This may strike some people, but note that our natural tendency to argue that 2 + 2 should be replaced with 4 (and not 5) comes from the fact that most of us have in our mind an equivalence rule that states "2 + 2 ↔ 4", which conflicts with the rule implemented in the program. But the program has no background rules apart from those that are explicitly stated within it and constitute its precise definition. Thus, from the computer's point of view, the rule "2 + 2 → 5" has no further meaning than "Replace the string of symbols '2' '+' '2' with the symbol '5'", and is just as valid as "Replace the string of symbols '2' '+' '2' with the symbol '4'".

3.5

Another illuminating example is a program that ends with a message from the computer saying something like e.g. "Error: null pointer assignment". Is this an error in the implementation of the model? Not necessarily; the programmer may well be an instructor trying to show his students certain conditions under which the program is trying to access an illegal memory address; assuming so, the instructor would undoubtedly insist that he has made no error in programming, since his intention was precisely to obtain such a message from the computer, and the program is performing in accordance to his specifications. In other words, since there is no mismatch between what the developer thinks the model is, and what it actually is, then there is no error.

3.6

A final example: why do people say that there has been a floating-point error when observing (in any IEEE-754 compliant platform) that the operation "0.6 - 0.4 - 0.2" yields a number strictly below 0? Only because people (explicit or implicitly) generally assume that our intention is to use real arithmetic. But the program has no more assumptions than those embedded in it, and it may well be the case (and usually is) that the computer does not contain the rules of real arithmetic within it. Thus, the statement "0.6 - 0.4 - 0.2 < 0" is an error if real arithmetic is intended but not if IEEE-754 floating-point arithmetic is intended. A programmer in charge of developing an IEEE-754 compliant platform would see no error in the statement "0.6 - 0.4 - 0.2 < 0"; rather the opposite, he would readily admit having made an error if he observed "0.6 - 0.4 - 0.2 == 0". Thus it is clear that whether an implementation of a program has errors or not crucially depends on the programmer's intentions.

3.7

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* 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[8]. 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. 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. Note that an artefact is no longer an artefact as soon as it is discovered, in the same sense that a lost item is no longer lost as soon as it is found.

3.8

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 Schelling-Sakoda model of segregation, which was designed to investigate the causal link between individual preferences and global patterns of segregation (Schelling 1971; Sakoda 1971; Schelling 1978).

3.9

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.

Appearance of Errors and Artefacts

3.10

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.

3.11

Errors and artefacts may appear at various stages of the modelling process (Galan and Izquierdo 2005). In this section we use the extended framework explained in section 2.3 to identify the critical stages of the modelling process where errors and artefacts are most likely to occur.

3.12

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*.

3.13

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 all of them would produce essentially the same results.

3.14

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.

3.15

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.

3.16

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.

3.17

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.

3.18

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.

Activities Aimed at Detecting Errors and Artefacts

3.19

In this section we identify various activities that the different roles defined in the previous sections can undertake to detect errors and artefacts. Note that, while all the techniques that we indicate have been successfully used in different cases, not all of them may be applicable to every model.

3.20

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 Galan and Izquierdo (2005), who studied different instantiations of one single conceptual model by implementing different evolutionary selection mechanisms. Takadama et al. (2003) conducted a similar exercise implementing three different learning algorithms for their agents. In a collection of papers, Klemm, Eguiluz, Toral and Miguel (2003a; 2003b; 2003c; 2005) investigate the impact of various accessory assumptions in

Axelrod's model for the dissemination of culture (Axelrod 1997a). Flache and Hegselman (2001) conducted a thorough robustness test on two of their cellular automata models by changing their (originally regular) grid structure. Izquierdo et al. analysed the impact of using different structures of social networks (Izquierdo and Izquierdo 2007), of introducing noise (Izquierdo, Izquierdo and Gotts 2008) and of changing various structural assumptions (Izquierdo and Izquierdo 2006) on the results obtained with several models. Another example of studying different formal models that address one single problem is provided by Kluver and Stoica (2003). In those cases where the order of agents' decisions in the model is considered an accessory assumption, it is convenient to experiment with different scheduling alternatives (Caron-Lormier, Humphry, Bohan, Hawes and Thorbek 2008; Miller and Page 2004; Schönfisch and De Roos 1999).

- 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 a 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, Marsili, and Vespignani (2000) and Galan and Izquierdo (2005).
- 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 and Terna 2000).
- 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 Galan and Izquierdo (2005), Edwards et al. (2003), Castellano, Marsili, and Vespignani (2000), Huet et al. (2007), Mabrouk et al. (2007), Vilà (2008) and Izquierdo et al. (2007; 2008) 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 in general, but they may suggest parts of the model where there may be errors or artefacts.
- In some cases it is not necessary to make abstractions of the formal model because the model itself -or more often some parts of it- can be mathematically formalised and analysed. In these situations it may be possible to formally check if the requirements imposed by the *thematician* are fulfilled or not. If it is discovered that such requirements are not observed, then an error has been uncovered. This type of exercise is most often carried out by scientists who were not involved in the original modelling process. An example of this activity can be found in Ehrentreich's analytical work (Ehrentreich 2002; Ehrentreich 2006) on the Artificial Stock Market (Arthur, Holland, LeBaron, Palmer and Tayler 1997; LeBaron, Arthur and Palmer 1999) where, using Markov chain analyses, he demonstrates that the mutation operator used in the design of the model is not neutral to the learning rate, but it introduces an upward bias in the model[9]. A more positive example is provided by Izquierdo et al. (2007; 2008), who used mathematical analysis to confirm and advance various insights on reinforcement learning put forward by Macy and Flache (2002) and Flache and Macy (2002) using computer simulation. Similarly, Galan and Izquierdo (2005) analysed Axelrod's (1986) agent-based model as a Markov chain, which allowed the authors to conclude that the *long-run* behaviour of that model was independent of the initial conditions, in contrast to the initial conclusions of the original analysis. Izquierdo et al. (2009) illustrate the usefulness of the theory of Markov chains by analysing 10 well-known models in the Social Simulation literature as Markov chains.

3.21

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, Cohen and Axelrod's (2001) model of cooperation among agents using tags. Edmonds reimplemented the model using SDML (a declarative language), whereas Hales reprogrammed the model in Java (a procedural language).
- Rerun the same code in different computers, using different operating systems, with different pseudo-random number generators, etc. 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 more 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.

3.22

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, Cioffi-Revilla and Luke (2007) reimplemented Sugarscape (Epstein and Axtell 1996) using MASON. Xu, Gao and Madey (2003) implemented one single model in Swarm and Repast. Wilenski and Rand (2007) re-implemented in NetLogo an agent-based model on the evolution of ethnocentrism proposed by Hammond and Axelrod (2006a; 2006b) and initially implemented in Ascape. 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 and Terna 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, model very simple environments, etc. This activity is common practice in the field.

Validation Stages and Checks

3.23

This paper has covered the activities that are relevant to the model construction and testing phases - the downward pointing arrows of Figure 2. We have not attempted to identify the errors that can occur in the analysis, interpretation or application phases. For example one might have re-implemented a model completely correctly but interpreted the results simply wrongly. It may be that the results obtained were a rare freak and would only have occurred in 1 in a 1000 runs of a model, but unless the model was run many times we would never know this.

3.24

Nor has this paper attempted to discuss how activities during these "upward-pointing-arrow stages" in Figure 2 might be used to check a model. For example it does happen that discrepancies between the model outcomes and the outcomes the model is to be judged against lead and guide a modeller to look for mistakes in the model programming (although this is rarely documented, so that the reader often does not know the extent to which the outcomes independently *validate* the model and how much the model has been already conditioned against those outcomes). Clearly this option is more relevant for those models that are more closely related to observations of the target of modelling than those that are really the animation of an abstract mental theory.

3.25

We are not aware of a single summary of the methods and pitfalls in these stages, but the reader can find some discussions in (Axelrod 1997b, Edmonds 2001, Gilbert 2007, Gilbert and Terna 2000, Gilbert and Troitzsch 1999, Kleijnen 1995, Moss and Edmonds 2005).

Summary

4.1

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 they are due to errors or artefacts in the design or implementation of the model.

4.2

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.

4.3

In this paper 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 types of errors and artefacts that may occur at each of the construction stages of the modelling process. Finally, we have proposed several activities that can be conducted to help avoid each type of error or artefact during these stages. These 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 any particular simulation model and, ultimately, help improve their reliability.

Notes

[1] A formal model is a model expressed in a formal system (Cutland 1980), which is a system used to derive one expression from one or more other previous expressions in the same system. Basically, a formal system consists of a formal language (a set of symbols and rules to combine them) together with a deductive system (a set of inference rules and/or axioms).

[2] By "mathematically intractable" we mean that applying deductive inference to the mathematically formalised model, given the current state of development of mathematics, does not provide a solution or clear insight into the behaviour of the model, so there is a need to resort to techniques such as simulation or numerical approximations in order to study the input-output relationship that characterises the model.

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

[4] 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.

[5] 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.

[6] 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 , Pavón and Gómez-Sanz 2006). These efforts are aimed at automatically generating several implementations of one single executable model (in various different simulation platforms).

[7] See a complete epistemic review of the validation problem in Kleindorfer et al. (1998) and discussion about the specific domain of agent-based modelling in Windrum et al (2007) and Moss (2008).

[8] If we accept, as Edmonds and Hales (2005) propose, that a computational simulation is a theoretical experiment, then our definition of the concept of accessory assumption could be assimilated by analogy to a particular case of auxiliary assumption as defined in the context of the Duhem-Quine thesis (Windrum, Fagiolo and Moneta 2007). Notwithstanding, in order to be considered an artefact, an assumption not only needs the condition of auxiliary hypothesis but also the condition of significant assumption.

[9] This finding does not refute some of the most important conclusions of the 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) and of the Junta de Castilla y León (project BU034A08). We are also very grateful to Nick Gotts, Gary Polhill and Cesáreo Hernández for many discussions on the Philosophy of modelling.

References

ARTHUR W B, Holland J H, LeBaron B, Palmer R and Tayler P (1997) Asset Pricing under Endogenous Expectations in an Artificial Stock Market. In Arthur W B, Durlauf S, and Lane D

(Eds.) *The Economy as an Evolving Complex System II*: 15-44. Reading, MA: Addison-Wesley Longman.

AXELROD R M (1986) An Evolutionary Approach to Norms. *American Political Science Review*, 80(4), pp. 1095-1111.

AXELROD R M (1997a) The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, 41(2), pp. 203-226.

AXELROD R M (1997b) Advancing the Art of Simulation in the Social Sciences. In Conte R, Hegselmann R, and Terna P (Eds.) *Simulating Social Phenomena, Lecture Notes in Economics and Mathematical Systems 456*: 21-40. Berlin: Springer-Verlag.

AXTELL R L (2000) Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences. In Macal C M and Sallach D (Eds.) *Proceedings of the Workshop on Agent Simulation: Applications, Models, and Tools*: 3-24. Argonne, IL: Argonne National Laboratory.

AXTELL R L and Epstein J M (1994) Agent-based Modeling: Understanding Our Creations. *The Bulletin of the Santa Fe Institute*, Winter 1994, pp. 28-32.

BIGBEE T, Cioffi-Revilla C and Luke S (2007) Replication of Sugarscape using MASON. In Terano T, Kita H, Deguchi H, and Kijima K (Eds.) *Agent-Based Approaches in Economic and Social Complex Systems IV*: 183-190. Springer Japan.

BONABEAU E (2002) Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(2), pp. 7280-7287.

BOX G E P and Draper N R (1987) *Empirical model-building and response surfaces*. New York: Wiley.

CARON-LORMIER G, Humphry R W, Bohan D A, Hawes C and Thorbek P (2008) Asynchronous and synchronous updating in individual-based models. *Ecological Modelling*, 212(3-4), pp. 522-527.

CASTELLANO C, Marsili M and Vespignani A (2000) Nonequilibrium phase transition in a model for social influence. *Physical Review Letters*, 85(16), pp. 3536-3539.

CHALMERS A F (1999) *What is this thing called science?* Indianapolis: Hackett Pub.

CHRISTLEY S, Xiang X and Madey G (2004) Ontology for agent-based modeling and simulation. In Macal C M, Sallach D, and North M J (Eds.) *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*: Chicago, IL: Argonne National Laboratory and The University of Chicago. http://www.agent2005.anl.gov/Agent2004.pdf.

CIOFFI-REVILLA C (2002) Invariance and universality in social agent-based simulations. *Proceedings of the National Academy of Sciences of the United States of America*, 99(3), pp. 7314-7316.

CUTLAND, N (1980) *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press.

DROGOUL A, Vanbergue D and Meurisse T (2003) Multi-Agent Based Simulation: Where are the Agents? In Sichman J S, Bousquet F, and Davidsson P (Eds.) *Proceedings of MABS 2002 Multi-Agent-Based Simulation, Lecture Notes in Computer Science 2581*: 1-15. Bologna, Italy: Springer-Verlag.

EDMONDS B (2000) The Purpose and Place of Formal Systems in the Development of Science, CPM Report 00-75, MMU, UK. http://cfpm.org/cpmrep75.html.

EDMONDS B (2001) The Use of Models - making MABS actually work. In Moss S and Davidsson P (Eds.) *Multi-Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1979*: 15-32. Berlin: Springer-Verlag.

EDMONDS B (2005) Simulation and Complexity - how they can relate. In Feldmann V and Mühlfeld K (Eds.) *Virtual Worlds of Precision - computer-based simulations in the sciences and social sciences*: 5-32. Münster, Germany: Lit-Verlag.

EDMONDS B and Hales D (2003) Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation*, 6(4) http://jasss.soc.surrey.ac.uk/6/4/11.html.

EDMONDS B and Hales D (2005) Computational Simulation as Theoretical Experiment. *Journal of Mathematical Sociology*, 29, pp. 1-24.

EDWARDS M, Huet S, Goreaud F and Deffuant G (2003) Comparing an individual-based model of behaviour diffusion with its mean field aggregate approximation. *Journal of Artificial Societies and Social Simulation*, 6(4) http://jasss.soc.surrey.ac.uk/6/4/9.html.

EHRENTREICH N (2002) The Santa Fe Artificial Stock Market Re-Examined - Suggested Corrections. *Economics Working Paper Archive at WUSTL*. http://econwpa.wustl.edu:80/eps/comp/papers/0209/0209001.pdf.

EHRENTREICH N (2006) Technical trading in the Santa Fe Institute Artificial Stock Market revisited. *Journal of Economic Behavior & Organization*, 61(4), pp. 599-616.

EPSTEIN J M (1999) Agent-based computational models and generative social science. *Complexity*, 4(5), pp. 41-60.

EPSTEIN J M and Axtell R L (1996) *Growing Artificial Societies. Social Science From the Bottom Up*. Cambridge, MA: Brookings Institution Press-MIT Press.

FENSEL D (2001) *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Berlin: Springer.

FLACHE A and Hegselmann R (2001) Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics. *Journal of Artificial Societies and Social Simulation*, 4(4) http://jasss.soc.surrey.ac.uk/4/4/6.html.

FLACHE A and Macy M W (2002) Stochastic Collusion and the Power Law of Learning. *Journal of Conflict Resolution*, 46(5), pp. 629-653

GALAN J M and Izquierdo L R (2005) Appearances Can Be Deceiving: Lessons Learned Re-Implementing Axelrod's 'Evolutionary Approach to Norms'. *Journal of Artificial Societies and Social Simulation*, 8(3) 2 http://jasss.soc.surrey.ac.uk/8/3/2.html.

GILBERT N (2007) *Agent-Based Models. Quantitative Applications in the Social Sciences*. London: SAGE Publications.

GILBERT N and Terna P (2000) How to build and use agent-based models in social science. *Mind and Society*, 1(1), pp. 57-72.

GILBERT N and Troitzsch K G (1999) *Simulation for the social scientist*. Buckingham, UK: Open University Press.

HAMMOND R A and Axelrod R M (2006a) Evolution of contingent altruism when cooperation is expensive. *Theoretical Population Biology*, 69(3), pp. 333-338.

HAMMOND R A and Axelrod R M (2006b) The Evolution of Ethnocentrism. *Journal of Conflict Resolution*, 50(6), pp. 926-936.

GOTTS N M, Polhill J G and Adam W J (2003) Simulation and Analysis in Agent-Based Modelling of Land Use Change. *Online proceedings of the First Conference of the European Social Simulation Association, Groningen, The Netherlands, 18-21 September 2003*. http://www.uni-koblenz.de/~kgt/ESSA/ESSA1/proceedings.htm.

GRUBER T R (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), pp. 199-220.

HARE M and Deadman P (2004) Further towards a taxonomy of agent-based simulation models in environmental management. *Mathematics and Computers in Simulation*, 64(1), pp. 25-40.

HEYWOOD J G (1990) *The Navier-Stokes equations. Theory and numerical methods. Proceedings of a conference held at Oberwolfach, FRG, Sept. 18-24, 1988. Lecture Notes in Mathematics*. Berlin: Springer-Verlag.

HUET S, Edwards M and Deffuant G (2007) Taking into Account the Variations of Neighbourhood Sizes in the Mean-Field Approximation of the Threshold Model on a Random Network. *Journal of Artificial Societies and Social Simulation*, 10(1) 10 http://jasss.soc.surrey.ac.uk/10/1/10.html.

IZQUIERDO L R, Izquierdo, S S, Galán, J M and Santos, J I (2009) Techniques to Understand Computer Simulations: Markov Chain Analysis. *Journal of Artificial Societies and Social Simulation*, 12(1) 6 http://jasss.soc.surrey.ac.uk/12/1/6.html.

IZQUIERDO S S, Izquierdo, L R and Gotts N M (2008) Reinforcement learning dynamics in social dilemmas. *Journal of Artificial Societies and Social Simulation*, 11(2) 1 http://jasss.soc.surrey.ac.uk/11/2/1.html.

IZQUIERDO L R, Izquierdo S S, Gotts N M and Polhill J G (2007) Transient and Asymptotic Dynamics of Reinforcement Learning in Games. *Games and Economic Behavior*, 61(2), pp. 259-276. http://dx.doi.org/10.1016/j.geb.2007.01.005.

IZQUIERDO L R and Polhill J G (2006) Is your model susceptible to floating point errors? *Journal of Artificial Societies and Social Simulation*, 9(4) 4 http://jasss.soc.surrey.ac.uk/9/4/4.html.

IZQUIERDO S S and Izquierdo L R (2006) On the Structural Robustness of Evolutionary Models of Cooperation. In Corchado E, Yin H, Botti V J, and Fyfe C (Eds.) *Intelligent Data Engineering and Automated Learning - IDEAL 2006. Lecture Notes in Computer Science 4224*: 172-182. Berlin Heidelberg: Springer.

IZQUIERDO S S and Izquierdo L R (2007) The Impact on Market Efficiency of Quality Uncertainty without Asymmetric Information. *Journal of Business Research*, 60(8), pp. 858-867. http://dx.doi.org/10.1016/j.jbusres.2007.02.010.

KLEIJNEN J P C (1995) Verification and validation of simulation models. *European Journal of Operational Research*, 82(1), pp. 145-162.

KLEINDORFER G B, O'Neill L and Ganeshan R (1998) Validation in simulation: Various positions in the philosophy of science. *Management Science*, 44(8), pp. 1087-1099.

KLEMM K, Eguíluz V M, Toral R and San Miguel M (2003a) Role of dimensionality in Axelrod's model for the dissemination of culture. *Physica A*, 327, pp. 1-5.

KLEMM K, Eguiluz V M, Toral R and Miguel M S (2003b) Global culture: A noise-induced transition in finite systems. *Physical Review E*, 67(4).

KLEMM K, Eguiluz V M, Toral R and San Miguel M (2003c) Nonequilibrium transitions in complex networks: A model of social interaction. *Physical Review E*, 67(2).

KLEMM K, Eguiluz V M, Toral R and San Miguel M (2005) Globalization, polarization and cultural drift. *Journal of Economic Dynamics & Control*, 29(1-2), pp. 321-334.

KLUVER J and Stoica C (2003) Simulations of group dynamics with different models. *Journal of Artificial Societies and Social Simulation*, 6(4) http://jasss.soc.surrey.ac.uk/6/4/8.html.

LEBARON B, Arthur W B and Palmer R (1999) Time series properties of an artificial stock market. *Journal of Economic Dynamics & Control*, 23(9-10), pp. 1487-1516.

LEOMBRUNI R and Richiardi M (2005) Why are economists sceptical about agent-based simulations? *Physica A*, 355, pp. 103-109.

MABROUK N, Deffuant G and Lobry C (2007) Confronting macro, meso and micro scale modelling of bacteria dynamics. *M2M 2007: Third International Model-to-Model Workshop, Marseille, France, 15-16 March 2007*. http://m2m2007.macaulay.ac.uk/M2M2007-Mabrouk.pdf.

MACY M W and Flache A (2002) Learning Dynamics in Social Dilemmas. *Proceedings of the National Academy of Sciences of the United States of America*, 99(3), pp. 7229-7236.

MILLER J H and Page S E (2004) The standing ovation problem. *Complexity*, 9(5), pp. 8-16.

MOSS S (2008) Alternative Approaches to the Empirical Validation of Agent-Based Models. *Journal of Artificial Societies and Social Simulation*, 11(1) 5 http://jasss.soc.surrey.ac.uk/11/1/5.html.

MOSS S and Edmonds B (2005) Sociology and Simulation: - Statistical and Qualitative Cross-Validation, *American Journal of Sociology*, 110(4), pp. 1095-1131.

MOSS S, Edmonds B and Wallis S (1997) Validation and Verification of Computational Models with Multiple Cognitive Agents. *Centre for Policy Modelling Report*, No.: 97-25 http://cfpm.org/cpmrep25.html.

PARUNAK H V D, Savit R and Riolo R L (1998) Agent-based modeling vs. equation-based modeling: A case study and users' guide. In Sichman J S, Conte R, and Gilbert N (Eds.) *Multi-Agent Systems and Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1534*: 10-25. Berlin, Germany: Springer-Verlag.

PAVÓN J and Gómez-Sanz J (2003) Agent Oriented Software Engineering with INGENIAS. In Marik V, Müller J, and 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 2691*: 394-403. Berlin Heidelberg: Springer-Verlag.

PIGNOTTI E, Edwards P, Preece A, Polhill J G and Gotts N M (2005) Semantic support for computational land-use modelling. *5th International Symposium on Cluster Computing and the Grid (CCGRID 2005)*: 840-847. Piscataway, NJ: IEEE Press.

POLHILL J G and Gotts N M (2006) A new approach to modelling frameworks. *Proceedings of the First World Congress on Social Simulation*: 50-57. Kyoto.

POLHILL J G and Izquierdo L R (2005) Lessons learned from converting the artificial stock market to interval arithmetic. *Journal of Artificial Societies and Social Simulation*, 8(2) http://jasss.soc.surrey.ac.uk/8/2/2.html.

POLHILL J G, Izquierdo L R and Gotts N M (2006) What every agent-based modeller should know about floating point arithmetic. *Environmental Modelling & Software*, 21(3), pp. 283-309.

POLHILL J G, Izquierdo L R and Gotts N M (2005) The ghost in the model (and other effects of floating point arithmetic). *Journal of Artificial Societies and Social Simulation*, 8(1) http://jasss.soc.surrey.ac.uk/8/1/5.html.

POLHILL J G, Pignotti E, Gotts N M, Edwards P and Preece A (2007) A Semantic Grid Service for Experimentation with an Agent-Based Model of Land-Use Change. *Journal of Artificial Societies and Social Simulation*, 10(2) 2 http://jasss.soc.surrey.ac.uk/10/2/2.html.

RICHIARDI M, Leombruni R, Saam N J and Sonnessa M (2006) A Common Protocol for Agent-Based Social Simulation. *Journal of Artificial Societies and Social Simulation*, 9(1) 15 http://jasss.soc.surrey.ac.uk/9/1/15.html.

RIOLO R L, Cohen M D and Axelrod R M (2001) Evolution of cooperation without reciprocity. *Nature*, 411, pp. 441-443.

SAKODA J M (1971) The Checkerboard Model of Social Interaction. *Journal of Mathematical Sociology*, 1(1), pp. 119-132.

SALVI R (2002) *The Navier-Stokes equations. Theory and numerical methods. Lecture Notes in Pure and Applied Mathematics*. New York, NY: Marcel Dekker.

SANSORES C and Pavón J (2005) Agent-based simulation replication: A model driven architecture approach. In Gelbukh A F, de Albornoz A, and Terashima-Marín H (Eds.) *MICAI 2005: Advances in Artificial Intelligence, 4th Mexican International Conference on Artificial Intelligence, Monterrey, Mexico, November 14-18, 2005, Proceedings. Lecture Notes in Computer Science 3789*: 244-253. Berlin Heidelberg: Springer.

SANSORES C, Pavón J and Gómez-Sanz J (2006) Visual modeling for complex agent-based simulation systems. In Sichman J S and Antunes L (Eds.) *Multi-Agent-Based Simulation VI, International Workshop, MABS 2005, Utrecht, The Netherlands, July 25, 2005, Revised and Invited Papers. Lecture Notes in Computer Science 3891*: 174-189. Berlin Heidelberg: Springer.

SARGENT R G (2003) Verification and Validation of Simulation Models. In Chick S, Sánchez P J, Ferrin D, and Morrice D J (Eds.) *Proceedings of the 2003 Winter Simulation Conference*: 37-48. Piscataway, NJ: IEEE.

SCHELLING T C (1978) *Micromotives and macrobehavior*. New York: Norton.

SCHELLING T C (1971) Dynamic Models of Segregation. *Journal of Mathematical Sociology*, 1(1), pp. 147-186.

SCHÖNFISCH B and De Roos A (1999) Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51(3), pp. 123-143.

STANISLAW H (1986) Tests of computer simulation validity. What do they measure? *Simulation and Games* 17, pp. 173-191.

TAKADAMA K, Suematsu Y L, Sugimoto N, Nawa N E and Shimohara K (2003) Cross-element validation in multiagent-based simulation: Switching learning mechanisms in agents. *Journal of Artificial Societies and Social Simulation*, 6(4) http://jasss.soc.surrey.ac.uk/6/4/6.html.

TAYLOR A J (1983) The Verification of Dynamic Simulation Models. *Journal of the Operational Research Society*, 34(3), pp. 233-242.

VILÀ X (2008) A Model-To-Model Analysis of Bertrand Competition. *Journal of Artificial Societies and Social Simulation*, 11(2) 11 http://jasss.soc.surrey.ac.uk/11/2/11.html.

WILENSKY U and Rand W (2007) Making Models Match: Replicating an Agent-Based Model. *Journal of Artificial Societies and Social Simulation*, 10(4) 2 http://jasss.soc.surrey.ac.uk/10/4/2.html.

WINDRUM P, Fagiolo G and Moneta A (2007) Empirical Validation of Agent-Based Models: Alternatives and Prospects. *Journal of Artificial Societies and Social Simulation*, 10(2) http://jasss.soc.surrey.ac.uk/10/2/8.html.

XU J, Gao Y and Madey G (2003) A Docking Experiment: Swarm and Repast for Social Network Modeling. *Seventh Annual Swarm Researchers Meeting (Swarm2003)*: Notre Dame, IN.

Return to Contents of this issue