What is a Simulation Model?

Durán, Juan M.

DOI
10.1007/s11023-020-09520-z

Publication date
2020

Document Version
Final published version

Published in
Minds and Machines

Citation (APA)
Durán, J. M. (2020). What is a Simulation Model? *Minds and Machines*, *30*(3), 301-323. https://doi.org/10.1007/s11023-020-09520-z

Minds and Machines
https://doi.org/10.1007/s11023-020-09520-z

What is a Simulation Model?

Juan M. Durán[1]

Received: 2 November 2019 / Accepted: 28 February 2020

Abstract
Many philosophical accounts of scientific models fail to distinguish between a simulation model and other forms of models. This failure is unfortunate because there are important differences pertaining to their methodology and epistemology that favor their philosophical understanding. The core claim presented here is that simulation models are rich and complex units of analysis in their own right, that they depart from known forms of scientific models in significant ways, and that a proper understanding of the type of model simulations are fundamental for their philosophical assessment. I argue that simulation models can be distinguished from other forms of models by the many algorithmic structures, representation relations, and new semantic connections involved in their architecture. In this article, I reconstruct a general architecture for a simulation model, one that faithfully captures the complexities involved in most scientific research with computer simulations. Furthermore, I submit that a new methodology capable of conforming such architecture into a fully functional, computationally tractable computer simulation must be in place. I discuss this methodology—what I call *recasting*—and argue for its philosophical novelty. If these efforts are heading towards the right interpretation of simulation models, then one can show that computer simulations shed new light on the philosophy of science. To illustrate the potential of my interpretation of simulation models, I briefly discuss simulation-based explanations as a novel approach to questions about scientific explanation.

Keywords Computer simulations · Simulation models · Computer-based explanation · Recasting · Representation · Novelty of computer simulations

✉ Juan M. Durán
j.m.duran@tudelft.nl

[1] Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands

Published online: 07 March 2020

1 Introduction

In recent years, philosophers have turned their attention to computer simulations and their epistemological value in scientific modeling and scientific practice. When faced with these issues, authors employ one of several strategies: they compare the epistemological power of computer simulations to laboratory experimentation (Morgan 2003; Parker 2009; Symons and Alvarado 2019; Ionescu 2018; Boge 2019); they analyze different forms of inferring knowledge from simulations (Winsberg 2001; Beisbart 2012); they hold extensive discussions on the notion of simulated data as different in kind from experimental and observational data (Barberousse and Marion 2013; Humphreys 2013); or they show how computer simulations are at the center of methodological, conceptual, and industrial changes in scientific research (Lenhard 2014).

There are many insightful discussions in the philosophical literature about the epistemology of computer simulations. Most of this literature, however, takes computer simulations to consist of the implementation of some kind of special model running on the physical computer. This leaves a conceptual vacuum regarding the specific nature of such special models and the philosophical implications in connection with them. Although serious efforts can be found in the literature, most prominently Humphreys (1990, 2004), Hartmann (1996), Winsberg (1999), and more recently Durán (2018), there seems to be a discrepancy between the way philosophers conceptualize these models and the complexities attached to the practice of simulating.

The primary aim of this article is to articulate the richness and complexity of *simulation models*, that is, the type of models at the basis of computer simulations. To this end, I examine the general architecture for simulation models with the objective of recognizing practices, structures, and relations commonly found in computational practice and which differ from other forms of modeling (see Symons and Alvarado 2019; Durán 2018). In particular, I argue that simulation models require new, arguably unprecedented techniques for their design and construction. I call these techniques *recasting*, and I understand them as ways to make compatible a multiplicity of models into one single simulation model for its implementation as a fully functional computer simulation. To show their novelty, I contrast recasting against forms of sub-modeling and multi-modeling.

The article is also a contribution to efforts showing the philosophical novelty of computer simulations. To many, the fact that computer simulations are a *scientific novelty* also entails their *philosophical novelty*. However, this claim has found resistance in the work of Frigg and Reiss (2009), who have claimed that "the philosophical problems that do come up in connection with simulations are not specific to simulations and most of them are variants of problems that have been discussed in other contexts before" (Frigg and Reiss 2009, 595). Their objection touches upon matters in the metaphysics, the epistemology, the semantics, and the methodology of computer simulations, and assert that any issues stemming from these subjects are already answered by more familiar work on scientific modeling and experimentation. Frigg and Reiss conclude that, although

computer simulations are indeed a scientific novelty, they do not explore completely new and uncharted philosophical territory.

Are Frigg and Reiss correct in uncoupling the scientific novelty of computer simulations from their philosophical novelty? Whereas their objection seems to be ambiguous—after all what does constitute a novelty in philosophy? (Humphreys 2009)—we need to take Frigg and Reiss' concerns seriously if we want to argue that simulation models are units of analysis for genuine philosophical inquiry. To my mind, there are two ways of doing this. A first approach consists in arguing that some issues in connection with computer simulations are *unprecedented* in the philosophical arena. Humphreys (2009) has addressed this line of discussion by showing how the *anthropocentric predicament*[1] is a problem of genuine philosophical value brought up by computer science and has no precedent in the general philosophy of science. A second approach, complementary to the first, consists in showing that computer simulations shed new light on established claims in the philosophy of science, thus proving their (novel) philosophical value. This article advances the second approach. Specifically, I show how the notion of simulation model presented here challenges established philosophical ideas about scientific explanation. If my considerations are correct, this article not only offers an unprecedented and detailed description of simulation models, but it also sheds new light on the philosophical novelty of computer simulations in the philosophy of science.

To achieve these various goals, the article is structured as follows. Section 2 revisits the main interpretations of computer simulation found in the specialized literature. The aim here is to furnish my claim that little has been said about how 'special' simulation models are, even by partisans of the philosophical novelty of computer simulations. I focus my attention on Humphreys' notion of *computational model* as the most complete of these methodologies. Section 3 amplifies and illuminates Humphrey's notion with a study on the architecture of simulation models.[2] Section 3.3 introduces the notion of *recasting* as the most salient methodological feature of simulation models and the core determinant for their architecture. Section 3.4 deals exclusively with distancing recasting from sub-modeling and multi-modeling, two customary practices in scientific modeling. Section 3.5 illustrates the architecture of simulation models with two examples of scientifically relevant computer simulations.[3] Finally, Sect. 4 presents the philosophical novelty of computer simulations understood as challenges posed to the current philosophy of science. This last section argues, *pace* Frigg and Reiss, that standard model-based treatment of scientific explanation cannot accommodate computer simulations. We must keep

[1] The predicament is the question of "how we, as humans, can understand and evaluate computationally based scientific methods that transcend our own abilities" (Humphreys 2009, 617). Humphreys also works on the philosophical novelty of computer simulations in his book (Humphreys 2004).

[2] Because Humphreys is interested in computational science in general, he calls these models *computational models*. Because I am interested in computer simulations in particular, I use the more accurate term *simulation models*.

[3] Whereas it can be shown that the architecture of simulation models fits different computer simulations of varying complexity, the philosophical implications discussed in Sect. 4 are more visible with the type of complex, multi-modeling simulations that researchers are currently implementing.

in mind that it is not within the intentions of this paper to elaborate at large on this complex philosophical issue. Rather, it will suffice to show that using model-based explanations fails to accommodate explanation for computer simulations. The article ends with a short survey of the philosophical tradition where the main claims can be located.

2 Computer Simulations: the Essential Contrast

Philosophers have made numerous efforts to provide a proper characterization, if not a straightforward definition of computer simulations and its distinctive methodological features. Famously, in 1990 Humphreys presented a *working definition* that reads: "[a] computer simulation is any computer-implemented method for exploring the properties of mathematical models where analytic methods are unavailable" (Humphreys 1990, 501). Whereas this working definition still stands up as an accepted characterization for computer simulations, Hartmann has suggested two amendments to it. On the one hand, any definition of computer simulations must also stress the dynamic character of the mathematical model. According to Hartmann, "scientists reserve the term 'simulation' exclusively for the exploration of *dynamic* models" (Hartmann 1996, 84).[4] On the other hand, Hartmann pointed out that computer simulations are also successfully used even when analytic methods are available.

Based on these objections, Hartmann offered his own definition, later adopted by many philosophers (e.g., Guala 2002; Parker 2009): "[s]imulations are closely related to dynamic models. More concretely, a simulation results when the equations of the underlying dynamic model are solved. This model is designed to imitate the time-evolution of a real system. To put it another way, *a simulation imitates one process by another process*. […] If the simulation is run on a computer, it is called a *computer simulation*" (Hartmann 1996, emphasis in original, 83).

Thus understood, Hartmann's definition stands on three statements,

(a) a *simulation* is the result of solving the equations of a dynamic model;
(b) a *computer simulation* is the result of having a *simulation* running on a physical computer;
(c) a *simulation* imitates another process

To marshal the evidence, (a) takes it that the equations of a dynamic model are solved by means of mathematical inference—and thus, the model itself is mathematical. As Hartmann puts it: "[t]he corresponding dynamic model is conveniently formulated in the language of differential equations" (Hartmann 1996, 84). When (a) is complemented with (b), we are in the presence of a *computer simulation*. Taken

[4] 'Exploration' here means finding the space of solutions to the dynamic model. This explains why all five functions of simulations described by Hartmann heavily resemble finding solutions to the underlying model (Hartmann 1996, 84–85).

together, (a) and (b) entail that a computer simulation results when a dynamic mathematical model is implemented on the physical computer.

To illustrate these points, consider Hartmann's example of the Boltzmann-Uehling-Uhlen-beck (BUU) model of the dynamic behavior of low energy nuclear collisions. The model is representative of a typical mathematical model in the sense that it is highly idealized, it abstracts from collisions as well as from relativity theory, and ignores specific quantum interactions, among other characteristics (Hartmann 1996, 93). Since the set of final equations is analytically unsolvable and cognitively intractable, it is a good idea to opt for a computer to do the numerical work. This intuition is well captured by Hartmann and goes along with Humphreys' working definition.

The above definitions have been cogently captured by Frigg and Reiss under the *narrow sense* of simulations, which "refers to the use of a computer to solve an equation that we cannot solve analytically, or more generally to explore mathematical properties of equations where analytical methods fail" (Frigg and Reiss 2009, 596). If this is the predominant interpretation, it follows that the philosophical inquiry that arises in connection with computer simulations are variants of issues already discussed in different philosophical contexts. For instance, claims that the computer solves a mathematical model does not motivate claims about the methodology of computer simulations. Similarly, there is no genuine epistemological interest in computer simulations if their use is only justified when mathematical models have no solutions. In more general terms, and under the current definitions, computer simulations do not raise more philosophical interest than any other method that facilitates calculations.

Thus understood, these definitions treat mathematical models and simulation models on an equal methodological and epistemological footing. Indeed, a computer simulation is no more than a mathematical model solved mechanically, and thus inferences need only to include considerations about the accuracy of results.[5]

More refined accounts of computer simulations entertain the idea that a proper methodology must include extra elements, whether they be extra layers of modeling or the mathematical manipulation of such models. Take, for instance, Winsberg, who argues for a hierarchy of models where, at the end of several transformations, we find the 'computational model' (Winsberg 1999). Morrison also urges for more attention to be given to the methodology of simulation models, which are the "result of applying a particular kind of discretization to the theoretical/mathematical model" (Morrison 2015, 219). To these authors, as well as to many others (e.g., Lenhard 2019;[6] Varenne 2018), the methodology of computer simulations plays a central role in their philosophical assessment, and thus simulation models

[5] For more detail on the implications of these definitions, see (Durán 2019), in particular, the PST viewpoint on computer simulations.

[6] Lenhard is a recent example of taking simulations as a 'special' kind of model. In his 2019 book, he discusses computer simulations as "a new type of mathematical modeling" and further suggests that their 'special' status stems from a handful of general characteristics, such as plasticity, visualization, and others. Yet another interesting case is the book "Mathematics as a Tool" (Lenhard and Carrier 2017), where, again, computer simulations are generally approached as a 'special' kind of mathematical model.

are treated as a 'special' kind of model. However, little efforts are put into working out an architecture for simulation models.

Frigg and Reiss call this second sense of 'simulation' the *broad sense*, which refers "to the entire process of constructing, using, and justifying a model that involves analytically intractable mathematics" (Frigg and Reiss 2009, 596). Thus understood, the broad sense presents a richer view on computer simulations, one that acknowledges, among other things, the existence of a methodology for computer simulations. Properly understood, however, this methodology seeks to locate simulations in the 'methodological map' (Rohrlich 1990; Galison 1996), but it has no interest in discussing the organization, relation, and functionality of the inner constituents of simulation models. In other words, the broad sense is more about figuring out exactly what simulations take from theory and experimentation than to account for the constitutive structures and relations of simulation models. My job here is precisely to elaborate on a detailed account of simulation models, one that also fosters their philosophical novelty.

The exception here is Humphreys, who, after abandoning the working definition, introduced the notion of a *computational model* (Humphreys 2004). To my mind, the sextuple: <Template, Construction Assumptions, Correction Set, Interpretation, Initial Justification, Output Repre-sentation> (Humphreys 2004, 103) is the closest we have today to an architecture of simulation models. Each member of this sextuple carries its own characteristic and raises its own set of technical and philosophical difficulties. A short description indicates that the *template* consists of a computationally tractable theoretical template, that is, a set of partial differential equations with the appropriate initial and boundary conditions suitable for the computer. The *construction assumptions* and the *correction set* are responsible for adjusting the computational model to good representation and computation of the template—they are also responsible for the accuracy of the results. The *interpretation* is originally employed in the construction and acceptance of the template. Finally, the *output representation* is essentially the visualization stage of the simulation (Humphreys 2004, 72ff).

Humphreys' treatment of computational models is not exhaustive, thus leaving sufficient room for further development. One could, for instance, focus on the output representation and argue, together with Perini and others, that visual representations are a genuine, non-linguistic form of scientific arguments (Perini 2005). Drawing from and expanding on Perini's work, one could then show that visualizations of computer simulations can bear truth without resorting to mathematical models (Durán 2018).

My interest here focuses on amplifying and illuminating Humphreys' work by elaborating on a detailed architecture of simulation models. Such an architecture, nevertheless, differs from Humphreys' computational model in several specific respects. One such difference is that I take simulation models to be an amalgamate of mathematical models, databases, integration modules, and other constituents. Furthermore, I bring attention to the techniques of recasting and clustering numerous models into one simulation model. In fact, I believe that Humphreys does not take notice of the multiplicity of models that compose the computational model, but rather of the different stages to which *one* mathematical model—and its multiple

differential equations—undergoes before it becomes part of the template (see his examples in Humphreys 2004, 72). Above and beyond this interpretation, a more detailed analysis of the architecture of simulation models is urgently needed, especially when philosophers are still discussing the epistemology of computer simulation, and much of that discussion is based on the misleading claim that mathematical models are directly implemented on the physical computer.[7] Moreover, if I am correct about the architecture of computer simulations, then we have solid grounds for claims about the philosophical novelty of computer simulations. These are the topics for the following sections.

3 A Survey on the Architecture of Simulation Models

This section surveys the architecture of *simulation models* (*SM*). The *SM*, I will argue, is a rich and complex structure that departs in important ways from standard models used in scientific research. This is the case because the *SM* encompasses at least two different units of analysis (i.e., kernel simulations and integration modules) and involves three forms of relations (i.e., representation, implementation, and integration. See Fig. 1). Furthermore, I will argue that the construction of the *SM* is possible because of a new methodology that is in place. I call it *recasting*, and it consists of clustering a multiplicity of models into one fully computational *SM*. Let me mention that I will exclude from the analysis strategies aimed at choosing the right programming language and the decision process that leads to selecting certain abstract data types over others (e.g., pointers, classes, etc.). I also exclude practices exclusively pertaining to the design and programming of the *SM*, such as the specification of *SM* (see Durán 2018, Section 2.2), the social organization and division of labor involved in developing software, and other general software engineering practices (Pfleeger and Atlee 2010). The focus is then set on elaborating on the rich and complex structure that makes the *SM* a special type of scientific model.

3.1 Models, Kernel Simulations, and Integration Modules

Let us begin with Fig. 1, which illustrates the typical *SM* of the kind of computer simulation of interest. In a bottom-up fashion, we first identify the *kernel simulation* (KS_i), understood as the implementation of mathematical models (M_i) in the formalism of a programming language. Then come the *integration modules* (IM_k), which play two fundamental roles, namely, they integrate external databases, protocols, libraries and the like with the KS_i, and ensure the synchronization and compatibility among them. The purpose of the IM_k, as we will discuss later, is to ensure the computational tractability of the *SM*. Let us begin by briefly clarifying the nature of

[7] This article supports the claim that our general understanding of computer simulations can be significantly advanced if we shift away from the study of simulations as branches of mathematics and physics, and focus more on computer science and software engineering as basic disciplines. This is the position of authors such as Symons and Alvarado (2019), Durán (2018), and Boyer-Kassem (2014).

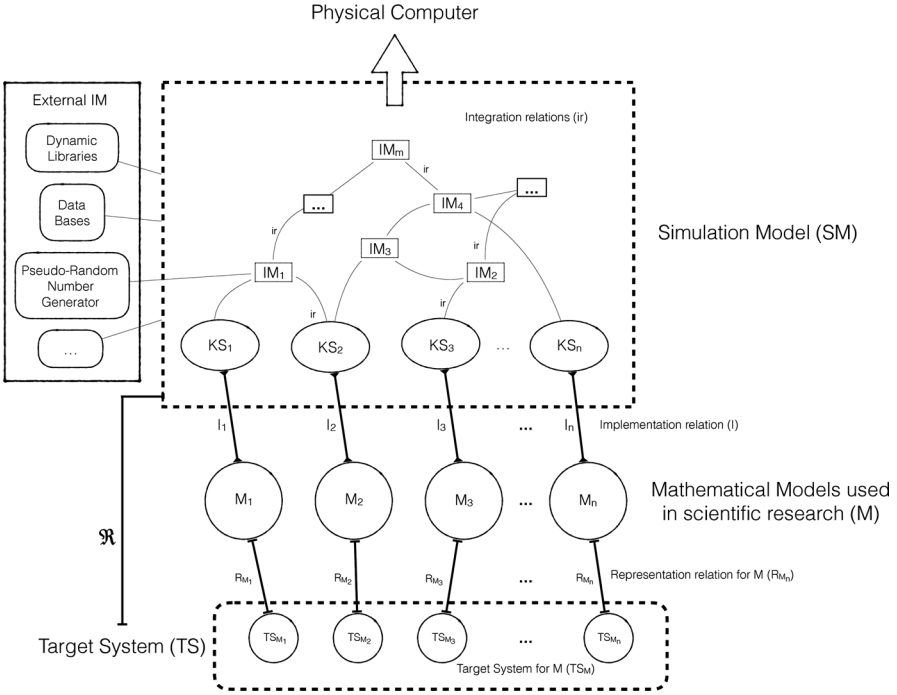

Fig. 1 A general architecture for simulation models

M_i, KS_i, and IM_k, and then move forward to discuss the relations of *representation*, *implementation*, and *integration*.

A first approach takes M_i to be any mathematical model typically present in scientific research. This includes phenomenological models, data models, theoretical models, causal models, and the like (Frigg and Hartmann 2006; Altman 2018), instrumented in a mathematical or logical formalism, pseudo-code (Durán 2018, 46), and similar formats. The Boltzmann-Uehling-Uhlen-beck (BUU) model of dynamic behavior for low energy in nuclear collisions furnishes a good example, but so does the liquid drop model of the atomic nucleus and Schelling's model of segregation. Thus understood, M_i could be any mathematical model used in scientific research with the purpose of implementing it as (part of) a computer simulation. Strictly speaking, the M_i is not part of the *SM*. However, it would be a mistake to remove the use of M_i altogether and favor new ways of modeling real-world phenomena directly into the KS_i (more on this shortly).

The KS_i, on the other hand, is indeed a member of the *SM*. It is understood as an algorithmic structure that implements the M_i in a suitable language capable of computation by a physical computer (on this point, see Durán 2018, 37–54).[8]

[8] We are not interested in discussing the nature of algorithms. Important insights can be found in (Blass et al. 2009; Primiero 2014; Hill 2016); as for programming languages, computational traditions, and software engineering methodologies, the work of (Eden and Turner 2007; Turner 2007; Eden 2007; Turner 2018) added much clarity to the problem.

Examples of KS_i include the calculation of the Greatest Common Divisor (Durán 2018, 47), the implementation of equations of energy, angular momentum, and inward force for an orbiting satellite (Woolfson and Pert 1999a), and many other cases. A closer look on how M_i are successfully implemented into KS_i is discussed in more detail in Sect. 3.2, but the classic source of examples is Knuth (1973).

It is interesting to note that current simulational practice sometimes allows mathematical and logical formalism the be omitted in favor of a ready-made algorithmic structure. This means that, on an increasing number of occasions, researchers prefer to dispense themselves the trouble of first developing an M_i and then figuring out its implementation as a KS_i, by directly coding their systems as a KS_i. For instance, (DeAngelis and Grimm 2014; Peck 2012) show how a simulation model—or parts of it—may be nothing more than an algorithm that frames the agents' behavior. The model's representation takes place directly at the level of algorithmic structures KS_i and without mediating any standard M_i. The representation, then, is built up from suspected relational structures abstracted from the target system and directly coded as KS_i. For such a case, the KS_i simply corresponds to the M_i.

In addition to M_i and KS_i, we also find *integration modules* (IM_k) as central components of the *SM*. IM_k are meant to facilitate the integration, connection, and functionality among all KS_j as a fully functional and computable *SM*. An IM_k is understood as a computationally tractable algorithmic structure that fulfills different functionalities and purposes, such as managing routines of control and performance checks, executing protocols, ensuring I/O operations, and providing access to external databases, among other tasks. It is not within the scope of an IM_k, however, to hold representational relations to a target system in the same way that the M_i and KS_i do (see Sect. 3.2). A good example of IM_k, then, are computer shared libraries that allow a KS_i written in FORTRAN to be fully functional in a KS_j written in Language C. Another example are off-the-shelf databases that feed the simulation with the relevant data, like the International Air Transport Association database used by GLEaM, the simulation of the dynamics of epidemic transmission discussed in Sect. 3.5.

The rather abstract ideas presented so far will be made clear in Sect. 3.5 where I discuss two examples of computer simulations as used in contemporary research. I will also be discussing the philosophical intake of KS_i and IM_k in Sect. 4. For now, it is sufficient to recap our findings that M_i are implemented into KS_i and integrated with IM_k in order to conform to a fully functional, computationally tractable *SM*. This interpretation of *SM* deviates significantly from the general idea that philosophers have of simulation models and computer simulations. Unlike the portrayal of simulations computing a single mathematical model which render results (Humphreys 1990, 1996; Guala 2002), most models implemented as simulations in fact require a multiplicity of models, input of data from different sources, *ad-hoc* solutions for the internal communication of modules in the simulation, and the like. Figure 1 is a close description of the architecture of many simulation models.

3.2 The *Representation* Relation, the *Implementation* Relation, and the *Integration* Relation

Pertaining to the architecture of the *SM*, there are three relations of importance, namely, the *representation* relation, the *implementation* relation, and the *integration* relation. The *representation relation* can be further subdivided into two types. That is, R_{M_i} understood as the representation relation that holds between an M_i and its sub-target system TS_{M_i}, and $\mathfrak{R}$ understood as the representation relation holding between the *SM* and the target system *TS*. Taken individually, a given R_{M_i} can be characterized in terms of the philosopher's favorite representation relation, such as isomorphism, homomorphism, similarity, or any other. Taken together, however, the representational relation ($\mathfrak{R}$) for the whole *SM* can be identified neither with any individual nor joint combination of R_{M_i}. $\mathfrak{R}$ is, therefore, a representational relation of significant philosophical value. Strikingly, philosophers have seldom put any efforts into theorizing the representation relation $\mathfrak{R}$.[9] Instead, philosophers either assume that the *SM* somehow inherits the representational capacity of the *M* implemented (see the advocates of the PST viewpoint Durán 2019), or it is claimed without further argumentation that the *SM* holds a representational relation with real-world phenomena (see the partisans of the DPB viewpoint Durán 2019).

Admittedly, I am not offering a theory of representation for *SM*, and thus I am subject to my own criticism. Instead, I engage in the much simpler job of pointing out what needs to be considered in order to satisfy such a future theory. To see this, consider the following. For a multiplicity of M_i to be receptive to becoming part of the *SM*, there must exist an equal multiplicity of R_{M_i} for each TS_{M_i} (i.e., one representational relation for each corresponding model and sub-target system). It follows that the representational relation for the whole *SM*, i.e., $\mathfrak{R}$, needs to be philosophically evaluated separately as a unique relation. This for two reasons. First, because, as mentioned, neither R_{M_i} stands out as representative for $\mathfrak{R}$. This is true for all instances except for the simple case of implementing a unique M_1 (hence, $M_1 \equiv SM$). In such a trivial case, $R_{M_i} \equiv \mathfrak{R}$.[10] Second, because the target system of the simulation model (TS_M) could not be identified with, nor is the joint composition of each individual target system TS_{M_i}. The exception is, again, the case of a unique model M_1 where TS_{M_1} *is* the TS_M. In this case $TS_{M_1} \equiv TS_M$.[11]

Thus far, I have assumed that each R_{M_i} holds for an M_i and its target system TS_{M_i}. However, this assumption only aims maintaining some degree of simplicity in the architecture of simulation models, for it does not hold for all types of simulations. There is an increasing use of *SM* that implement M_i whose R_{M_i} may simply not exist. Illuminating cases are found in economics (Grüne-Yanoff 2009; Mäki 2009) and

[9] To the best of my knowledge, only Bueno (2014) has offered a theory of representation for computer simulations. For criticism, see (Durán 2019).

[10] There is the possibility that all the R_{M_i} are of the same type, say, similarity. In such a case, $\mathfrak{R}$ could only be taken to be of the same type of all R_{M_i}. However, the differences between each TS_{M_i} and target system of the simulation model TS_M that I discuss next still remain.

[11] Let us note that the interpretation offered by Humphreys (2004, 2009), Winsberg (1999, 2010) are special cases to my architecture where the two exceptions apply.

neuroscience (Chirimuuta 2013), both with applications in computer simulations. Discussing such cases will introduce a number of subtleties that this article cannot handle at this stage. The reader is advised, however, that the architecture of *SM* here presented is not intended to account for such complex R_{M_i} and M_i.

As for the *implementation relation* (I_i), that is, the relation that enables the application of M_i as an algorithmic structure KS_i, there are several ways to deal with it. Typically, the question to answer is how a physical system P (e.g., the physical computer, the brain) implements an abstraction C (e.g., an algorithm or a mental calculation). Chalmers has famously argued that a physical system P implements a computation C when the causal structure of P mirrors the formal structure of C (Chalmers 1994, 392). In this view, implementing an algorithm involves instantiating the appropriate pattern of causal interaction among internal states of the computer (i.e., those that P mirrors from C).[12] This position, known as the structuralist view of computational implementation,[13] is shared by many proponents (Copeland 1996; Dresner 2010 Godfrey-Smith 2008; Scheutz 2001). On the opposite side is Rescorla, who argues that structuralism predicts incorrect implementation conditions for some, though perhaps not all, computations. To Rescorla's mind, it is perfectly possible to find cases where P implements C in conditions where P does not mirror the formal structure of C (Rescorla 2013, 683).

Unlike these positions, I am not interested in theorizing on the implementation relation between a physical system and an abstract one, but rather on the implementation relationship between M_i and its respective KS_i, both being abstract entities. I understand that this choice neglects the limitations that the physical computer imposes on the computation of KS_i, such as the order of precision in the results. But insofar as we keep in mind that any KS_i needs to be implemented on the physical computer, and thus be computable, and that the physical implementation imposes restrictions on the realizability of that KS_i as discussed by Chalmers and Rescorla, then momentarily ignoring the implementation on the physical computer should not constitute a problem for us.[14]

The notion of implementation of interest here can be approached semantically and methodologically. Regarding the former, I draw on Rapaport's idea that *implementation is semantic interpretation* (Rapaport 1999, 2005, 2019). In its simplest form, this means that a semantic domain (e.g., KS_i), standardly characterized by rules of symbolic manipulation, *interprets* a syntactic domain (e.g., M_i) (Rapaport 1999, 110). Interpretation here is taken as a correspondence or mapping relation between the properties, structure, operations, states, and the like of the syntactic

[12] This seems to be the assumption that drives Parker (2009) to claim that the materiality of the physical computer plays some relevant role in the interpretation of the results of computer simulations (Durán 2013, 2018).

[13] Not to be confused with other forms of structuralism in the literature, such as structuralism about the natural numbers (Benacerraf 1965; Halbach and Horsten 2005) and structural realism (Ladyman 2016), among others.

[14] *Computational Reliabilism* is a promising framework for entrenching the epistemic reliability of any computational system, including computer simulations (Durán and Formanek 2018).

domain into the semantic domain (Rapaport 2005, 386–389).[15] To illustrate these points, consider a simulation of a satellite orbiting around a planet under tidal stress. For such a case, researchers might want to implement standard equations of energy, angular momentum, and inward force (Woolfson and Pert 1999a). In order to implement such equations (i.e., M_i) into KS_i, a series of functions, data types, conditionals and control flow statements among other instructions must be in place. Take for instance the implementation of the equations of angular momentum as presented in Woolfson and Pert (1999b). These equations are interpreted as the summation of the squares of three real variables representing the position of the satellite. Each one of these variables correspond, in turn, to a three-dimensional array corresponding to the FORTRAN programming language.

Let us further note that implementing any M_i into a KS_i also triggers a series of processes and relations—many of which the researchers do not have control over nor program themselves—aimed at making the implementation computable. For instance, the equations of angular momentum require, during the execution of the simulation, special allocations of memory (i.e., as arrays), interpretation of the data types involved (i.e., real data type), the execution of protocols in the operating system (e.g., for avoiding deadlocks), etc. At any rate, the logical analysis of I_i shows that the implementation relation consists of a series of mapping of the entities, structures, relations, operations, and the like found on the M_i into an appropriate algorithmic way (i.e., the KS_i).

From a methodological perspective, I_i also includes a number of discretization techniques and *ad-hoc* solutions (Winsberg 1999). For instance, there is an explicit decision of implementing the stress of the satellite by three masses connected by springs each of the same unstrained length (Woolfson and Pert 1999a). This is an example of a typical *ad-hoc* solution for a problem where the stress of the satellite cannot be represented by a point mass. To ensure high degrees of fidelity, the KS_i inherits core features of the M_i that it is implementing.[16] For instance, if the M_i is a nonparametric statistical model, then the KS_i will include a call to a pseudo-number engine, the treatment of infinity, and a way to represent the distribution of spaces; if the M_i represents the time-evolution of two variables, then the KS_i must also represent its dynamics in finite time. These are a few among several chief characteristics that simulations maintain in order to offer reliability and trustworthiness of their results (Durán and Formanek 2018).

Finally, the *integration relation* (*ir*) can be epistemically and methodologically understood as being similar to I_i. The *ir* connects or binds KS_i with IM_k for the purposes of ensuring, among other things, synchronization and compatibility among the KS_i. The *ir*, then, can be simply seen as linkages of IM_k for the compatibility of SM. In this respect, I do not think that the *ir* has any philosophical relevance in itself, and

[15] It is important to keep in mind that the notions of 'semantic' and 'syntactic' used by Rapaport do not attempt to capture philosophical viewpoints on theories and models.

[16] I am loosely reinterpreting Weisberg's *fidelity criteria* as describing how similar the KS_i must be to the M_i in order to be considered a computationally tractable implementation (Weisberg 2007, 221).

thus I shall not pursue any further characterization of it. It is important to mention it, however, for the sake of technical completeness.

3.3 Recasting

Frequently, philosophers take the analysis of computer simulations to be exhausted in the fact that a model is implemented as a simulation. We have seen this in the work of Hartman, Parker, and Guala, who take that a given *M* is somehow directly computed by a physical machine. Winsberg and Humphreys, on the other hand, argue that a given *M* is implemented into the *SM* by alterations of *M* and extra levels of modeling (for more on these points, see Durán 2019). I submit that the design and construction of an *SM* entails more processes and a higher level of complexity than accounted so far.

Unlike these philosophers, I take the *SM* to be a more complex structure, one with two salient characteristics. First, it clusters a *multiplicity* of M_i into a fully operational *SM*. Second, the *SM* includes modules, structures and aggregations that go beyond the mere alteration and implementation of M_i into the corresponding KS_i. These two characteristics are an indication of the richness and complexity of the *SM*, and ultimately speak in its favor as a unit of philosophical analysis in its own right. I call *recasting* the whole process of converting the *SM* into a fully operational simulation. To be more specific, the process of recasting is divided into three main procedures, namely, the *implementation* of the multiplicity of M_i into their corresponding KS_i, the *integration* of KS_i into the *SM* by means of modules and structures IM_k, and the *aggregation* of the external IM_k into the *SM*. Let us discuss them in turn.

The *implementation* relation of M_i into KS_i has been discussed earlier. By bringing it up again, I intend to draw attention to the fact that it is not *one* M_i implemented as the algorithmic structure KS_i, as computer simulations are often addressed in the specialized literature, but rather of a multiplicity of M_i that are ultimately clustered in the *SM*.

The *integration step*, on the other hand, has largely been neglected by the dominant literature on computer simulations and does require our attention. In an increasing number of contexts, the *SM* implements a multiplicity of M_i, many of which are of different kind (e.g., phenomenological models, data models, theoretical models Frigg and Hartmann 2006) and hold different structures (e.g., time and space scale, grid resolution, etc.). In this respect, a fully functional, computationally tractable, and representationally sound *SM* requires all KS_i to be integrated in the right way.[17] To this end, researchers make use of IM_k to ensure full cohesion and operationalization. For instance, if a KS_i implements a non-parametric statistical model M_i whereas another KS_j implements a M_j that represents the time-evolution of two variables, then there must be an IM_k in place that either makes the two *KS* compatible

[17] Let it be noted that I group under the same umbrella a series of disparate computer processes: a database and a daemon are different in fundamental respects. Nevertheless, and for the purposes of this article, I see no objection in referring to all of them as *integration modules* (IM_k).

with each other, or collects the outcome of each individual *KS* into the input for a third KS_m.

Thus understood, integration modules IM_k are binding units into the *SM* with the purpose of making the latter computable. As discussed earlier, there are multiple purposes for the IM_k, including providing cohesion and interaction among the KS_i (e.g., communication protocols, synchronization in parallel and distributed computing, synchronization of sub-processes); enabling their integration (e.g., time and space multi-scale integration, parameter integration); and coupling, understood as the degree of interaction between IM_k and KS_i (e.g., deadlock detection in distributed databases with its error control). Furthermore, IM_k are operationalized to uphold standards of tractability for the *SM* (e.g., response time and processing speed), computability (e.g., managing routines of error control, performance checks, and I/O), and modularity (e.g., scalability, maintenance), among others. It is indeed in virtue of recasting a number of M_i into KS_i that differ in scale, parameters, resolution, and the like, that many IM_k gain their value: they are the binding units that ultimately enable the *SM* to be a successful simulation.

Finally, the *aggregation step* consists in including external IM_k as a constitutive part of the *SM*. Typical examples are databases, pseudo-random number engines, I/O and basically any other form of software, library, structured data, etc., that contribute to the success of the simulation. Let us note that none of these are related to the implementation of an M_i into the *SM*. Section 3.5 shows an example where the IATA database is constitutional for the success of a medical simulation, and which cannot be taken as part of any of the M_i implemented.

Recasting, then, makes plain the structural richness of *SM*, their value as units of analysis in their own right, and ultimately provides the basis for claims about the philosophical novelty of computer simulations. Before ending this section, there is one more point to discuss, that is, that recasting is possible because of a new form of *abstraction* uniquely found in computational systems. Typically, the construal of a given M_i requires the researcher to subject the phenomenon in the real-world to processes of abstraction, idealizations, and fictionalizations, thus having to decide which aspects of the phenomenon will actually be considered (Weisberg 2007). The *SM* does not necessarily abstract from the real-world, but rather inherits, so to speak, such abstractions from the M_i. To this abstraction, we also need to factor in the integration and aggregation steps just discussed. In this context, recasting is impossible without tools that hide, but do not neglect, details about the implementation of each M_i, the integration of each KS_i, and aggregations of each IM_k. This is to say that the properties, structures, operations, relations and the like present in each M_i can be effectively implemented into the KS_i without stating explicitly how such implementation is carried out (the same can be said about the integration and aggregation steps). Colburn and Shute call this form of abstraction *information hiding*, and to them it constitutes a novel form of abstraction introduced by computer science (Colburn 1999; Colburn and Shute 2007). Examples include abstracting from the details how the messages among computer processes are passed on, how the computer hardware represents the value of parameters, and how programming languages handle irrational numbers, among other exclusively computational related issues (Colburn and Shute 2007). Owing to information hiding, researchers are entitled to

several claims, including that the process of recasting a multiplicity of M_i and IM_k into a fully functional *SM* is done minimizing the loss of information, that $\mathfrak{R}$ represents the target system in the intended way, that the *SM* is a reliable model (Durán and Formanek 2018), and that the results of *SM* are (approximate) correct of the target system, among others.

Before illustrating the architecture of *SM* with two examples and discussing further their philosophical implications, let me briefly examine *sub-modeling* and *multi-modeling* as two known practices in modeling that could be deemed as candidates for replacing recasting. If my arguments against sub-modeling and multi-modeling as forms of recasting is convincing, then I believe we are in the presence of a new form of modeling.

3.4 Sub-modeling and Multi-modeling

Sub-modeling is another term for partitioning an unmanageable model into smaller, easier-to-handle parts. In this respect, it is more a modeling strategy than a way to conceive of the model. Once a scientific model has been partitioned, each sub-model is a representational unity in itself that addresses different aspects of a phenomenon, and which is decoupled from other sub-models. The aim, then, is not to integrate a diversity of models into a larger and more encompassing model, but rather to 'disintegrate' a unity into simpler, more manageable, ontologically similar units. In other words, the researcher's motivations, aims and final results in sub-modeling differ in relevant aspects from recasting. Bailer-Jones puts this issue in the following way: "[p]ortioning the problem-solving task in this way makes the individual tasks much more manageable, and scientists with their specific expertise can work on individual submodels, while remaining comparatively unconcerned with the issues arising in other submodels" (Bailer-Jones 2009, 132).

Naturally, at a later stage the sub-models need to be integrated back into the original model. This methodological step is, again, hardly the same case as recasting where there is no breakup of the model in the first place, and the main aim consists in integrating a disperse and—in most cases—incompatible set of models into a computable unity. Unlike sub-modeling, then, in recasting researchers cannot remain unconcerned with issues rising from the different M_i, their implementation as KS_i, and the integration with IM_k.

Multi-modeling, on the other hand, is another term for a hierarchy of models that are neither equally valid nor equally successful. Each model in the hierarchy has its proper functions, purposes, representational capacity, and limitations that differ from other models in the hierarchy. Now, since multi-modeling imposes a complexity that the analytic treatment is not always prepared to cope with, they are kept relatively simple. This means that their treatment is of a simple individual model within the hierarchy.

Similar to sub-modeling, in multi-modeling each model might be decoupled from the others. Moreover, just like sub-modeling, the main objective in multi-modeling consists in decomposing or partitioning rather than composing a target system. Unlike sub-modeling, however, multi-models are models in themselves, located

somewhere in a hierarchy of models, and not a 'piece' of a larger model. Thus understood, there is a sense in which multi-modeling resembles recasting and *SM* to some detail.

Suppose for a moment that recasting entails a hierarchy of models. This means that, in order to accommodate multi-modeling to a form of recasting, we need each individual model in the multi-model hierarchy to relate, *vis á vis* each KS_i in the simulation model. In other words, the *SM* is conceived as a well-structured hierarchy of kernel simulations. It is in this way that the *SM* becomes a structural image, hierarchically organized, of a multi-model. Unfortunately, this is as far as the analogy can be established. First, this image does not capture what researchers typically call a computer simulation. Although there might be some cases of a hierarchical *SM*, those are very unusual in the actual practice of computer simulations. Furthermore, and as I discussed in Sect. 3, a computable and fully functional *SM* also requires the integration of KS_i via IM_k. This means that the hierarchy of models must also reflect these extra modules. However, the inclusion of IM_k breaks the hierarchical structure since their job is, precisely, to integrate the multiplicity KS_i into a computational (not necessarily hierarchical) *SM*.

Admittedly, we cannot logically exclude the possibility that, in some very specific cases, recasting takes the form of multi-modeling. However, we have made its occurrence so rare and unlikely that it is safe to conclude that recasting, and along with it the *SM*, are not part of a charted philosophical territory. Let us now have a brief look at the architecture of computer simulations with an actual working example.

3.5 Example: Medical Simulations

In order to illustrate the architecture of computer simulations, consider two simulations of an epidemic outbreak. Ajelli et al. (2010) provide a side-by-side comparison of a stochastic agent-based model and a structured meta-population stochastic model (GLobal Epidemic and Mobility-GLEaM). The agent-based model includes an explicit representation of the Italian population through highly detailed data on the socio-demographic structure—in our parlance, KS_1 implements M_1. In addition, and for determining the probability of commuting from municipality to municipality, Ajelli et al. use a general gravity model used in transportation theory—that is, KS_2 implements M_2. However, the epidemic transmission dynamic is based on an ILI (Influenza-like Illness) compartmentalization based on stochastic models that integrate susceptible—M_3, latent—M_4, asymptotic infections—M_5, and symptomatic infections—M_6 (Ajelli et al., 2010, 5). In our schemata, each M_i, $3 \leq i \leq 6$ represents a type of infection TS_{M_i} and it is implemented as a KS_i in the *SM*. The authors define their agent-based model as "a stochastic, spatially-explicit, discrete-time, simulation model where the agents represent human individuals […] One of the key features of the model is the characterization of the network of contacts among individuals based on a realistic model of the socio-demographic structure of the Italian population" (Ajelli et al. 2010, 4). Typically, *IM* are not explicitly mentioned in general descriptions of the simulations, but they are nonetheless central during the stages of designing and coding of the actual *SM*. In this case, the authors mentioned that both

GLEaM and the agent-based model are dynamically calibrated in that they share exactly the same initial and boundary conditions. Such calibration is carried out by a module *IM* that integrates both simulations (Ajelli et al. 2010, 6).

On the other hand, the GLEaM is a multiscale mobility network based on high-resolution population data that estimates the population with a resolution given by cells of 15×15 min of arc. Balcan et al. (2009) explain that a typical GLEaM consists of three data layers. A first layer, where the population and mobility are implemented, allows for the partitioning of the world into geographical regions—i.e., KS_1 implements M_1. This partition defines a second layer, the sub-population network, where the inter-connection represents the fluxes of individuals via transportation infrastructures and general mobility patterns—KS_2 implements M_2. Finally, and superimposed onto this layer, is the epidemic layer, that defines the disease dynamic inside each sub-population—KS_3 implements M_3—(Balcan et al. 2009). In the study conducted by Ajelli et al., the GLEaM also represents a grid-like partition where each cell is assigned the closest airport (this could actually be another M_i or simply an external IM_k, like a database). The sub-population network uses geographic census data—IM_1—and the mobility layers obtain data from different databases, including the International Air Transport Association database consisting in a list of airports worldwide connected by direct flights (i.e., $\{IM_2, IM_3, \ldots\}$).

The example of the two computer simulations give us a good sense of the interconnectivity between different M_i, their implementation as KS_i, and the final integration into a fully functional *SM* via IM_k. Let us now summarize our findings. In a given *SM*, several IM_k integrate a number of KS_i for computational purposes. Each KS_i, in turn, implements a multiplicity of M_i in such a way that is computationally tractable. Since most of the structures found in an M_i can be reconstructed in terms of a given programming language, it is reasonable to take that KS_i is an accurate implementation of such M_i. The process of information hiding discussed earlier warrants the implementation.

Let us now turn to my last discussion, where I address the philosophical novelty of *SM*.

4 The Philosophical Novelty of Computer Simulations: the Case of Scientific Explanation

The previous sections made an effort to present and discuss the architecture of simulation models, and to show how this structure better accounts for the current practice of simulation modeling. I argued for a rich and complex structure that builds from different sources ranging from mathematical models to sizable external databases. I now argue how, by means of taking *SM* as units of analysis in their own right, we could give content to the idea that computer simulations introduce new challenges to the philosophy of science. To make this visible, I briefly revisit two recent discussions on scientific explanation for computer simulations and show why the whole explanatory enterprise fails when taking simulations as the simple implementation of mathematical models.

The first attempts to elaborate on the logic of scientific explanation for computer simulations is found in the works of Krohs (2008) and Weirich (2011).[18] To these authors, a mathematical model *M* (in Krohs' terminology, a theoretical model) explains a real-world phenomenon (RW_p) in the usual way, that is, by relating properties of the former that account for the latter. On occasion, the *M* includes equations that are too complex for humans to solve analytically, and therefore the representation of RW_p cannot be guaranteed. This means that an explanation of RW_p cannot take effect. Computer simulations amend this by finding the set of solutions to *M*, and thus, claim Krohs and Weirich, reestablish the representational relation to RW_p. As Krohs summarily puts it: "[t]he explanatory relation holding between simulation and real world is therefore an indirect one. In the triangle of real-world process, theoretical model, and simulation, explanation of the real-world process by simulation involves a detour via the theoretical model" (Krohs 2008, 284). Thus understood, computer simulations play the instrumental role of solving *M*, but not the explanatory role of accounting for RW_p, which ultimately is left to *M*. It is in the sense of being a mere instrument that computer simulations do not pass the mark of what constitute a philosophical issue.

Opposing this view is my own work (2017), where I argue that the logic of explanation for computer simulations must include the *SM* into the explanatory relation. To this end, it must be first shown that the *SM* is in a better position to explain than any M_i. This is accomplished precisely by noticing that a given *SM* is a collection of models, relations, and external modules that exceed any one M_i. Furthermore, I call attention to the fact that researchers are interested in explaining the results of their simulations with the purpose of *understanding* real-world phenomena, but not to explain the latter directly. Decoupling explanation from understanding allows me to fully circle the logic of explanation for computer simulations. Indeed, researchers first explain the results of their simulations, and only after a further epistemic step are they able to understand something about the real world (Durán 2017, 40).

A simple example will help to contrast the two approaches. Consider a simulation of a satellite orbiting a planet under tidal stress, as presented by Woolfson and Pert (1999b). The visualization of the results shows a series of spikes that represent the tidal stress that the satellite undergoes during orbiting. Researchers are naturally inclined to first explain the behavior of the simulated satellite, such as the spikes (i.e., the results of the simulation), before designing, building, and operating the real satellite. In the context of this example, I call attention to the fact that the visualization shows the spikes having a steadily downwards trend, an effect that is the result of a truncation error during the computation of *SM*. I conclude that, if an *M* exogenous to the simulation were to be used for explanatory purposes (v.gr., a mathematical model of the planetary orbit), as Krohs and Weirich propose, then researchers would either be unable to account for the trend downwards or, having explained the spikes with *M*, they would have wrongly ascribed the downwards trend to the real

[18] The works of Miłkowski (2016) and Fernández (2003), for instance, are better understood as attempts to have explanation using computer simulations.

world (i.e., they would have misleadingly claimed that the satellite's orbit will eventually become circular). I then show how, by reconstructing the *SM* as the *explanans*, researchers are able to account for this trend and thus understand the overall behavior of the satellite. Motivated by these results, I offer an in-depth account of explanation for computer simulations that, as a matter of fact, includes the *SM* in the explanatory relation.

The fundamental difference between my approach, on the one hand, and Krohs and Weirich, on the other, is that we understand computer simulations in radically different ways. I build my claims around the idea that the *SM* is a rich and complex model, with methodological and epistemological value in and by itself. Krohs and Weirich, echoing authors such as Hartmann (1996), Frigg and Reiss (2009), Parker (2009), and others, mingle the analysis of simulations with the analysis of other forms of mathematical models.

Now, if these claims are correct, we find ourselves with a handful of genuine philosophical issues attached to the logic of scientific explanation for computer simulations. On the one hand, and taking my approach seriously, further arguments need to be provided on how and to what extent researchers are able to access and reconstruct the *SM* for the purpose of explaining. Although I offer a detailed reconstruction of the *SM* as the *explanans* for the simulation of the satellite, I also notice that cognitively accessing every function, procedure, and data-type in the *SM* is challenging. In this respect, the *right levels of description* for the *explanans* is a novel issue that needs to be discussed in detail if this account is to be successful. On the other hand, a central philosophical question that seeks an answer is how, by explaining the results of a computer simulation, researchers understand RW_p. These issues draw on problems related to representation, realism, and the notion of understanding in the context of computer simulations. Unfortunately, these topics have also received little attention in the specialized literature (except for Bueno (2014). See footnote 9). The bottom line is that we have on our hands a computationally-based explanation that differs in important aspects from a model-based explanation, and thus offers significant philosophical value.

5 Final Remarks

The philosophical analysis of computer simulations must recognize the rich structure and methodology of *SM*, and refrain from reducing the discussion to the mere implementation of an *M* onto a physical computer. The latter way of looking at computer simulations impoverishes their nature and importance in scientific practice, ultimately assimilating simulations as tools with mere instrumental value.

Over the past years, much has been discussed concerning the heterogeneity of models and the myriad of methods by which they can be constructed, applied, and evaluated in their own right. The works of Suárez and Cartwright on the *piecemeal* borrowing of models from a range of different domains (Suárez and Cartwright 2008), as well as the many works collected under the idea of models as *mediating* between our theories and the world (Morgan and Morrison 1999), furnish two good examples of these kinds of philosophical discussions. The work presented here is an

attempt to support the claims behind these and other philosophers who share similar viewpoints. I depart from them, however, in that I focus on a new kind of scientific model, that is, the simulation model. I am, of course, not alone in this enterprise. As mentioned, Humphreys (1990) already attempted to distinguish mathematical models from computational methods, and authors sensitive to the methodology of computer simulations have followed a similar path (Winsberg 2010; Morrison 2015; Varenne 2018; Symons and Alvarado 2019).

My approach departs from theirs on several accounts, including the use of studies in computer science and software engineering as the most suitable body of knowledge for understanding simulation models. With this firmly in mind, I showed that recasting is a significant methodological move that brings new insight into studies on simulation models and computer simulations. Moreover, recasting makes sense of the fact that simulation models handle many parameter values and thus hold great representational capability and model expressiveness, arguably the most celebrated characteristics of computer simulations along with their computing power.

Admittedly, with some discussions, I have only scratched the surface of what is now deemed a pressing issue. This is most visible in my treatment of the novelty of simulation models for explanations. In spite of these—and other, possibly unknown—shortcomings, the article also accomplishes its original aim of taking Frigg and Reiss' objections to computer simulations as philosophically novel seriously, as well as offers an in-depth, unprecedented architecture of simulation models that could be—and, in many cases should be—the basis for the philosophical study of computer simulations.

Acknowledgements Thank are due to Björn Schembera for discussions on the architecture of simulation models, and to Johannes Lenhard and Nico Formanek for their close reading of the article.

References

Ajelli, M., Gonçalves, B., Balcan, D., Colizza, V., Hu, H., Ramasco, J. J., et al. (2010). *Comparing Large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models., 10*(190), 1–13.

Altman, A. A. (2018). Causal models. In: *The Stanford Encyclopedia of Philosophy (Summer 2019 Edition)*. https://plato.stanford.edu/archives/sum2019/entries/causal-models/.

Bailer-Jones, D. (2009). *Scientific Models in Philosophy of Science*. Pittsburgh: University of Pittsburgh Press.

Balcan, D., Colizza, V., Gonçalves, B., Hu, H., Ramasco, J. J., & Vespignani, A. (2009). Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences, 106*(51), 21484–21489.
Barberousse, A., & Marion, V. (2013). *Computer Simulations and empirical data*. Newcastle upon Tyne: Cambridge Scholars Publishing.
Beisbart, C. (2012). How can computer simulations produce new knowledge? *European Journal for Philosophy of Science, 2,* 395–434.
Benacerraf, P. (1965). What numbers could not be. *The Philosophical Review, 74*(1), 47.
Blass, A., Dershowitz, N., & Gurevich, Y. (2009). When are two algorithms the same? *The Bulletin of Symbolic Logic, 250,* 145–168.
Boge, F. J. (2019). Why computer simulations are not inferences, and in what sense they are experiments. *European Journal for Philosophy of Science, 9*(1), 13. https://doi.org/10.1007/s13194-018-0239-z.
Boyer-Kassem, T. (2014). Layers of models in computer simulations. *International Studies in the Philosophy of Science, 28*(4), 417–436.
Bueno, O. (2014). Computer simulation: An inferential conception. *The Monist, 97*(3), 378–398.
Chalmers, D. J. (1994). On implementing a computation. *Minds Mach, 4*(4), 391–402.
Chirimuuta, M. (2013). Minimal models and canonical neural computations: The distinctness of computational explanation in neuroscience, explanation and explain, explanation and explain and explanatory, explanation and explain and explanatory and simulation. *Synthese, 191*(2), 127–153.
Colburn, T. R. (1999). Software, abstraction, and ontology. *The Monist, 82*(1), 3–19.
Colburn, T., & Shute, G. (2007). Abstraction in computer science. *Minds and Machines, 17*(2), 169–184.
Copeland, J. (1996). What is computation? *Synthese, 108*(3), 335–359.
DeAngelis, D. L., & Grimm, V. (2014). Individual-based models in ecology after four decades. *F1000Prime Reports, 6*(39), 1–6.
Dresner, E. (2010). Measurement-theoretic representation and computation-theoretic realization. *Journal of Philosophy, 107*(6), 275–292.
Durán, J. M. (2013). Computer simulations and the changing face of scientific experimentation. In J. M. Durán & E. Arnold (Eds.), *Computer simulations and the changing face of scientific experimentation* (pp. 76–98). Newcastle upon Tyne: Cambridge Scholars Publishing.
Durán, J. M. (2017). Varieties of simulations: From the analogue to the digital. In M. K. A. Resch & P. Gehring (Eds.), *The Science and art of simulation* (pp. 175–192). Berlin: Springer.
Durán, J. M. (2018). *Computer simulations in science and engineering. Concepts—practices—perspectives*. Berlin: Springer.
Durán, J. M. (2019). A formal framework for computer simulations: Surveying the historical record and finding their philosophical roots a formal framework for computer simulations: Surveying the historical record and finding their philosophical roots. *Philosophy & Technology*. https://doi.org/10.1007/s13347-019-00388-1.
Durán, J. M., & Formanek, N. (2018). Grounds for trust: Essential epistemic opacity and computational reliabilism. *Minds and Machines, 28*(4), 645–666.
Eden, A. H. (2007). Three paradigms of computer science. *Minds Mach, 17*(2), 135–167.
Eden, A. H., & Turner, R. (2007). Problems in the ontology of computer programs. *Applied Ontology, 2*(1), 13–36.
Fernández, J. (2003). Explanation by computer simulation in cognitive science. *Minds and Machines, 13,* 269–284.
Frigg, R., & Hartmann, S. (2006). *Scientific Models* (pp. 740–749). Abingdon: Routledge.
Frigg, R., & Reiss, J. (2009). The Philosophy of simulation: Hot new issues or same old stew? *Synthese, 169*(3), 593–613.
Galison, P. (1996). Computer simulations and the Trading Zone. In P. Galison & D. J. Stump (Eds.), *The disunity of science: Boundaries, contexts, and power* (pp. 118–157). Palo Alto: Stanford University Press.
Godfrey-Smith, P. (2008). Triviality arguments against functionalism. *Philosophical Studies, 145*(2), 273–295.
Grüne-Yanoff, T. (2009). Learning from minimal economic models. *Erkenntnis, 70*(1), 81–99.
Guala, F. (2002). *Models, simulations, and experiments* (pp. 59–74). Berlin: Kluwer Academic.
Halbach, V., & Horsten, L. (2005). Computational structuralism. *Philosophia Mathematica, 13*(2), 174–186.

Hartmann, S. (1996). The world as a process. In R. Hegselmann, U. Mueller, & K. G. Troitzsch (Eds.), *Modelling and simulation in the social sciences from the philosophy of science point of view* (pp. 77–100). Berlin: Springer.
Hill, R. K. (2016). What an algorithm is. *Philos Technol, 29*(1), 35–59.
Humphreys, P. W. (1990). Computer simulations. *PSA, 2,* 497–506.
Humphreys, P. W. (2004). *Extending ourselves: Computational science, empiricism, and scientific method*. Oxford: Oxford University Press.
Humphreys, P. W. (2009). The Philosophical novelty of computer simulation methods. *Synthese, 169*(3), 615–626.
Humphreys, P. W. (2013). *What are data about?*. Newcastle upon Tyne: Cambridge Scholars Publishing.
Ionescu, T. B. (2018). Simulation, epistemic opacity, and 'envirotechnical ignorance' in nuclear crisis. *Minds and Machines, 7317*(467), 753.
Knuth, D. E. (1973). *The art of computer programming*. Boston: Addison-Wesley.
Krohs, U. (2008). How digital computer simulations explain real-world processes. *International Studies in the Philosophy of Science, 22*(3), 277–292.
Ladyman, J. (2016). Structural realism. In: Zalta, E. N., ed., *The stanford encyclopedia of philosophy (Winter edition)*.
Lenhard, J. (2014). Disciplines, models, and computers: The path to computational quantum chemistry. *Studies in History and Philosophy of Science, 48,* 89–96.
Lenhard, J. (2019). *Calculated surprises*. Oxford: Oxford University Press.
Lenhard, J., & Carrier, M. (2017). *Mathematics as a tool tracing new roles of mathematics in the sciences*. Berlin: Springer.
Mäki, U. (2009). MISSing the world. models as isolations and credible surrogate systems. *Erkenntnis, 70*(1), 29–43.
Miłkowski, M. (2016). A mechanistic account of computational explanation in cognitive science and computational neuroscience. *Computing and philosophy* (pp. 191–205). Berlin: Springer.
Morgan, M. S. (2003). Experiments without Material Intervention. In H. Radder (Ed.), *The philosophy of scientific experimentation* (pp. 216–235). Pittsburgh: University of Pittsburgh Press.
Morgan, M. S., & Morrison, M. (Eds.). (1999). *Models as mediators: Perspectives on natural and social sciences*. Cambridge: Cambridge University Press.
Morrison, M. (2015). *Reconstructing Reality. Models, mathematics, and simulations*. Oxford: Oxford University Press.
Parker, W. S. (2009). Does matter really matters? Computer Simulations, experiments, and Materiality. *Synthese, 169*(3), 483–496.
Peck, S. L. (2012). Agent-based models as fictive instantiations of ecological processes. *Philosophy and Theory in Biology, 4*(20170609), 1–2.
Perini, L. (2005). The truth in pictures. *Philosophy of Science, 72*(1), 262–285.
Pfleeger, S. L., & Atlee, J. M. (2010). *Software engineering: Theory and practice*. Upper Saddle River: Prentice Hall.
Primiero, G. (2014). On the ontology of the computing process and the epistemology of the computed. *Philosophy and Technolgoy, 27*(3), 485–489.
Rapaport, W. J. (1999). Implementation is semantic interpretation. *The Monist, 82*(1), 109–130.
Rapaport, W. J. (2005). Implementation is semantic interpretation: Further thoughts. *Journal of Experimental & Theoretical Artificial Intelligence, 17*(4), 385–417.
Rapaport, W. J. (2019). Syntax, semantics, and computer programs. *Philosophy & Technology, 29,* 35–59. https://doi.org/10.1007/s13347-019-00365-8.
Rescorla, M. (2013). Against structuralist theories of computational implementation. *The British Journal for the Philosophy of Science, 64*(4), 681–707.
Rohrlich, F. (1990). Computer simulation in the physical sciences. *PSA, 2,* 507–518.
Scheutz, M. (2001). Computational versus causal complexity. *Minds and Machines, 11,* 544–566. https://doi.org/10.1023/A:1011855915651.
Suárez, M., & Cartwright, N. (2008). Theories: tools versus models. *Studies in History and Philosophy of Science, 39*(1), 62–81.
Symons, J., & Alvarado, R. (2019). Epistemic entitlements and the practice of computer simulation. *Minds and Machines, 48*(4), 729.
Turner, R. (2007). Computable models. *Journal of Logic and Computation, 18*(2), 283–318.
Turner, R. (2018). *Computational artifacts towards a philosophy of computer science*. Berlin: Springer.
Varenne, F. (2018). *From Models to Simulations*. Abingdon: Routledge.

Weirich, P. (2011). The Explanatory power of models and simulations: A philosophical exploration. *Simulation & Gaming, 42*(2), 155–176.
Weisberg, M. (2007). Who is a modeler? *The British Journal for the Philosophy of Science, 58*(2), 207–233.
Winsberg, E. (1999). Sanctioning Models: The epistemology of simulation. *Science in Context, 12,* 275–292.
Winsberg, E. (2001). Simulations, models, and theories: Complex physical systems and their representations. *Philosophy of Science, 68,* S442.
Winsberg, E. (2010). *Science in the age of computer simulation*. Chicago: University of Chicago Press.
Woolfson, M. M., & Pert, G. J. (1999a). *An Introduction to computer simulations*. Oxford: Oxford University Press.
Woolfson, M. M. & Pert, G. J. (1999b). SATELLIT.FOR.