Holism, or the Erosion of Modularity: A Methodological Challenge for Validation

Johannes Lenhard*†

Modularity is a key concept in building and evaluating complex simulation models. My main claim is that in simulation modeling modularity tends to degenerate for reasons inherent to simulation methodology. The argument will proceed by analyzing the techniques of parameterization, tuning, and kludging. They are—to a certain extent—inevitable when building complex simulation models but erode modularity. As a result, the common account of validating simulations faces a major problem, namely, a problem of holism. In the conclusion, I will ask to what extent holism sets limits to validation.

1. Introduction. Modularity is used in many guises and is not a particularly philosophical notion. It stands for first breaking down complicated tasks into small and well-defined subtasks and then reassembling the original global task following a well-defined series of steps. Modularity features prominently in the context of complex design, planning, and building—from architecture to software engineering.

Holism is a term more common in philosophy. The Stanford Encyclopedia, for instance, includes about 100 entries that mention holism of methodological, metaphysical, relational, or other kinds. Holism generically states that the whole is greater than the sum of its parts, meaning that the parts are in such intimate interconnection that they cannot exist independently of the whole or cannot be understood without reference to the whole. Especially Quine has made the concept popular, not only in philosophy of language but

*To contact the author, please write to: Johannes Lenhard Bielefeld University, Department of Philosophy, Universitaetsstr. 25 33615 Bielefeld, Germany; e-mail: Johannes .lenhard@uni-bielefeld.de.

†I would like to thank Nic Fillion, Rob Moir, and the reviewers of this journal who helped to improve this article. The work was funded by German Research Foundation (DFG) priority program (SPP) 1689.

Philosophy of Science, 85 (December 2018) pp. 832–844. 0031-8248/2018/8505-0008$10.00

also in philosophy of science, where one speaks of the so-called Duhem-Quine thesis. This thesis is based on the insight that one cannot test a single hypothesis in isolation but that any such test depends on "auxiliary" theories or hypotheses, for example, how the measurement instruments work. Thus, any test addresses a whole ensemble of theories and hypotheses.

Lenhard and Winsberg (2010) have discussed the problem of confirmation holism in the context of validating complex climate models. They argued that "due to interactivity, modularity does not break down a complex system into separately manageable pieces" (256). In a sense, I want to pick up on this work but place the thesis in a much more general context. Namely, I want to point to a dilemma that is built on the tension between modularity and holism and that occurs quite generally in simulation modeling. The potential philosophical novelty of simulation is debated controversially in philosophy of science (e.g., Humphreys 2009 vs. Frigg and Reiss 2009). The latter authors deny novelty but concede issues of holism might be an exception. My article confirms that holism should be a key concept when reasoning about simulation. (I see more reasons speaking in favor of philosophical novelty, though.)

My main claim is as follows: According to the rational picture of design, modularity is a key concept in building and evaluating complex systems, including complex models. In simulation modeling, however, modularity erodes for reasons inherent to simulation methodology. Moreover, the very condition for successful simulation undermines the most basic pillar of rational design. The resulting problem for validating models is one of (confirmation) holism.

Section 2 discusses modularity and its central role for the so-called rational picture of design. Herbert Simon's highly influential parable of the watchmakers will feature prominently. It paradigmatically captures complex systems as a sort of large clockwork mechanism. I want to emphasize the disanalogy to how simulation modeling works. Simulation is based on an iterative and exploratory mode of modeling that erodes modularity.

I will present two arguments that support the erosion claim, one from parameterization and tuning (sec. 3), the other from klu(d)ging (sec. 4). Both are, in practice, part and parcel of simulation modeling, and both erode modularity. The article will conclude by drawing lessons about the limits of validation (sec. 5). Most accounts of validation either explicitly or implicitly require modularity and are incompatible with holism. In contrast, the exploratory and iterative mode of modeling restricts validation, at least to a certain extent, to testing (global) predictions. The arguments establish the erosion of modularity as a tendency, not an inescapable fact.

2. The Rational Picture.

The design of complex systems has a long tradition in architecture and engineering. Nonetheless, it has not been covered much in the literature. One of the exceptions is Frederic Brooks, software and computer expert (and former manager at IBM) as well as hobby architect. Brooks (2010) describes the widely significant rational model of design that is often adopted in practice. According to the rational picture, the design process starts with an overview of all options at hand. This agrees with Herbert Simon, for whom the theory of design is the general theory of search through large combinatorial spaces (Simon 1969, 54). The rational model then presupposes a utility function and a design tree, which spans the space of possible designs. Brooks rightly points out that these elements are normally not at hand. Nevertheless, design is conceived as a systematic step-by-step process. In their voluminous work, Pahl and Beitz (1984, plus revised editions 1996, 2007) try to detail these steps in their rational order. For the current discussion, it suffices to note that some hierarchical order is a key element of the rational picture of design. Such order presumes modularity.

Let me illustrate this point. Consider first a simple brick wall. It consists of a multitude of modules, each with certain form and static properties. These are combined into potentially very large structures. It is a strikingly simple example, because all modules (bricks) are similar. A related example is a building like the auxiliary building of Bielefeld University in front of my former office that is put together from container modules, some of which work as office space, others as restrooms, and so on.

These examples illustrate how deeply ingrained modularity is in our way of building (larger) objects. This applies also to the design of complex (software) systems. Some complex overall task is split up into modules that can be tackled independently and by different teams. The hierarchical structure ensures the modules can be integrated to make up the envisioned complex system. Modularity not only plays a key role when designing and building complex systems, it also is of crucial importance when investigating and evaluating built systems. Validation is usually conceived in the very same modular structure: independently validated modules are put together in a controlled way for ensuring the bigger system is also valid. The standard account of how computational models are verified and validated gives very rigorous guidelines that are all based on the systematic realization of modularity (Oberkampf and Roy 2010; see also Fillion 2017). In short, modularity is key for designing as well as for validating complex systems.

This observation is paradigmatically expressed in Simon's parable of the two watchmakers. It can be found in Simon's 1962 paper "The Architecture of Complexity" that became a chapter in his immensely influential *The Sciences of the Artificial* (Simon 1969). There, Simon investigates the structure of complex systems. The stable structures, according to Simon, are the hierarchical ones. He expressed his idea by recounting the parable of two watchmakers named Hora and Tempus (90–92). Agre describes the setting in the following words: "According to this story, both watchmakers were equally skilled, but only one of them, Hora, prospered. The difference between them lay in the design of their watches. Each design involved 1000 elementary components, but the similarity ended there. Tempus' watches were not hierarchical; they were assembled one component at a time. Hora's watches, by contrast, were organized into hierarchical subassemblies whose 'span' was ten. He would combine ten elementary components into small subassemblies, and then he would combine ten subassemblies into larger subassemblies, and these in turn could be combined to make a complete watch" (2003, 416).

Because Hora takes additional steps and builds modules, Tempus's watches need less time for assembly. However, it was Tempus's business that did not thrive, because of an additional condition not yet mentioned, namely, some kind of noise. From time to time, the telephone rings, and whenever one of the watchmakers answers the call, all cogwheels and little screws fall apart, and that watchmaker has to recommence the assembly. Whereas Tempus had to start from scratch, Hora could keep all the finished modules and work from there. In the presence of noise, so the lesson goes, the modular strategy is superior by far. For Simon (and Agre), only modular structures can make up well-engineered and functioning systems. Presumably, Simon was aware that the clockwork picture is limited, and he even mentioned that complicated interactions could lead to a sort of pragmatic holism.[1] Do we have to look at simulation models as a sort of gigantic clockworks? In what follows, I argue that this viewpoint is seriously misleading. Simulation models differ from watches in important ways, and I want to focus on this disanalogy.[2]

3. Erosion of Modularity 1: Parameterization and Tuning. I want to discuss two separate although related arguments, the first from parameterization and tuning and (in the next section) the second from kluging. Both are, for different reasons, part and parcel of simulation modeling, and both make the modularity of models erode. Parameterization and tuning are key elements of simulation modeling that stretch the realm of tractable subject matter far beyond what is covered by theory alone. Therefore, they are indispensable in most simulation models.

Before I start discussing an example, let me add a few words about terminology. There are different expressions that specify what is done with parameters. The four most common ones are (in alphabetical order): adaptation, adjustment, calibration, and tuning. These notions describe very similar activities but also valuate differently what parameters are good for. Calibration is commonly used in the context of preparing an instrument, like a one-off

1. This kind of holism, hence, can occur even when modules are "independently validated," since these modules, when connected, could interact with each other in unpredicted ways. This is a strictly weaker form of holism than the one I am going to discuss.

2. There are several disanalogies. One I am not discussing is that clockworks lack multifunctionality.

calibration of a scale. Tuning has a more pejorative tone, for example, achieving a fit with artificial measures or fit to a particular case. Adaptation and adjustment have more neutral meanings.

Atmospheric circulation is a typical example. It is modeled on the basis of accepted theory (fluid dynamics, thermodynamics, motion) on a grand scale. Climate scientists call this the "dynamic core" of their models, and there is more or less consensus over this part. What climate scientists call 'the physics' means those processes that are not completely specified from the dynamic core. These processes include convection schemes or cloud dynamics, among others.

Often, such processes are not known in full detail, and some aspects (at least) depend on what happens on a subgrid scale. The dynamics of clouds, for instance, depend on a staggering span of very small (molecular) scales and much larger scales of many kilometers. Hence, even if the laws that guide these processes were known, they could not be treated explicitly in the simulation model because this would require a superfine grid that is computationally not feasible on a global scale. Modeling 'the physics' has to bring in parameterization schemes.[3]

For example, how do cloud processes work? Rather than trying to investigate and model all microphysical details, like how exactly water vapor is entrained into air (building clouds), scientists use a parameter, or a scheme of parameters, that controls moisture uptake so that known observations are (roughly) met. Often, such parameters do not have a direct physical interpretation, nor do they need one, like when a parameter stands for a mixture of (partly unknown) processes not resolved in the model. One then speaks of using an 'effective' parameter. The important property of working parameterizations is not accuracy in representation on the small scale.[4] Rather, the parameterization scheme has to be flexible, so that the parameters of such a scheme can be changed in a way that makes the overall model match some known data or reference points. Representation then is relevant on a much larger scale.

From this rather straightforward observation follows an important fact. A parameterization, including assignments of parameter values, makes sense only in the context of the larger model. Observational data are not compared to the parameterization in isolation. The *Fourth Assessment Report of the Intergovernmental Panel on Climate Change* acknowledges the point that "pa-

3. Parameterization schemes and their status in between theoretical and instrumental aspects are discussed in the literature on climate simulation models; e.g., Smith (2002), Stainforth et al. (2007), Gramelsberger and Feichter (2011), or Parker (2014).

4. Of course, physically well-motivated, 'realistic' parameterizations increase the credibility of a model. But if they are not available, more pragmatic parameterizations offer a way out.

rameterizations have to be understood in the context of their host models" (Solomon et al. 2007, 8.2.1.3).

Adapting parameters, however, endangers the modular structure. One particular parameter value (controlling moisture uptake) is judged according to the results it yields for the overall behavior (like global energy balance). In other words, tuning is a local activity that is oriented at global behavior. Furthermore, tuning parameters is not only oriented at the global model performance, it tends to blur how the local behavior is brought about. This is because no model will be perfect, since it contains technical errors, works with insufficient knowledge, and so on—which is just the normal case in scientific practice. Now, tuning a parameter according to the overall behavior of the model then means that the errors, gaps, and bugs compensate against each other (if in an opaque way). Mauritsen et al. (2012) have pointed this out in their pioneering article on tuning in climate modeling.

In climate models, cloud parameterizations play an important role, because they influence key statistics of the climate and, at the same time, cover major (remaining) uncertainties about how an adequate model should look (see, e.g., Stevens and Bony 2013). Building parameterization schemes that work with "physical" parameters (that have a straightforward physical interpretation) is still a big challenge (Hourdin et al. 2013). Over the process of adjusting the (physical and nonphysical) parameters, these schemes become inevitably convoluted. The simulation then is based on the balance of these parameters in the context of the overall model (including other parameterizations). I leave aside that models of atmosphere and oceans get coupled, which arguably aggravates the problem.

Tuning is part and parcel of simulation modeling methodology. It poses great challenges, like finding a good parameterization scheme for cloud dynamics, which is a recent area of intense research in meteorology. But when is a parameterization scheme a good one? On the one side, a scheme is sound when it is theoretically well motivated; on the other side, the key property of a parameterization scheme is its adaptability. Both criteria do not point in the same direction. There is, therefore, no optimum; finding a balance is still considered an art. I suspect that the widespread reluctance against publishing about practices of adjusting parameters comes from reservations against aspects that call for experience and art rather than theory and rigor.

I want to highlight that nothing in the above argumentation is specific to climate modeling. The point holds for simulation modeling of some complexity quite generally. For lack of space, I can mention another example only briefly. Adjusting parameters is also occurring in thermodynamics, an area of physics with a very sound theoretical reputation. So-called equations of state find wide applications also in chemical engineering. They are typically very specific for certain substances and require extensive adjustment of parameters, as Hasse and Lenhard (2017) have described and analyzed. They argue that being able to process specific adjustment strategies based on parameterization schemes is a crucial success condition. Typically, up-to-date equations of state contain 20 or more adjustable parameters of which only a minority has physical meaning. The argumentation above applies mutatis mutandis. Simulation methods have made thermodynamics applicable in many areas of practical relevance, exactly because equations of state can be adapted to particular cases of interest.

Let me take stock regarding the first argument for the erosion of modularity. Tuning, or adjusting, parameters is not merely an ad hoc procedure; rather, it is a pivotal component for simulation modeling. Adjusting parameters convolutes heterogeneous parts that do not have a common theoretical basis. Tuning proceeds holistically on the basis of global model behavior. How particular parts function often remains opaque. By interweaving local and global considerations, and by convoluting the interdependence of various parameter choices, tuning erodes modularity.

4. Erosion of Modularity 2: Kluging. The second argument regarding the erosion of modularity approaches the issue from a different angle, namely, from a specific practice in developing software known as kluging (also spelled kludging).[5] "Kluge" is a term from colloquial language that was adopted in computer slang. In the words of Wikipedia, a kluge is "a workaround or quick-and-dirty solution that is clumsy, inelegant, difficult to extend and hard to maintain, yet an effective and quick solution to a problem" (https://en.wikipedia.org/wiki/Kludge).

Andy Clark, in a similar manner, stresses the important role played by kluges in computer modeling. For him, a kluge is "an inelegant, 'botched together' piece of program; something functional but somehow messy and unsatisfying"; it is—Clark refers to Sloman—"a piece of program or machinery which works up to a point but is very complex, unprincipled in its design, ill-understood, hard to prove complete or sound and therefore having unknown limitations, and hard to maintain or extend" (Clark 1987, 278).

Kluges proceeded from programmers' language into the body of philosophy, guided by scholars such us Clark (1987) and Wimsatt (2007) who are inspired both by computer modeling and evolutionary theory. The important point in the current context is that kluges typically function for a whole system, that is, for the performance of the entire simulation model. In contrast, they have no meaning in relation to the submodels and modules: "what is a kludge considered as an item designed to fulfill a certain role in a large system, may be no kludge at all when viewed as an item designed to fulfill a somewhat different role in a smaller system" (Clark 1987, 279).

5. Both spellings "kluge" and "kludge" are used. There is not even agreement of how to pronounce the word. In a way, that fits to the very concept. I will use "kluge" but will not change the habits of other authors cited with "kludge."

Because kluging stems from colloquial language and is also not viewed as good practice, examples are difficult to find in published scientific literature. This observation notwithstanding, kluging is a widespread phenomenon. Let me give an example that I know from visiting an engineering laboratory. There, researchers (chemical process engineers) are working with simulation models of an absorption column, the large steel structures in which reactions take place under controlled conditions. The scientific details do not matter here, because the point is that the engineers build their model on the basis of a couple of already existing modules, including proprietary software that they integrate into their simulation without having access to the code. Moreover, it is common knowledge in the community that this (unknown) code is of poor quality. Because of programming errors and because of ill-maintained interfaces, using this software package requires modifications on the part of the code outside the package. These modifications are not there for any good theoretical reason, albeit for good practical ones. They make the overall simulation run as expected (in known cases), and they allow working with existing software. Hence, the modifications are typical kluges.

Another relevant and common phenomenon is the increasing importance of 'exception handling', that is, of finding effective repairs when the software or the model performs in unanticipated and undesired ways. In this situation, the software might include a bug that is invisible (does not affect results) most of the time but becomes active under particular conditions. Often, extensive testing is needed to find out about unwanted behavior that occurs in rare and particular situations that are conceived of as 'exceptions'—indicating that researchers do not aim at a major reconstruction but at a local repair to counteract this particular exception.

Presumably all readers who ever contributed to a large software program are familiar with experiences of this kind. It is commonly accepted that the more comprehensive a piece of software gets, the more energy new releases will require in order to handle exceptions. Operating systems of computers, for example, often receive weekly patches (i.e., repair updates). Many scientists who work with simulations face a similar situation, although not obviously so.

Here is an example by personal communication. I have attended meetings of a group of atmospheric modelers. They had released a new generation of their atmospheric circulation model. Many processes had been refined, new knowledge added, and so on. The new simulation model worked fine on several evaluative criteria. When they connected their new model to the already existing chemistry module, however, the overall performance deteriorated in a striking manner. There had to be one or several important mistakes in the code, the group figured. When they found out the reason, they were surprised. There existed parts in the chemistry module that had been introduced to balance out shortcomings in a much older version of the atmospheric circulation model. In other words, these parts were kluges in an older version of the model but had not been removed and were overlooked because they did not affect the performance of the newer model. Only later, with the newest (recent) atmospheric model, the outdated kluges produced unwanted model behavior and thus became manifest.

Why should these examples be seen as typical instances and not as exceptions? Because they arise from practical circumstances of developing software, and this is a core part of simulation modeling. Software engineering is a field that was envisioned as the 'professional' answer to the increasing complexity of software. And I frankly admit that there are well-articulated concepts that would, in principle, ensure software is clearly written, aptly modularized, well maintained, and superbly documented. However, the problem is that science in principle differs from science in practice.

In practice, strong and constant forces drive software developers to resort to kluges. Providing a comprehensive analysis would require space not available here. Instead, I refer to the account of Foote and Yoder, prominent leaders in the field of software development, who give an ironic and funny account of how attempts to maintain a rationally designed software architecture constantly fail in practice.

> While much attention has been focused on high-level software architectural patterns, what is, in effect, the de-facto standard software architecture is seldom discussed. This paper examines this most frequently deployed of software architectures: the BIG BALL OF MUD. A big ball of mud is a casually, even haphazardly, structured system. Its organization, if one can call it that, is dictated more by expediency than design. Yet, its enduring popularity cannot merely be indicative of a general disregard for architecture. . . . Even systems with well-defined architectures are prone to structural erosion. The relentless onslaught of changing requirements that any successful system attracts can gradually undermine its structure. Systems that were once tidy become overgrown as piecemeal growth gradually allows elements of the system to sprawl in an uncontrolled fashion. (Foote and Yoder 2000, 2–3)

I would like to repeat the statement from above that there is no necessity for the corruption of modularity and rational architecture. Again, this is a question of science in practice versus science in principle. "A sustained commitment to refactoring can keep a system from subsiding into a big ball of mud," Foote and Yoder admit (2000, 3).

Robert Martin, pioneering the "clean code" school, proposes keeping code clean in the sense of not letting the first kluge slip in. And surely there is no principled reason why one should not be able to avoid this. However, even Martin accepts the diagnosis of current practice. Similarly, Richard Gabriel (1996), another guru of software engineering, makes the analogy to housing architecture and Alexander's concept of "habitability" that aims to integrate modularity and piecemeal growth into one "organic order." In any case, when describing the starting point, he more or less duplicates what I mention above from Foote and Yoder.

Finally, I want to point out that the nature of kluging resembles what is discussed in philosophy of science under the heading of opacity (like in Humphreys 2009). Highly kluged software becomes opaque. One can hardly disentangle the various reasons that led to particular pieces of code, because kluges are meaningful only in the particular context at one particular time. In this important sense, simulation models are historical objects. They carry around—and depend on—their history of modifications. There are interesting analogies with biological evolution that became a topic when Winograd and Flores, for instance, come to a conclusion that also holds in our context here: "each detail may be the result of an evolved compromise between many conflicting demands. At times, the only explanation for the system's current form may be the appeal to this history of modification" (1991, 94).

Two conditions work in favor of kluging. First, the exchange of software parts is more or less motivated by flexibility and economic requirements. This thrives on networked infrastructure. Second, iterations and modifications are easy and cheap to perform. Because of the unprincipled nature of kluges, their construction requires repeated testing of whether they actually work in the factual circumstances. Hence, kluges fit the exploratory and iterative mode of modeling that characterizes simulations. Furthermore, layered kluges solidify themselves. They make code hard or impossible to understand, modifying pieces that are individually hard to understand will normally lead to a new layer of kluges, and so on. Thus, kluging erodes modularity. This is another reason why simulation modeling systematically undermines modularity.

5. The Limits of Validation. What does the erosion of modularity mean for the validation of computer simulations? In the context of simulation models the community speaks of verification and validation, or V&V. Whereas verification checks the model internally, validation checks whether the model adequately represents the target system. A standard definition states that "verification [is] the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model." In contrast, validation is defined as "the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model" (Oberkampf and Trucano 2000, 3).

Because of the increasing usage and growing complexity of simulations, the issue of V&V is itself an expanding field in simulation literature. One example is the voluminous monograph by Oberkampf and Roy (2010) that meticulously defines and discusses the various steps to be included in V&V procedures. A first move in this analysis is to separate model form from model parameters. Each parameter then belongs to a particular type of parameter that determines which specific steps in V&V are required. Oberkampf and Roy give the following list of model parameter types: “measurable properties of the system or the surroundings,” “physical modeling parameters,” “ad hoc model parameters,” “numerical algorithm parameters,” “decision parameters,” and “uncertainty modeling parameters” (623). The adjustable parameters I discussed above do not appear in this list. This is not simply a matter of terminology. My point is that these parameters belong to model form and to model performance at the same time. The performance part of my claim is clear, since adjusting these parameters is oriented at model performance. The crucial point is that these parameters also belong to the model form, because without assignment of parameters neither the question about representational adequacy nor the question about behavioral fit can be addressed. A cloud parameterization scheme makes sense only if its parameter values have been assigned already, and the same holds for an equation of state. Before the process of adjustment, the mere form of the scheme can hardly be called adequate or inadequate. I admit that my claim does not hold for extreme cases like a purely physically motivated parameterization scheme whose adequacy can be judged independently of the model behavior. In simulation models, as I have shown, (predictive) success and adjustment normally are entangled. The adjustable parameters I discussed above are of a type that evades the V&V fencing.

The separation of verification and validation thus cannot be fully maintained in practice, that is, in the cases in which simulation models employ adjustable parameters. It is not possible to first verify that a simulation model is ‘right’ before tackling the ‘external’ question whether it is the right model. Performance tests, hence, become the main handle for confirmation. This is a version of confirmation holism that points toward the limits of analysis. This does not lead to a conceptual breakdown of V&V. Rather, holism comes in degrees,[6] and it is a pernicious tendency (not necessity) that undermines the verification-validation divide.[7]

In the current article, I have questioned the rational picture of design that is so crucially based on modularity. My criticism works, if you want, from

6. I thank Rob Moir for pointing this out to me.

7. My conclusion about the inseparability of verification and validation is in good agreement with Winsberg’s (2010) more specialized claim in which he argues about model versions that evolve because of changing parameterizations, which has been criticized by Morrison (2015). As far as I can see, her arguments do not apply to the case made in this article, which rests on a tendency toward holism rather than a complete conceptual breakdown.

'within'. It is the very methodology of simulation modeling, and how it works in practice, that erodes modularity and therefore challenges the rational picture.

REFERENCES

Agre, Philip E. 2003. "Hierarchy and History in Simon's 'Architecture of Complexity.' " *Journal of the Learning Sciences* 3:413–26.

Brooks, Frederick P. 2010. *The Design of Design*. Reading, MA: Addison-Wesley.

Clark, Andy. 1987. "The Kludge in the Machine." *Mind and Language* 2 (4): 277–300.

Fillion, Nicolas. 2017. "The Vindication of Computer Simulations." In *Mathematics as a Tool*, ed. Johannes Lenhard and Martin Carrier, 137–56. Boston Studies in History and Philosophy of Science 327. Cham: Springer.

Foote, Brian, and Joseph Yoder. 2000. "Big Ball of Mud." In *Pattern Languages of Program Design 4*, ed. Neil Harrison, Brian Foote, and Hans Rohnert. Boston: Addison-Wesley. http://laputan.org/pub/foote/mud.pdf.

Frigg, Roman, and Julian Reiss. 2009. "The Philosophy of Simulation: Hot New Issues or Same Old Stew?" *Synthese* 169 (3): 593–613.

Gabriel, Richard P. 1996. *Patterns of Software: Tales from the Software Community*. New York: Oxford University Press.

Gramelsberger, Gabriele, and Johann Feichter, eds. 2011. *Climate Change and Policy: The Calculability of Climate Change and the Challenge of Uncertainty*. Heidelberg: Springer.

Hasse, Hans, and Johannes Lenhard. 2017. "On the Role of Adjustable Parameters." In *Mathematics as a Tool*, ed. Johannes Lenhard and Martin Carrier, 93–116. Boston Studies in History and Philosophy of Science 327. Cham: Springer.

Hourdin, Frédéric, et al. 2013. "LMDZ5B: The Atmospheric Component of The IPSL Climate Model with Revisited Parameterizations for Clouds and Convection." *Climate Dynamics* 40:2193–222.

Humphreys, Paul. 2009. "The Philosophical Novelty of Computer Simulation Methods." *Synthese* 169 (3): 615–26.

Lenhard, Johannes, and Eric Winsberg. 2010. "Holism, Entrenchment, and the Future of Climate Model Pluralism." *Studies in History and Philosophy of Modern Physics* 41:253–62.

Mauritsen, Thorsten, et al. 2012. "Tuning the Climate of a Global Model." *Journal of Advances in Modeling Earth Systems* 4 (3): M00A01. doi:10.1029/2012MS000154.

Morrison, Margaret. 2015. *Reconstructing Reality: Models, Mathematics, and Simulations*. New York: Oxford University Press.

Oberkampf, William L., and Christopher J. Roy. 2010. *Verification and Validation in Scientific Computing*. Cambridge: Cambridge University Press.

Oberkampf, William L., and Timothy G. Trucano. 2000. *Validation Methodology in Computational Fluid Dynamics*. Washington, DC: US Department of Energy.

Pahl, Gerhard, and Wolfgang Beitz. 1984. *Engineering Design: A Systematic Approach*. Berlin: Springer.

Parker, Wendy. 2014. "Values and Uncertainties in Climate Prediction, Revisited." *Studies in History and Philosophy of Science* 46:24–30.

Simon, Herbert A. 1969. *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

Smith, Leonard A. 2002. "What Might We Learn from Climate Forecasts?" *Proceedings of the National Academy of Sciences of the USA* 4 (99): 2487–92.

Solomon, Susan, Dahe Qin, Martin Manning, Z. Chen, Melinda Marquis, Kristen B. Averyt, Melinda M. B. Tignor, and Henry L. Miller, eds. 2007. *Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge: Cambridge University Press.

Stainforth, David A., Thomas E. Downing, Richard Washington, Anna Lopez, and Mark New. 2007. "Issues in the Interpretation of Climate Model Ensembles to Inform Decisions." *Philosophical Transactions of the Royal Society* A 365 (1857): 2145–61.

Stevens, Björn, and Sandrine Bony. 2013. “What Are Climate Models Missing?” *Science* 340: 1053–54.

Wimsatt, William C. 2007. *Re-engineering Philosophy for Limited Beings: Piecewise Approximations to Reality*. Cambridge, MA: Harvard University Press.

Winograd, Terry, and Fernando Flores. 1991. *Understanding Computers and Cognition*. Reading, MA: Addison-Wesley.

Winsberg, Eric. 2010. *Science in the Age of Computer Simulation*. Chicago: University of Chicago Press.