Holism, or the Erosion of Modularity – a Methodological Challenge for Validation

Draft to be presented at PSA 2016

Johannes Lenhard, Bielefeld University

abstract

Modularity is a key concept in building and evaluating complex simulation models. My main claim is that in simulation modeling modularity degenerates for systematic methodological reasons. Consequently, it is hard, if not impossible, to accessing how representational (inner mathematical) structure and dynamical properties of a model are related. The resulting problem for validating models is one of holism.

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 corrode modularity. As a result, the common account of validating simulations faces a major problem and testing the dynamical behavior of simulation models becomes all the more important. Finally, I will ask in what circumstances this might be sufficient for model validation.

1. Introduction

For the moment, imagine a scene at a car racing track. The air smells after gasoline. The pilot of the F1 racing car has just steered into his box and is peeling himself out of the straight cockpit. He puts off his helmet, shakes his sweaty hair, and then his eyes make contact to the technical director with a mixture of anger, despair, and helplessness. The engine had not worked as it should, and for a known reason: the software. However, the team had not been successful in attributing the miserable performance to a particular parameter setting. The machine and the software interacted in unforeseen and intricate ways. This explains the exchange of glances between pilot and technical director. The software's internal interactions and interfaces proved to be so complicated that the team had not been able to localize an error or a bug, rather remained

suspicious that some complex interaction of seemingly innocent assumptions or parameter settings was leading to the insufficient performance.

The story happened in fact[1] and it is remarkable since it displays how invasive computational modeling is into areas that smell most analogous. I reported this short piece for another reason, however, namely because the situation is typical for complex computational and simulation models. Validation procedures, while counting on modularity, run against a problem of holism.

Both concepts, modularity and holism, are notions at the fringe of philosophical terminology. Modularity is used in many guises and is not a particularly philosophical notion. It features prominently in the context of complex design, planning, and building – from architecture to software. Modularity stands for first breaking down complicated tasks into small and well-defined sub-tasks and then re-assembling the original global task with a well-defined series of steps. It can be argued that modularity is the key pillar on which various rational treatments of complexity rest – from architecture to software engineering.

Holism is a philosophical term to a somewhat higher degree and is covered in recent compendia. The Stanford Encyclopedia, for instance, includes (sub-)entries on methodological, metaphysical, relational, or meaning holism. Holism generically states that the whole is greater than the sum of its parts, meaning that the parts of a whole are in intimate interconnection, such that they cannot exist independently of the whole, or cannot be understood without reference to the whole. Especially W. V. O. Quine has made the concept popular, not only in philosophy of language, but 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." (2010, 256) In a sense, I want to pick up on this work, but put the thesis into a much more general context, i.e. pointing

[1] In spring 2014, the Red Bull team experienced a crisis due to recalcitrant problems with the Renault engine, due to a partial software update.

out a dilemma that is built on the tension between modularity and holism and that occurs quite generally in simulation modeling. The potential philosophical novelty is debated controversially in philosophy of science, for instance Humphreys (2009) vs. Frigg and Reiss (2009). The latter authors deny novelty, but concede issues of holism might be an exception. My paper confirms that holism is a key concept when reasoning about simulation. (I see more reasons for philosophical novelty, though.)

My main claim is the following: According to the rational picture of design, modularity is a key concept in building and evaluating complex models. In simulation modeling, however, modularity erodes for systematic methodological reasons. Moreover, the very condition for success of simulation undermines the most basic pillar of rational design. Thus 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. This perspective suggests the computer would enlarge the tractability of complex systems due to its vast capacity for handling (algorithmic) mechanisms. Complex simulations then would appear as the electronic incarnation of a gigantic assembly of cogwheels. This viewpoint is misleading, I will argue. Instead, I want to emphasize the dis-analogy to how simulation models work. The methodology of building complex simulation models thwarts modularity in systematic ways. Simulation is based on an iterative and exploratory mode of modeling that leads to a sort of *holism that erodes modularity*.

I will present two arguments for the erosion claim, one from parameterization and tuning (section 3), the other from klu(d)ging (section 4). Both are, in practice, part-and-parcel of simulation modeling and both make modularity erode. The paper will conclude by drawing lessons about the limits of validation (section 5). Most accounts of validation require, if often not explicitly, 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) predictive virtues. This observation shakes the rational (clockwork) picture of design and of the computer.

2. The rational picture

The design of complex systems has a long tradition in architecture and engineering. At the same time, it has not been much covered in literature, because design was conceived as a matter for experienced craftsmanship rather than analytical investigation. The work of Pahl and Beitz (1984, plus revised editions 1996, 2007) gives a relatively recent account of design in engineering. A second, related source for reasoning about design is the design of complex computer systems. Here, one can find more explicit accounts, since the computer led to complex systems much faster than any tradition of craftsmanship could grow. A widely read example is Herbert Simon's "Sciences of the Artificial" (1969). Still up to today, techniques of high-level languages, object-oriented programming, etc. make the practice of design change on a fast scale.

One original contributor to this discussion is Frederic Brooks, software and computer expert (and former manager at IBM) and also hobby architect. In his 2010 monograph "The Design of Design", he describes the rational model of design that is widely significant, though it is much more often adopted in practice than explicitly formulated in theoretical literature. The rational picture starts with assuming an overview of all options at hand. According to Simon, for instance, 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 are spanning the space of possible designs. Brooks rightly points out that these are normally not at hand. Nevertheless, design is conceived as a systematic step-by-step process. Pahl and Beitz aim at detailing these steps in their rational order. Also, Simon presupposes the rational model, arguably motivated by making design feasible for artificial intelligence (see Brooks 2010, 16). Wynston Royce, to give another example, introduced the "waterfall model" for software design (1970). Royce was writing about managing the development of large software systems and the waterfall model consisted in following a hierarchy ("downward"), admitting to iterate steps on one layer, but not with much earlier ("upward") phases of the design process. Although Royce actually saw the waterfall model as a straw man, it was cited positively as paradigm of software development (cf. Brooks on this point).

Some hierarchical order is a key element of the rational picture of design and 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 more complicated, though closely related, example is the one depicted in figure 1 where an auxiliary building of Bielefeld University is put together from container modules.

Figure 1: A part of Bielefeld University is built from container modules.

These examples illustrate how deeply ingrained modularity is in our way of building (larger) objects. Figure 2 displays a standard picture for designing and developing complex (software) systems.

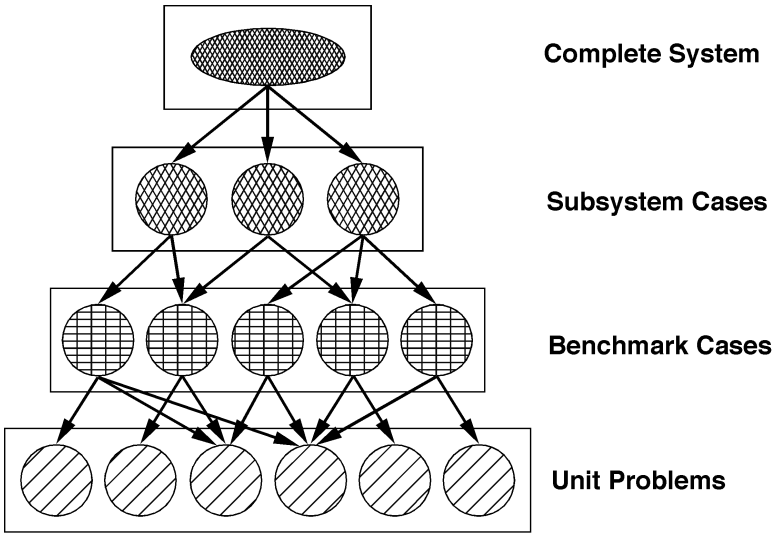

Figure 2: Generic architecture of complex software, from the AIAA Guide for the Verification and Validation of Computational Fluid Dynamics Simulations (1998). Modules of one layer might be used by different modules on a higher layer.

Some complex overall task is split up into modules that can be tackled independently and by different teams. The hierarchical structure shall ensure the modules can be integrated to make up the original complex system. Modularity not only plays a key role when designing and building complex systems, it also is of crucial importance when taking account of the system. Validation is usually conceived in the very same modular structure: independently validated modules are put together in a controlled way for making up a validated bigger system. 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. You find it in Simon's 1962 paper "The Architecture of Complexity" that has become a chapter in his immensely influential "The Sciences of the Artificial" (Simon 1969). There, Simon investigates the structure of complex systems. The stable structures, so Simon argues, are the hierarchical ones. He expressed his idea by telling the parable of the two watchmakers named Hora and Tempus (1969, 90-92). P. Agre describes the setting with 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." (Agre 2003)

Since Hora takes additional steps for building modules, Tempus' watches need less time for assembly. However, it was Tempus' 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 he has to re-start the assembly. While Tempus had to start from scratch, Hora could keep all finished modules and work from there. In the presence of noise, so the lesson goes, the modular

strategy is by far superior. Agre summarizes that modularity, he speaks of the functional role of components, comes out as a necessary element when designing complex systems:

"For working engineers, hierarchy is not mainly a guarantee that subassemblies will remain intact when the phone rings. Rather, hierarchy simplifies the process of design cognitively by allowing the functional role of subassemblies to be articulated in a meaningful way in terms of their contribution to the function of the whole. Hierarchy allows subassemblies to be modified somewhat independently of one another, and it enables them to be assembled into new and potentially unexpected configurations when the need arises. A system whose overall functioning cannot be predicted from the functionality of its components is not generally considered to be well-engineered." (Agre 2003)

Now, the story works with rather particular examples insofar as watches exemplify complicated mechanical devices. The universe as a giant clockwork has been a common metaphor since the seventeenth century. Presumably, Simon was aware the clockwork picture is limited and he even mentioned that complicated interactions could lead to a sort of pragmatic holism.[2] Nonetheless, the hierarchical order is established by the interaction of self-contained modules.

There is an obvious limit to the watchmaker picture, namely systems have to remain manageable by human beings (watchmakers). There are many systems of practical interest that are too complex – from the earth's climate to the aerodynamics of an airfoil. Computer models open up a new path here, since simulation models might contain a wealth of algorithmic steps far beyond what can be conceived in a clockwork picture.[3] From this point of view, the computer appears as a kind of amplifier that helps to revitalize the rational picture. Do we have to look at simulation models as a sort of gigantic clockworks? In the following, I will argue that this viewpoint is seriously misleading. Simulation models are different from watches in important ways and I

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

[3] Charles Babbage had designed his famous „Analytical Engine" as a *mechanistic* computer. Tellingly, it did encounter serious problems exactly because of the mechanical limitations of its construction.

want to focus on the dis-analogy.[4] Finally, we will learn from the investigation of simulation models about our picture of rationality.

3. Erosion of modularity 1: Parameterization and tuning

In stark contrast to the cogwheel picture of the computer, the methodology of simulation modeling erodes modularity in systematic ways. I want to discuss two separate though related aspects, firstly, parameterization and tuning and, secondly, kluging (also called kludging). Both are, for different reasons, part-and-parcel of simulation modeling; and both make modularity of models erode. Let us investigate them in turn and develop two arguments for erosion.

Parameterization and tuning are key elements of simulation modeling that stretch the realm of tractable subject matter much beyond what is covered by theory. Furthermore, simulation models can make predictions even in fields that *are* covered by well-accepted theory only with the help of parameterization and tuning. In this sense, the latter are success conditions for simulations.

Before we start with 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 calibrating a scale one time for using it very often in a reliable way. Tuning has a more pejorative tone, like achieving a fit with artificial measures, or fitting 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 "dynamical core" of their models and there is more or less consensus about this part. Although the employed theory is part of physics, climate scientists mean a different part of their models when they speak of "the physics". It includes all the processes that are not completely specified from the dynamical core. These processes include convection schemes, cloud dynamics, and many more.

[4] There are several dis-analogies. One I am not discussing is that clockworks lack multi-functionality.

The "physics" is where different models differ and the physics is what modeling centers regard as their achievements and try to maintain even if their models change into the next generation.

The physics acts like a specifying supplement to the grand scale dynamics. It is based on modeling assumptions, say which sub-processes are important in convection, what should be resolved in the model and what should be treated via a parameterization scheme. Often, such processes are not known in full detail, and some aspects (at least) depend on what happens on a sub-grid scale. The dynamics of clouds, for instance, depends 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 would be known, they could not be treated explicitly in the simulation model. Modeling the physics has to bring in parameterization schemes.[5]

How does moisture transport, for example, work? Rather than trying to investigate into the molecular details of how water vapor is entrained into air, scientists use a parameter, or a scheme of parameters, that controls moisture uptake so that known observations are 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 processes not resolved in the model. The important property rather is that they make the parameterization scheme flexible, so that the parameters of such a scheme can be changed in a way that makes the properties of the scheme (in terms of climate dynamics) match some known data or reference points.

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 IPCC acknowledges the point that "parameterizations have to be understood in the context of their host models" (Solomon et al. 2007, 8.2.1.3)

The question of whether the parameter value that controls moisture uptake (in our oversimplified example) is adequate can be answered only by examining how the entire parameterization behaves and, moreover, how the parameter value in the parameterization in the larger simulation model behaves. Answering such questions would require, for instance, looking at more global properties like mean cloud cover in tropical regions, or the amount of rain in some area. Briefly

[5] Parameterization schemes and their more or less autonomous status are discussed in the literature, cf. Parker 2013, Smith 2002, or Gramelsberger and Feichter 2011.

stated, parameterization is a key component of climate modeling and tuning is part-and-parcel of parameterization.[6]

It is important to note that tuning one parameter takes the values of other parameters as given, be they parameters from the same scheme, or be they parts of other schemes that are part of the model. A particular parameter value (controlling moisture uptake) is judged according to the results it yields for the overall behavior (like cloud cover). In other words, tuning is a local activity that is oriented at global behavior. Researchers might try to optimize parameter values simultaneously, but for reasons of computational complexity, this is possible only with a rather small subset of all parameters. A related issue is statistical regression methods that might be caught up in a local optimum. In climate modeling, skill and experience remain to be important for tuning (or adjustment).

Furthermore, tuning parameters is not only oriented at the global model performance, it tends to blur the local behavior. This is because every model will be importantly imperfect, since it contains technical errors, works with insufficient knowledge, etc. – 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 get compensated against each other (if in an opaque way). Mauritsen et al. (2012) have pointed this out in their pioneering paper about 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 like. Typically, such a parameterization scheme includes more than two dozens of parameters; most of them do not carry a clear physical interpretation. The simulation then is based on the balance of these parameters in the context of the overall model (including other parameterizations). Over the process of adjusting the parameters, these schemes become inevitably convoluted. I leave aside that models of atmosphere and oceans get coupled, which arguably aggravates the problem.

[6] The studies of so-called perturbed physics ensembles convincingly showed that crucial properties of the simulation models hinge on exactly how parameter values are assigned (Stainforth et al. 2007).

Tuning is inevitable, 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 into 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 rigorous methodology.

I want to maintain that nothing in the above argumentation is particular to climate. Climate modeling is just one example out of many. The point holds for simulation modeling quite generally. Admittedly, climate might be a somewhat peculiar case, because it is placed in a political context where some discussions seem to require that only ingredients of proven physical justification and realistic interpretation are admitted. Arguably, this expectation might motivate using the pejorative term of tuning. This reservation, however, ignores the very methodology of simulation modeling. Adjusting parameters is by no means particular to climate modeling, nor is it confined to areas where knowledge is weak.

Another example will document this. Adjusting parameters is also occurring thermodynamics, an area of physics with very high theoretical reputation. The ideal gas equation is even taught in schools, it is a so-called equation of state (EoS) that describes how pressure and temperature depend on each other. However, actually using thermodynamics requires to work with less idealized equations of state than the ideal gas equation. More complicated 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) describe and analyze. Clearly, being able to process specific adjustment strategies that are based on parameterization schemes is a crucial success condition. Simulation methods have made applicable thermodynamics in many areas of practical relevance, exactly because equations of state are tailored to particular cases of interest via adjusting parameters.

One further example is from quantum chemistry, namely the so-called density functional theory (DFT), a theory developed in the 1960s that won the Nobel prize in 1998. Density functionals

capture the information of the Schroedinger equation, but are much more computationally tractable. However, only many-parameter functionals brought success in chemistry. The more tractable functionals with few parameters worked only in simpler cases of crystallography, but were unable to yield predictions accurate enough to be of chemical interest. Arguably, being able to include and adjust more parameters has been the crucial condition that had to be satisfied before DFT could gain traction in computational quantum chemistry, which happened around 1990. This traction, however, is truly impressive. DFT is by now the most widely used theory in scientific practice, see Lenhard (2014) for a more detailed account of DFT and the development of computational chemistry.

Whereas the adjustment of parameters – to use the more neutral terminology – is pivotal for matching given data, i.e. for predictive success, this very success condition also entails a serious disadvantage.[7] Complicated schemes of adjusted parameters might block theoretical progress. In our climate case, any new cloud parameterization that intends to work with a more thorough theoretical understanding has to be developed for many years and then has to compete with a well-tuned forerunner. Again, this kind of problem is more general. In quantum chemistry, many-parameter adaptations of density functionals have brought great predictive success but at the same time render the rational re-construction of why such success occurs hard, if not impossible (Perdew et al. 2005, discussed in Lenhard 2014). The situation in thermodynamics is similar, cf. Hasse and Lenhard (2017).

Let us take stock regarding the first argument for the erosion of modularity. Tuning, or adjusting, parameters is not merely an *ad hoc* procedure to smoothen a model, rather it is a pivotal component for simulation modeling. Tuning convolutes heterogeneous parts that do not have a common theoretical basis. Tuning proceeds holistically, on 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 destructs modularity.

Looking back to Simon's clockmaker story, we see that its basic setting does not match the situation in a fundamental way. The perfect cogwheel picture is misleading, because it presupposes a clear identification of mechanisms and their interactions. In our examples, we saw

[7] There are other dangers, like over-fitting, that I leave aside.

that building a simulation model, different from building a clockwork, cannot proceed top-down. Moreover, different modules and their interfaces get convoluted during the processes of mutual adaptation.

4. Erosion of modularity 2: kluging

The second argument for the erosion of modularity approaches the matter from a different angle, namely from a certain practice in developing software known as kluging (also spelled kludging)[8]. "Kluge" is a term from colloquial language that became a term in computer slang. I remember when back in my childhood our family and another, befriended one drove towards holidays in two cars. In the middle of the night, while crossing the Alps, the exhaust pipe of our friends before us broke, creating a shower of sparks where the pipe met the asphalt. There was no chance of getting the exhaust pipe repaired, but the father did not hesitate long and used his necktie to fix it provisionally.

The necktie worked as a kluge, which is in the words of Wikipedia "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." The notion has been incorporated and become popular in the language of software programming and is closely related to the notion of bricolage.

Andy Clark, for instance, stresses the important role played by kluges in complex modular 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 carried forward their way from programmers' colloquial language into the body of philosophy guided by scholars like Clark and Wimsatt who are inspired both by computer

[8] 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".

modeling and evolutionary theory.[9] The important point in our present context is that kluges may function for a whole system, i.e. for the performance of the entire simulation model, whereas it has 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)

Since kluging stems from colloquial language and is not seen as a good practice anyway, examples cannot be found easily in published scientific literature. This observation notwithstanding, kluging is a widely occurring 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, since 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 remaining code outside the package. These modifications are there for no good theoretical reason, albeit for good practical reasons. They make the overall simulation run as expected (in known cases); and they allow working with existing software. The modifications thus are typical kluges.

Again, kluging occurs in virtually every site where large software programs are built. Simulation models hence are a prime instance, especially when the modeling steps of one group build on the results (models, software packages) of other groups. One common phenomenon is the increasing importance of "exception handling", i.e. 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 effective under particular conditions. Often extensive testing is needed for finding 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,

[9] The cluster of notions like bricolage and kluging common in software programming and biological evolution would demand a separate investigation. See, as a teaser, Francois Jacob's account of evolution as bricolage (1994).

counteracting this particular exception. Exception handling can be part of a sound design process, but increased use of exception handling is symptomatic of excessive kluging.

Presumably all readers who ever contributed to a large software program know about experiences of this kind. It is commonly accepted that the more comprehensive a piece of software gets, the more energy for exception handling new releases will require. Operating systems of computers, for example, often receive weekly patches. Many scientists who work with simulations are in a similar situation, though not obviously so.

If, for instance, meteorologists want to work on, say, hurricanes, they will likely take a meso-scale (multi-purpose) atmospheric model from the shelf of some trusted modeling center and add specifications and parameterizations relevant for hurricanes. Typically, they will not know in exactly what respects the model had been tuned, and also lack much other knowledge about strengths and weaknesses of this particular model. Consequently, when preparing their hurricane modules, they will add measures into their new modules that somehow balance out undesired model behavior. These measures can also be conceived as kluges.

Why should we see these examples as typical instances and not as exceptions? Because they arise from practical circumstances of developing software, which 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* is different from science *in practice*.

In practice, there are strong and constant forces that drive software development into resorting to kluges. Economic considerations are always a reason, be it on the personal scale of research time, be it on the grand scale of assigning teams of developers to certain tasks. Usually, software is developed "on the move", i.e. those who write it have to keep up with changing requirements and a narrow timeline, in science as well as industry. Of course, in the ideal case the implementation is tightly modularized. A virtue of modularity is that it is much quicker incorporating "foreign" modules than developing them from scratch.

If these modules have some deficiencies, however, the developers will usually not start a fundamental analysis of how unexpected deviations occurred, but rather spend their energy for

adapting the interfaces so that the joint model will work as anticipated in the given circumstances. In common language: repair, rather than replace. Examples reach from integrating a module of atmospheric chemistry into an existing general circulation model up to implementing the new version of the operating system of your computer. Working with complex computational and simulation models seems to require a certain division of labor and this division, in turn, thrives on software traveling easily. At the same time, this will provoke kluges on the side of those that try to connect software modules.

Kluges thus arise from unprincipled reasons: throw-away code, made for the moment, is not replaced later, but becomes forgotten, buried in more code, and eventually fixed. This will lead to a cascade of kluges. Once there, they prompt more kluges, tending to become layered and entrenched.[10]

Foote and Yoder, prominent leaders in the field of software development, 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. (…) 2. Reason for degeneration: ongoing evolutionary pressure, piecemeal growth: 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 1999, ch. 29)

I would like to repeat the statement from above that there is no necessity in the corruption of modularity and rational architecture. Again, this is a question of science in practice vs. science in principle. "A sustained commitment to refactoring can keep a system from subsiding into a big

[10] Wimsatt (2007) writes about "generative entrenchment" when speaking about the analogy between software development and biological evolution, see also Lenhard and Winsberg (2010).

ball of mud," Foote and Yoder concede. There are even directions in software engineering that try to counteract the degradation into Foote's and Yoder's big ball of mud. The movement of "clean code", for instance, is directed against what Foote and Yoder describe. Robert Martin, the pioneer of this school, proposes to keep 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", which intends to integrate modularity and piecemeal growth into one "organic order". Anyway, when describing the starting point, he more or less duplicates what we heard above from Foote and Yoder.

Finally, I want to point out that the matter of kluging is related to 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 sensible only in the particular context at the 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 have become a topic when complex systems had become a major issue in discussion computer use. 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)[11]

Thus, the brief look into the somewhat elusive field of software development has shown us that two conditions foster kluging. First, the exchange of software parts that is more or less motivated by flexibility and economic requirements. This thrives on networked infrastructure. Second, iterations and modifications are easy and cheap. Due to the unprincipled nature of kluges, their construction requires repeated testing whether they actually work in the factual circumstances. Kluges hence fit to the exploratory and iterative mode of modeling that characterizes

[11] Interestingly, Jacob (1994) gives a very similar account of biological evolution when he writes that simpler objects are more dependent on (physical) constraints than on history, while history plays the greater part when complexity increases.

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 makes modularity erode and this is the second argument why simulation modeling systematically undermines modularity.

5. The limits of validation

What does the erosion of modularity mean for the validation of computer simulations? We have seen that the power and scope of simulation is built on the tendency toward holism. But holism and the erosion of modularity are two sides of the same coin. The key point regarding methodology is that holism is driven by the very procedure that makes simulation so widely applicable! It is through adjustable parameters that simulation models can be applied to systems beyond the control of theory (alone). It is through this very strategy that modularity erodes.

One ramification of utmost importance is about the concept of validation. In the context of simulation models the community speaks of verification and validation, or "V&V". Both are related, but the unanimous advice in the literature is to keep them separate. While verification checks the model internally, i.e. whether the software indeed captures what it is supposed to, 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." While 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) Though there is some leeway of defining V&V, you get the gist of it from the saying: verification checks whether the model is right[12], while validation checks whether we have the right model.

Due to the increasing usage and growing complexity of simulations, the issue of V&V is itself a growing 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

[12] This sloppy saying should not obscure that the process of verification comprises a package of demanding tasks.

parameter then belongs to a particular type of parameter that determines which specific steps in V&V are required. Oberkampf gives 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,
- uncertainty modeling parameters.” (Oberkampf and Roy 2010, section 13.5.1, p.623)

My point is that the adjustable parameters we discussed are of a type that is evading the V&V fencing. These parameters cannot be kept separate from the model form, since the form alone does not capture representational (nor behavioral) adequacy. A cloud parameterization scheme makes sense only with parameter values already assigned and the same holds for a many-parameter density functional. Before the process of adjustment, the mere form of the functional does not offer anything to be called adequate or inadequate. In simulation models, as we have seen, (predictive) success and adaptation are entangled.

The separation of verification and validation thus cannot be fully maintained in practice. 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 complete conceptual breakdown of verification and validation. Rather, holism comes in degrees[13] and is a pernicious tendency that undermines the verification-validation divide.[14]

Finally, we come back to the analogy, or rather dis-analogy between computer and clockwork. In an important sense, computers are not amplifiers, i.e. they are not analogous to gigantic clockworks. They do not (simply) amplify the force of mathematical modeling that has got stuck

[13] I thank Rob Muir for pointing this out to me.

[14] My conclusion about the inseparability of verification and validation is in good agreement with Winsberg’s more specialized claim in (2010) where he argues about model versions that evolve due to 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 paper, which rests on a tendency toward holism, rather than a complete conceptual breakdown.

in too demanding operations. Rather, computer simulation is profoundly *changing* the setting of how mathematics is used.

In the present paper I questioned the rational picture of design. Also Brooks did this when he observed that Pahl and Beitz had to include more and more steps to somehow capture an unwilling and complex practice of design, or when he refers to Donald Schön who criticized a one-sided "technical rationality" that underlies the Rational Model (Brooks 2010, chapter 2). However, my criticism works, if you want, from 'within'. It is the very methodology of simulation modeling, and how it works in practice, that challenges the rational picture by making modularity erode.

The challenge to the rational picture has quite fundamental ramification because this picture influenced so many ways we conceptualize our world. I will spare the philosophical discussion of how simulation modeling is challenging our concept of mathematization and with it our picture of scientific rationality for another paper. Just let me mention the philosophy of mind as one example. How we are inclined to think about mind today is deeply influenced by the computer and by our concept of mathematical modeling. Jerry Fodor has defended a most influential thesis that mind is composed of information-processing devices that operate largely separately (Fodor 1983). Consequently, re-thinking how computer models are related to modularity invites to re-thinking the computational theory of the mind.

I would like to thank …

References

Agre, Philip E., Hierarchy and History in Simon's "Architecture of Complexity", journal of the learning sciences, 3, 2003, 413-426.

Brooks, Frederic P, The Design of Design. Boston, MA: Addison-Wesley, 2010.

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

Fillion, Nicolas, 2017, The Vindication of Computer Simulations, in Lenhard, J., and Carrier, M. (eds.), *Mathematics as a Tool*, Boston Studies in History and Philosophy of Science, forthcoming.

Fodor, Jerry: *The Modularity of Mind*, 1983, MIT Press, Cambridge, MA.

Foote, Brian und Joseph Yoder, *Pattern Languages of Program Design 4* (= *Software Patterns*. 4). Addison Wesley, 1999.

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

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

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

Hasse, Hans, and Lenhard, J. (2017), On the Role of Adjustable Parameters, in Lenhard, J., and Carrier, M. (eds.), *Mathematics as a Tool*, Boston Studies in History and Philosophy of Science, forthcoming.

Humphreys, Paul, The Philosophical Novelty of Computer Simulation Methods, *Synthese*, 169 (3):615 - 626 (2009).

Jacob, Francois, *The Possible and the Actual*, Seattle: University of Washington Press, 1994.

Lenhard, Johannes, *Disciplines, Models, and Computers: The Path To Computational Quantum Chemistry*, Studies in History and Philosophy of Science Part A, 48 (2014), 89-96.

Lenhard, Johannes and Eric Winsberg, *Holism, Entrenchment, and the Future of Climate Model Pluralism*, in: Studies in History and Philosophy of Modern Physics, 41, 2010, 253-262.

Mauritsen, Thorsten, Bjorn Stevens, Erich Roeckner, Traute Crueger, Monika Esch, Marco Giorgetta, Helmuth Haak, Johann Jungclaus, Daniel Klocke, Daniela Matei, Uwe Mikolajewicz, Dirk Notz, Robert Pincus, Hauke Schmidt, and Lorenzo Tomassini, Tuning the climate of a global model, *Journal of Advances in Modeling Earth Systems*, 4, 2012.

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

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

Oberkampf, William L. and Trucano, T.G., *Validation Methodology in Computational Fluid Dynamics*, American Institute for Aeronautics and Astronautics, 2000 – 2549, 2000.

Pahl, G. and Beitz, W. 1984. *Engineering Design: A Systematic Approach*. Revised editions in 1996, 2007. Berlin: Springer.

Parker, Wendy, Values and Uncertainties in Climate Prediction, revisited, *Studies in History and Philosophy of Science* 2013.

Perdew, J. P., Ruzsinsky, A., Tao, J., Staroverov, V., Scuseria, G., & Csonka, G. (2005). Prescription for the design and selection of density functional approximations: More constraint satisfaction with fewer fits. *The Journal of Chemical Physics*, 123.

Royce, Wynston, Managing the Development of Large software Systems. *Proceedings of IEEE WESCON* 26 (August), 1970, 1–9.

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

Smith, Leonard A., What Might We learn From Climate Forecasts?, in: *Proceeedings of the National Academy of Sciences USA*, 4(99), 2002, 2487-2492.

Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M. Tignor and H.L. Miller (eds.), *Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel*

on Climate Change, 2007. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press.

Stainforth, D.A., Downing, T.E., Washington, R. and New, M. (2007) Issues in the interpretation of climate model ensembles to inform decisions, *Philosophical Transactions of the Royal Society*, Volume 365, Number 1857, 2145-2161.

Wimsatt, William C., *Re-Engineering Philosophy for Limited Beings. Piecewise approximations to reality*, Cambridge, MA and London, England: Harvard University Press, 2007.

Winograd, Terry und F. Flores, *Understanding Computers and Cognition*, Reading, MA: Addison-Wesley, [5]1991.

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