Forthcoming in J. Brown, M. Frappier, & L. Meynell, eds. *Thought Experiments in Philosophy, Science and the Arts*. (London: Routledge)

Computational Modeling: Is this the end of thought experiments in science?

Sanjay Chandrasekharan*, Nancy J. Nersessian*, Vrishali Subramanian**
*School of Interactive Computing
**School of Public Policy
Georgia Institute of Technology, Atlanta, USA

Corresponding Author

Nancy J. Nersessian
Regents' Professor and Professor of Cognitive Science,
School of Interactive Computing, College of Computing,
Georgia Institute of Technology,
Atlanta, GA 30332-0280
Phone: 404-894-1232
FAX: 404-894-0673
nancyn@cc.gatech.edu

1. Introduction

There is general agreement that thought experimenting has played an important role in the creation of modern science. Influential instances of thought experimenting include Newton's bucket, Maxwell's demon, Einstein's elevator and Schrödinger's cat. As James Brown has argued, thought experiments can help us “learn new things about nature without new empirical data;” that is, they help the scientist “get a grip on nature just by thinking” (Brown, 1996). This view leads to an obvious comparison between thought experiments and computational simulation models, which are ubiquitous in most fields of contemporary science. Indeed, the comparison has been noted, but the current framing of the discussion centers on whether computational simulations are “opaque” thought experiments or not (Di Paolo, 2000; Lenhard, 2010).

In this paper, we focus instead on the nature of the intellectual work computational modeling allows scientists to accomplish. Based on this analysis, we argue that computational modeling is largely replacing thought experimenting, and the latter will play only a limited role in future practices of science, especially in the sciences of complex, non-linear dynamical phenomena. We arrive at this conclusion based on three considerations:

- Thought experiments are a product of a more limited material and problem environment. The current material and problem environment is more suited to computational modeling.

- Building computational models can provide deeper insights into problems than building thought experiments.
- The central cognitive role played by thought experiments is a form of simulative model-based reasoning carried out with mental models. Computational models support this type of simulative model-based reasoning, but also allow more sophisticated simulation operations.

These points will be raised in this order in the following sections, but the central argument for them will be developed in two case studies of how computational models are built in bio-sciences and engineering fields, the discovery events associated with these computational models, and a discussion of the major issues raised by these cases (section 2). Section 3 sketches the mental modeling argument for thought experiments, and expands on some of the minor roles thought experiments might still fill in science practices involving complex, non-linear, dynamical phenomena.

1.1 The need for building models

A signature practice of contemporary bio-sciences and engineering is building physical (*in-vitro*) simulation models and computational (i*n-silico*) simulation models of phenomena of interest. However, very few accounts of this building process exist in the wide literature on modeling. An example of a built physical model is the *flow loop* (Nersessian and Patton, 2009), which is an engineered artifact that simulates, and thus helps examine, different types of blood flow (laminar, oscillatory), and their influence on gene expression and other cellular-level activity in tissue-engineered blood vessels, which are built-models of human arteries. Another example is a neuronal dish model (Nersessian and Patton, 2009; Nersessian and Chandrasekharan, 2009), which simulates learning in a living neural network, and enables researchers to examine the neural mechanisms underlying learning. Specifically, how neurons with their connections broken apart come together, to form new connections and learn new behavior. Examples of computational models built to mimic biological phenomena of interest include simulation models of the plant cell wall, and models of dopamine transport in Parkinson's disease. There are also "second order" computational models, such as a model built to simulate processes in the above-mentioned neuronal dish, which is a computational model of a physical model. Sometimes computational models are also built in a chain structure. The first model (say, a biophysical model) will generate virtual data of a downstream process (say, the generation of a metabolite), and these data help in the building of models of the second process, particularly in the estimation of model parameters. Building such facsimile models that mimic or approximate phenomena of interest is not a new practice. For instance, wind tunnels have helped mimic different aerodynamics for at least a hundred years (though they are now being replaced by computational versions). Recently, however, a combination of four factors has made this practice of building facsimiles more widespread, particularly in the bio-sciences and engineering fields:

- The complex, non-linear, and dynamic nature of the problems investigated in contemporary biology (and recent science in general), *requires* building such models. This is because it is almost impossible to develop detailed conceptual models of cellular and molecular-level interactions in your head, or using pencil and paper, as these processes involve many levels of structure, and can occur simultaneously, or across different time scales.

- Massive amounts of data are now generated by experimental work in many areas of biology, such as high-throughput data in genetics. Computational models, particularly statistical models, are usually required to interpret these data, since the interactions between different variables are complex. Also, the technology that is used to generate the data itself is based on statistical assumptions.

- Data in biology are closely tied to their context (specific cell lines, animals, diseases etc.), and there is no theory that helps structure all these disparate and scattered data. Building computational models helps bring together these data in a structured fashion – the models thus play a 'book-keeping' function, since they comprehensively capture data elements that are related, and also try to account for how the elements are related. Because this data structure is dynamic and can be run with various inputs, the models can be thought of as 'running literature reviews'.

- The development and easy availability of new technology that supports modeling and rapid prototyping has made modeling more widespread.

These factors, together with the technological resource environment of contemporary science, are driving the practice of building models. Thought experiments emerged in a resource environment where the only cognitive tools available were pencil and paper and the brain, and the problems tackled were usually highly idealized to fit this resource environment. As such, the thought experiment method cannot generate and test the complex, dynamic and non-linear phenomena investigated by contemporary science. Building physical and computational models can help in understanding such phenomena. Physical models, however, do create new experimental data, and their relation to thought experiments is more complex than computational models. For clarity of analysis, in this paper we focus on the processes involved in building computational models.

2. Building vs. Interpretation

As noted above, there is an obvious point of comparison between computational modeling and thought experimenting, since both help us learn new things without generating new empirical data. However, the current state of this comparison is limited in two respects. One is that the focus of much discussion is on interpretation, rather than building. Thought experiments (TEs) can be considered to consist of two phases, roughly, a 'building' phase and an 'interpretation' phase. When a thought experiment is presented, what we get is the final, polished product, usually a narrative, which is the end-point of a long building process (Nersessian, 1992). Since the building happens inside the presenter's head, others do not have access to this process, which is one of the reasons for the focus on the interpretation phase. The interpretation phase involves relating the final results of the TE to a theory or phenomena. However, we do have access to the processes of building computational models, so the focus on the interpretation phase is not justified in the case of models. As an illustrative instance of this focus, it has been argued, based on a comparison of the interpretation phase, that models are much more opaque than TEs, and models require 'systematic enquiry', i.e. probing of the model's behavior (Di Paolo, 2000). TEs are considered to be more 'lucid' than computational models (Lenhard, 2010), though it is not clear what is meant by lucid in this context, particularly given the extensive discussions about what some TEs – such as Einstein's elevator or Schrodinger's cat – actually show.

A second issue with existing comparisons of TEs and models puts the focus on possible identity relations between thought experiments and models (Are simulations thought experiments?) and the identity relations between modeling and other methods in science (Are simulations experiments?). These questions often are addressed using feature and taxonomic analysis (necessary or sufficient conditions for something to be an experiment/theory/TE), which is a good starting point for understanding the nature and contribution of computational modeling. However, such taxonomy exercises compare the end-points of computational modeling and thought experimenting, and ignore the insights that could be gained by studying the model building *process*, which can be tracked in the case of computational modeling. We hope to shift the discussion away from interpretation, taxonomies, and identity conditions to process-oriented analyses of modeling, and the role models play in scientific practice. Our analysis puts the focus on the process of building models and examines how studying this process can inform us about the relationship between computational models and TEs. In the following section, we present two case studies of building computational models, based on our ethnographic studies of two leading research labs, one in systems neuroscience, and the other in systems biology.

2.1 Case 1: A dish, a model, and the CAT

The episode of conceptual innovation summarized here occurred over a two-year period in a neural engineering laboratory, and involved constructing a computational model of an *in vitro* physical model of cortical neural network activity. The model was developed to understand certain thought-to-be undesired phenomena – spontaneous ‘bursts’ – taking place with high frequency in the *in vitro* model, but not in properly functioning *in vivo* animal brains. Importantly, as we will outline below, the computational model led to a reconceptualization of “burst” phenomena in the *in vitro* model, and the development of “programmable” neurons (See Nersessian & Chandrasekharan, 2009 for a fuller discussion).

The lab’s central research problem in the period of our investigation was to develop an account of learning and plasticity in networks of cultured neurons, which were thought to more closely model learning in the brain than single neurons. To address this problem experimentally, the lab had developed a physical model-system for constructing and investigating plasticity: an *in vitro* network of cultured neurons locally referred to as “the dish.” Building this *in vitro* model-system involves extracting neurons from embryonic rats, dissociating them (breaking the connections between neurons) and plating them in a single layer on a dish with embedded electrodes known as an MEA (multi-electrode array), where the neurons regenerate connections and become a network. The researchers “talk to the dish” by stimulating the neuronal network with different electrical signals (electrophysiology) and feeding the output to “embodiments” (robotic or simulated bodies) that support closed-loop feedback (using a translator program that maps the dish signal to motor commands). The embodiments include both robotic devices and visualized “animats” that move around in simulated computational worlds.

The stated goal of the research was to understand the dynamics of learning in the neuronal network in such a way that it would lead to the development of a control structure, which would allow the dish to be trained to control the embodiment systematically, using feedback. The design of the dish incorporated constraints from the current understanding of neuro-biology and chemistry, as well as those relating to electrical engineering and other technologies used in the lab. To begin with, the researchers explored the problem space by stimulating the neuronal network using different electrical signals, and tracking the output (“playing with the dish”). The first real experiment was initiated by researcher D4, where she tried, unsuccessfully, to replicate a plasticity result reported by another group. One of the problems she faced was bursting - a form of network wide electrical activity spontaneously exhibited by the *in vitro* neuronal networks. Bursting created a problem in understanding plasticity, because it prevented the detection of any systematic change that arose in the network in response to the controlled stimulation. According to D4, whenever she conducted a plasticity experiment, the network produced bursts. The lab interpreted those as “*noise in the data…noise interference in the way…so it is clouding the effects of the learning that we want to induce*.”

The group hypothesized that bursting arose because of deafferentation; that is, because the neurons lacked the sensory inputs they would ordinarily get if they were in a live animal’s brain. Given this view, and the engineering conception of noise as something to be eliminated, D4 began working on quieting bursts in the dish. She hypothesized that it would be possible to lower bursting by providing the network with artificial sensory input, and for this she chose electrical stimulation. She tried a range of stimulation patterns to lower the bursting activity in the networks, and achieved a breakthrough, managing to quiet bursts entirely in a neuronal network. However, in the next six months of research, D4 was unsuccessful in inducing learning in the quieted network. This was mostly because of a "drift" phenomenon: the activity pattern evoked by a stimulus did not stay constant across trials, but drifted away to another pattern. This drift prevented her from tracking the effect of a stimulus, because the network never responded in the same manner to a constant stimulus.

Early in the period when D4 was trying to quiet the network, D11 branched away from working with the *in vitro* model-system, to develop a computational model that mimicked it. As he put it, "*the advantage of modeling [computational] is that you can measure everything, every detail of the network...I felt that modeling could give us some information about the problem [bursting and control] we could not solve at the time* [using the *in vitro* dish model-system]." D11 felt that to understand the phenomena of bursting he needed to be able to 'see' the dish activity, make precise measurements of variables such as synaptic strength, and to run more controlled experiments than could be conducted with the physical dish. This different perspective illustrates a fundamental property of building models: the construction process supports a plurality of designs and views, and also provides a multifaceted approach to the problem. There is no requirement that everyone work with a standard model structure. Each team member can start from a broad definition of the problem, and build up his or her own model.

Interestingly D11 built the initial *in silico* model not using the experimental data from their dish, but drawing from intra-domain sources in neuroscience; in particular, from studies involving single neurons, brain slices, and other computationally simulated networks, and using only design features of their dish. His model was tested and optimized with data from other MEA dishes first, and only after the behavior replicated the experimental results in the literature did he use their own data (see Figure 1). The model, as developed, is thus a second-order *in vitro* system - an *in silico* model of the activity of a *generic* dish.

Importantly, the constraints the model adapted from the lab's dish were not the group's experimental outcomes (i.e. the behavior of their dish), but had to do with the construction and structure of the dish. These included the area of the artificial neurons, the placement grid of electrodes, the number of electrodes used for recording and stimulation, and the random location of the neurons. The other parameters (constraints) of the model, such as type of synapses, synaptic connection distance, percentage of excitatory and inhibitory neurons, conduction delay, conduction velocity, noise levels, action potential effects, and spontaneous activity were based on results reported in the source literature. Further constraints came from a neuroscience modeling platform (CSIM), using a standard simple model of neurons, known as 'leaky-integrate-fire.' (This name derives from the way the artificial neurons respond to stimulus).

[Insert Figure 1 here]

After several months of building tentative versions of the computational simulation and probing these, D11 started to get what he called a “feel” for how the computational network behaves under different conditions. He then developed a visualization that captured the activity of the network as it ran, which figured centrally in the development of the novel concept of ‘center of activity trajectory’ (CAT). As he stated: “*I need to—really need to look at the figure to see what is going on. So after I constructed this network – I just let it run and we visualized everything in it.*” Using the visualization, he was able to replicate successfully some of the results reported in the literature, and then results from their own dish. A major contribution of the visualization was that it enabled D11 to observe - literally see - interesting spatial patterns in the way the model responded to different stimuli. These patterns were novel and distinct from what was known about the behavior of cultured dishes.

It is important to highlight the differences between the computational model’s visualization and the visual representation that was being used for the *in vitro* dish. The computational model’s visualization of the network’s activity shows the movement of an activity pattern across the entire network, in real time. In the visual display of the *in vitro* model-system, the individual neuronal activity is hidden. One can see activity across a channel as in Figure 2, but this display can only track activity at each electrode of the *in vitro* system, using graphs they had developed for this purpose (see figure 2).

[Insert Figure 2 here]

The visualization captures which electrode is activated and by how much, but it does not have a representation of the entire network itself – it cannot capture *burst movement across the network*. Thus, it was not possible to see from the dish display whether there were patterns moving across the network, and there are no experimental methods that display such dynamic patterns. Such patterns, however, were revealed by the visualization of the *in silico* model. So, the image of changes across discrete channels, as in Figure 2, was replaced by an image of changes of the activity of the whole network as in Figure 3, for instance, from 3a to 3b. (Figure 3c represents the CATs for changes of activity, as we will discuss below.)

[Insert Figure 3 here]

The computational model offered other advantages in exploring burst phenomena. The simulated network could be stopped at any point and started again from there. Further, it became possible to provide detailed measures of significant variables, such as synaptic strength, which are not accessible using the *in vitro* model. Finally, a large number of experiments could be run at no cost, since the computational model could be changed easily and did not require the careful, laborious, and expensive processes involved in setting up and maintaining a living dish. When coupled with the visualization that enabled tracking the activity of the network as it was happening, these features proved to be a powerful combination. They gave D11 immediate access to a range of configurations and data that the living dish could not provide, and the data could be examined and reexamined, and comparison experiments run instantly. In the process of running numerous simulations, he noticed that there were repeated *spatial* patterns in activity, as the activity propagated across the network. The spatial patterns were seen both when spontaneous bursts arose in the model network, and when the model network responded to stimuli. Basically, he found that there were “similar looking bursts” that propagated across the network, and there appeared to be a limited number of what he called “burst types.” D11 began working with D4 and D2, a graduate student who had been working on dish embodiments. The group then decided to investigate bursts further together, as a possibly interesting pattern, i.e. *a signal*. Note the radical change in perspective here, where ‘burst’ moves from being something akin to noise that needed to be eliminated, to the status of a pattern, a signal that could possibly lead to a control structure for training the network. Note also that this change arises from *the group* running many computational simulations involving different situations and parameters over time, and this group process supported arriving at a consensus on the conceptual shift. This group process was possible because of the *manifest* nature of the built model.

Further research proceeded in large part by mapping potential problem solutions from the *in silico* model to the *in vitro* model. In particular, the computational model helped them get past the problem of the drift behavior of the dish. The group came up with a range of ways to quantify the spatial properties of moving bursts, using clustering algorithms and statistical techniques, and these measures proved to be immune to the drift problem. Of particular interest to us here is the fact that they were first developed for the computational model and then equivalents (by analogy) were developed for the *in vitro* model. These included several conceptual innovations: ‘burst types,’ ‘spatial extent’ (an estimate of the size and location of the burst in the dish), ‘center of activity trajectory’

(CAT, a vector capturing the spatial location of the electrode along with the firing rate), and ‘occurrence’ of burst types. These spatial measures of bursts were then shown to be better indicators of plasticity than responses to probe stimuli.   So, in the matter of a year, based to a significant extent on the patterns generated from the computational model’s visualization, the group’s theoretical position had shifted from bursts as noise to bursts as signal and a possible control structure.

All these measures were driven by the spatial patterns of activity noticed in the visualization of the computational model of dish activity. But of these, CAT is probably the most noteworthy, for two reasons. First, because it is a novel concept, articulated on an additional analogy with the physics notion of center of mass, which was applied to the neuronal activity pattern. Second, because it emerged entirely from the computational model, and would be almost impossible to conceptualize and properly formalize without it. Figure 3c is a visualization of the CAT for a changing network burst activity in an *in silico* dish. CAT is an averaging notion, similar to the notion of a population vector, which captures how the firing rates of a group of neurons that are only broadly tuned to an action or stimulus (say an arm movement), when taken together, provide an accurate representation of the action/stimulus. However, CAT is more complex than the population vector because it tracks the *spatial* properties of activity as it *moves* through the network. For instance, if the network is firing homogenously, the CAT will be at the center of the dish, but if the network fires mainly at the left corner, then the CAT will *move* in that direction. CAT thus tracks the flow of activity (not just activity) at the population scale, and on a much quicker time scale than population vectors.

When applied to the *in vitro* network, CAT provides a novel representation of neuronal activity - a new concept. Figure 4 provides a side-by-side comparison of a CAT visualization that corresponds to the MEAscope representation for an *in vitro* dish. The MEAscope representation (4a) shows activity across each recording channel over time, but the CAT (4b) is an integrated representation of activity across the entire network.

[Insert Figure 4 here]

The CAT is like a signature for a burst type in that each burst type has a corresponding range of similar-looking CATs specific to that type. Although the CAT concept arose from the visualization, the relation between the CAT concept and the visualization is not a direct mapping link. The visualization worked as a generator of many types of activity, which when put together, led the new CAT concept. Although our account ends here, in later work, the researchers combined CAT and techniques developed for burst quieting to develop a set of stimulation patterns (a control structure) for the dish that led to supervised learning by the living neuronal network, in effect making the *in vitro* dish neuron network programmable. Using these stimulation patterns, the living network in the dish was trained to control a computationally simulated creature, and a robotic drawing arm.

2.1.1 Discussion

The most puzzling aspect of this case is that the network visualization used by D11 is largely *arbitrary*. There is no reason why a network representation should be used – he could have used one similar to their graph representation of the living dish – and even after choosing a network representation, there are many ways in which the network activity could be represented. Thus the central question raised by this case of discovery-by-building is this:

How could the spatial analysis of an arbitrary representation translate to the actual dish model, and provide a control structure for supervised learning?

As D11 says: *"From the simulation, I definitely get some….y'know, if you run it for a year, y'know, you have this feeling for what the network you construct is doing…so I have a specific idea about what my simulated network is doing…but the problem is that I don't think it is exactly the same as in the living network…when our experiment worked in the living network, actually I am… I am surprised… I was surprised…"*

We believe the answer to this question lies partly in the iterative building process, and partly in the nature of the visual representation. In this case, the visual representation captures the underlying artificial network's structure in an integrated fashion. The underlying network is being continually reconfigured by each experiment replication – each replication is adding complexity to the network, as it builds on the network created by the previous replication. Since the visualization captures the network structure *as a whole*, it is being fine-tuned to more closely mimic the *behavior* of the network, until the system gains sufficient complexity to *enact the behavior* of the dish. Seeing these behaviors and having the ability to stop-and-poke them, while also examining the numbers that generate them, allowed the researchers to develop the CAT concept. CAT captures activity patterns, which is an abstraction that is delinked from the (arbitrary) representation. CAT can be applied to *any* activity. It captures the model's replication of the dish activity; its ability to capture the activity of the actual dish follows from this. Note that the CAT concept cannot be directly mapped to the visualization – the visualization is a "generator" of spatial activity (behavior), which leads up to the CAT concept, and this abstract concept is what transfers to the dish.

More generally, building a computational model involves developing a system that can *enact* the behavior of the phenomena under study. Each run during the building process reconfigures this enactive ability, until the model gains sufficient complexity to enact the behavior of the dish. The final runs are a *probing* of this enactive ability, the possibilities of which the modeler does not entirely understand, but has a "feel" for. The intellectual work at this stage is not interpretation of the results, but the use of different probes to generate a range of behaviors, and the capture of this range of behavior within a "margin of error" framework. This range of error is then used to develop a control structure.

In contrast, the standard criticism of a computational model not being able to provide any new knowledge (because what is “fed into it” is already known, Di Paolo, 2000) suggests that the building process is conceived as developing a representation that just incorporates all the existing data. The final running of the model then puts these data in a new configuration, and all the work is in interpreting how the new data configuration generates the resulting data. This criticism stems from an information “translation” approach to understanding computational modeling, which misses two important points. First is the idea that different kinds of representations provide different insights. Second is the idea that individual elements show different properties when combined into a global structure. The translation view misses the importance of the enaction of the behavior, since only the final numbers and their graphs are relevant. In this view, the transfer of the CAT concept from the visualization to the dish is mysterious, as it misses the critical thing about the visualization, which is that, even though arbitrary, it captures the enactive ability of the network as a complex *behavior*.

2.2 Case 2: pathways, parameters, and engineers

In our current project, we are studying practice in two integrative systems biology labs. The research in one of the labs (Lab G) is purely modeling – building computational models of biochemical pathways, to understand phenomena as varied as Parkinson’s disease, plant systems for bio-fuel production and atherosclerosis. Lab members mostly build ordinary differential equation (ODE) models, which capture how the concentration levels of different metabolites in a given biological pathway change over time.

The first step in this building process is the development of a pathway diagram, which shows the main reactions involved. The pathway diagram also captures positive and negative regulation effects, where the presence of different metabolites has a positive or negative influence on different reactions (Figure 5). A rough diagram of the pathway is usually provided by the lab’s experimental collaborators. But the modelers, who mostly come from engineering backgrounds, have to estimate the details of the pathway by themselves, particularly values of parameters related to metabolites, such as speed of change (kinetic order) and concentration level (rate constant), which are usually not measured by experimenters. Some of this information is available in rough form (with varying degrees of reliability) from some online databases, but most of the time these values need to be estimated, usually by testing the model using a range of numbers as parameter values.

Modelers also add some components to the pathway, usually metabolites that are known to interact with elements in the pathway provided by the experimenters. These connections are found by reading and searching biology journal articles and databases related to the problem they are modeling, and also based on results from preliminary models. Even though much of the pathway is provided by experimentalists, these kinds of additions based on literature searches are required, because the provided pathway does not identify all the components and the regulatory influences they have on the reaction.

[Insert Figure 5 here]

The pathway developed by the modeler thus brings together pieces of information that are spread over a wide set of papers and databases. This pathway is usually trimmed, based on some simplifying assumptions, mostly to lower the mathematical and computational complexity involved in numerically solving the differential equations, which are developed in the next step. After the trimming, equations are generated to capture the trimmed pathway. A variable is used to represent the metabolite, while the speed of its change (kinetic order) and its concentration level (rate constant) are represented by parameters, which can take many values. The next step involves estimating values for these parameters, and these values are then used to initialize simulations of the models. The simulation results are then compared to actual experimental results. Usually, modelers split available experimental data into two, one set is used to develop the model (training data), and the other set is used to test the completed model (test data). When the data do not fit the training data, the parameters are "tuned" to get model results that fit.

Importantly, the linear work flow suggested by the above description is very deceptive – the modeling process is highly recursive. For instance, to develop the pathway diagram, preliminary models are built using the network provided by the experimenters, and these are run using tentative parameter values, and the generated model data are fit to the training data. The parameter values are then revised based on this fit. If the model data do not fit after a large number of these parameter revisions, the modeler will add some components to the network, based on elements that are known (in the literature) to be related to the pathway. Thus the model's components – elements and values – are set by building and running the model. These revisions are discussed with the collaborators, and if a revision is considered "reasonable" by the experimenter, it is added to the pathway. The pathway identification process is usually "bottom-up" and creates a "composite" network, made up of parameter values, and metabolites, based on experiments in different species, different cell lines etc.

One of central problems the lab members face is the unavailability of rich, and dependable, data. In modeling, data are used for many purposes. One central use of data is to establish that the model captures a possible biological mechanism, and this is done by showing that the model's output matches the output from experiments (fitting data). Data are also used to tune parameter values during the training phase of building the model, since the fit with the experimental data from each training simulation can indicate how the parameter values need to be changed to generate model data that fit the training data. However, this latter use is dependent on the type of data available. Most of the time, the available data are 'qualitative' in nature – usually how an experimental manipulation changed a metabolite level from a baseline. Mostly, this is represented by a single data point, indicating the level going up or down, and then holding steady. However, when this type of "steady-state" data fit the results of the model, this fit does not indicate an accurate model, since sparse data can be fit with a range of parameter values. Further, since the pathway is an approximation, the modeler is uncertain as to whether the lack of a unique and accurate solution is due to poor estimation of parameters, or because some elements are missing from her pathway.

To get a unique or near-unique parameter set, the data should be quantitative or time-series, where the variation of the metabolites is captured across a large number of time points (i.e. not just a few time points and then steady state). If a pathway is known (usually by bottom-up, "composite" model building), then the time-series data provide a way of better estimating the parameters, since this type of data is "thicker," and a good fit would indicate that the model is close to an actual network. Apart from this advantage, time-series data also provide the possibility of inferring the structure of the network from the data. Extracting the network information, which is implicit in the data, is known as "top-down" modeling. It is also known as the inverse problem.

As an example instance of modeling in this lab, consider G12, an electrical engineer by training, who is modeling atherosclerosis. She works in collaboration with a nearby lab, which has data on how oscillatory flow in blood vessels leads to atherosclerosis via a mechano-transduction process (where the style of flow of blood – laminar vs. oscillatory – leads to the expression of different genes). This in turn leads to the attraction of monocytes to locations experiencing oscillatory flow, and over time, this monocyte recruitment leads to development of plaques that are characteristic of atherosclerosis.

When she started her modeling, she had no background on atherosclerosis. She was provided a rough outline of the pathway by her collaborators, and she learned more about the pathway by reading papers. The initial papers were from the collaborating lab, but then she spread out using the reference lists of those papers. The data available were mostly steady-state data (though for some parts of the network, there was also some time-series data). Once she had read a number of papers, she started building rudimentary computer models and testing these using available data. She then added some components to the model based on connections in the literature, and some of them were considered "reasonable" by her experimental collaborators.

Estimating parameter values for her model, however, was a tough problem, since the data were sparse. To get parameter values that generated model data that fit the training data, she ran a large number of simulations and compared the results with the training data. About this process, she said:

I kind of tune the parameter and and I don't know, it's not a method or, it's not algorithm, and I don't know how to explain.

Finally, she arrived at a set of values that generated data that roughly matched the training data. Once this was done, she tested her model against the test data, and got a rough fit there as well. Based on this fit, she generated a number of predictions from the model, by changing the parameter values.

This case is representative of much of the modeling in this lab. For this paper, we just wish to highlight the fact that engineers with no background in biology and experimentation use a recursive building strategy to add components to a pathway given by experts and estimate the parameter values for elements in the pathway.

2.2.1 Discussion

The central question raised by this case of discovery-by-building is this:

How could building a model suggest additions to the pathway, and generate values for its own variables? How could a model facilitate the prediction of its own parts?

Once again, the answer comes from the *behavior* of the model, which emerges from the complexity it has acquired during the building process. When an expected behavior does not emerge, the modeler can add "sham complexity" to the model, to see whether the desired behavior emerges. When it does, she can work backwards from what she added, to make a prediction about which actual component from the literature could be added to the pathway for a certain behavior to emerge. The global structure of the built-model thus sets constraints, and provides pointers, on how building can progress. Note that the translation approach cannot account for this result, as it assumes all information to exist before it is translated to the model. Building is thus a radically different activity from translation, because the structure and behavior of the model can specify the possible changes that could lead to a better fit with experimental data. A close analogy would be developing a scale model (called a maquette) in architecture, which is not a translation of the blueprint, since the scale model's three-dimensional features and global structure generates behavior that the blueprint does not have, such as balance and stability. Building computational models also help precipitate implicit knowledge, by bringing out individual elements known by an expert (say, a biologist), and inserting these elements into slots in a broader structure. The broad structure is built by the modeler, and the element to be inserted is identified by the iterative process of fitting the model data with experimental data. Not all the slots are filled in advance by the biologist, but some are proposed by the modeler, based on how the model behaves. These candidate elements are considered verified when they are considered "reasonable" by the biologist. The extension of the pathway thus happens through a form of consensus building and negotiation between two different practices, and it creates new knowledge by providing a larger context for individual pieces of knowledge, which themselves are tentative.

Such contextualization has been implicated in new discoveries, as in the case of the discovery of Buckministerfulerene, for which the discoverers got the Nobel Prize. The C_{60} football-shaped molecule was predicted to exist by a Japanese group many years earlier, and was also imaged by another Japanese group, but both these groups did not understand the significance of this molecule. The first group in fact considered it uninteresting. This, and other such cases of missed discoveries, have led to a distinction to be made between 'finding', and the 'recognition of a finding'', which is based on contextualizing a finding (Osawa, 1993; Berson, 1992). The models in Lab G can be considered to serve this contextualizing function for biological elements and can thus support significant discoveries.

2. 3. Translation vs. Enactment

The above two case studies show that the model building process is very complex, and is not a simple translation process. Specifically:

1) Arbitrary representations can gain sufficient complexity, to the extent that their spatial features can be used to control a system in the real world

2) A model can predict its own parts

To explain these features, we need to approach the model as an "enactive system" that gains sufficient complexity from building, such that it can act out the behavior of the phenomena under study. We contrast this enactive approach with a translation approach to computational modeling, which treats the building process as feeding existing information into a model (Di Paolo, 2000). In the latter view, any new information comes from changing the parameters in the final run, and interpreting the results. In the enaction view, the building process brings together information in an integrated fashion, creating a structure that does not exist in such a form in the literature, and thereby allowing the model to replicate a range of *behaviors* under investigation, rather than replicate scattered individual results. The model supports the generation of a range of such behaviors. Researchers can stop and study these behaviors at any point they want, while also studying the underlying mathematical structures that generate these behaviors. This means the interaction between the modeler and the final model is an "interrogation" process, not interpretation. As the model has complexity comparable to the real phenomena, the modeler is interrogating the *different possible variations of the real world* by proxy as well. That is, the computational simulation is not only a dynamical analog to the real-world system, but it also supports the generation of counter-factual scenarios not seen in the real-world, and also their existence conditions (different parameter settings).

This analysis suggests that computational models are close to physical models like the flow loop, and they thus allow researchers to do even more than "get a grip on nature just by thinking." Models serve an enactive function, and while they do not generate new empirical data, they can help to radically restructure the researchers' conceptual landscape. They also restructure the way researchers approach data, supporting an interrogation mode, rather than just an interpretation mode. For all the above reasons, computational models are more sophisticated conceptual tools than thought experiments.

3. Mental simulation: modeling meets thought experimenting

Our previous work has examined the cognitive processes that underlie the building of models and TEs. Mental simulation was proposed as the cognitive process underlying the building (and interpretation) of TEs (Nersessian, 1992). In recent work, we have proposed mental simulation as the mechanism underlying the building (and interpretation) of physical models (Nersessian, 2008) and computational models (Chandrasekharan, 2009). Combining these two views, TEs, physical models and computational models form a spectrum of simulative model-based reasoning. All these types of modeling generate and test counterfactual situations that are difficult to implement in the real world.

[Insert Figure 6 here]

However, TEs are built using concrete elements, while models are built using variables. This means TEs do not *naturally* support simulation of counterfactual scenarios beyond the one generated by the specific scenario and its elements, as the mental simulation process is driven by the behavior of the concrete components. A difficult and complex cognitive transformation is needed to move away from the concrete case to the abstract and generic case. Even when this happens, the generic case does not have much detail, since the elements in the generic case are abstracted from the elements of the TE, which tend to be constructed using visually salient entities. On the other hand, the abstract and generic case is what a modeler works with from the outset, and it is very detailed, with elements at the cellular or molecular level, and each element available for manipulation. Since models are made entirely of variables, they naturally support thinking about parameter spaces, possible variations to the design commonly seen in nature, and why this variation is commonly seen, instead of many others that are possible. The contrast between the commonly seen case and other possible cases, and the detailed manipulation and analyses of the parameters associated with these cases, can lead to the identification of broader design principles (such as thermodynamic constraints) that lead to one parameter set being commonly instantiated in nature.

Computational models, particularly ones with visualizations, go beyond TEs, as these models can generate complex behavior similar to natural phenomena, provide explanations of why these phenomena exist and not others, and they also allow interrogation and “drilling down” into the behavior for details. Such models thus enable the externalization of creative thinking in science, and thereby extend the cognitive capacity for mental simulation – similar to the way telescopes have extended the perceptual capacity for seeing (see Chandrasekharan & Nersessian, In Press, for a discussion of other cognitive roles played by external representations,). And just as no scientist studies the stars with the naked eye anymore, no one would use TEs to probe the complex phenomena studied by contemporary science.

3.1. Thought experiments to go mini

This does not mean thought experiments have come to an end. As systems that generate counterfactual scenarios, thought experiments will still be used in the practice of science, but they are unlikely to have the same importance as they have historically. They are unlikely to settle a debate. Rather, they will contribute to the design and building of computational models, which would provide detailed accounts of the counterfactual scenarios which TEs sought to test in the past. For instance, the scenario “Can giraffes swim?” which could only be examined using thought experiments, was recently examined in detail using a computational model, and it was shown that they could indeed swim, but poorly, and it would require holding their body in non-standard ways (Henderson & Naish, 2010).

The building of models also makes possible new thought experiments, both within the model structure (for instance, what would happen to the Giraffe if I replace the digital water with digital oil?), and across model structures (Would it be possible to integrate this model of foraging with that model of mate selection?). Finally, TEs in small scale (mini-TEs) can contribute to the building of the models themselves. For instance, mini-TEs were involved in choosing the constraints (Dish, CSIM) and parameters in the first case study. In the second case study, mini-TEs were involved in the trimming of the pathway (What would be the effect of removing this component?) and judging parameters (Given what we know about the data, what could be possible values for this parameter?)

In sum, there still will be a role for TEs in science, but it is likely to be a background and subsidiary role to computational modeling. We capture this role in the graphic below, where the two kinds of mental simulation processes still exist, but the output of the TE one is in shadow, as it plays only a background role. Further, the two outputs interact, as TEs contribute to models, and the building of the models generate further TEs.

[Insert Figure 7 here]

Conclusion

Based on the analysis of two case studies examining the building of computational models in neural engineering and integrative systems biology, we showed that building is not a direct translation of existing data. It is better understood as a constitution process that generates an enactive system that can replicate behaviors that are seen in the real world. The components of models interact in ways that are not readily apparent to the modeler, because the model brings together disparate data from a range of resources. However, since the model can be stopped at any point, and its internal states interrogated, the modeler gains intuitions into the way behaviors are generated. This allows her to develop abstractions that transfer to actual phenomena in the world. It also allows her to make predictions about elements that are missing from what is known about the phenomena.

Models thus function as externalized probes that extend human cognition, similar to probes like telescopes and microscopes that extend human vision. Both TEs and computational models support mental simulation of counterfactual scenarios, but TEs do not naturally support probing all possible scenarios, as they are concrete scenarios, built using specific components, while models are built using variables, which naturally support examining a range of possibilities within the parameter space, and why a specific one is instantiated. TEs are a product of a previous resource environment, where the only tools available were pencil, paper, and the brain. This method addressed problems that could be idealized to fit this resource environment, but it is not suited to the many of the current problems that need to retain their complexity. Computational modeling thus has features that go well beyond thought experimenting, and this suggests that just as no one uses their naked eye to study stars anymore, no one would use TEs to study the complex, dynamic and non-linear behaviors that are the focus of contemporary science. TEs still have a role to play in science, but largely within the context of computational modeling – helping design, build and extend models.

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation ROLE Grants REC0106773, DRL0411825, and REESE grant DRL0909971 in conducting this research. Our analysis has benefited from discussions with members of our research group, especially Wendy Newstetter, Lisa Osbeck, Christopher Patton and Costas Mannouris. We thank the members of the research labs for allowing us into their work environment, letting us observe them, and granting us numerous interviews. Thanks to Joshua Aurigemma for the modeling process graphic.

<u>Figure legends</u>

Figure 1: Iterative modeling processes

Figure 2: MEAscope display of bursting phenomena

Figure 3: Visual representations of changes in activity across the in silico network. (a) Time 1, (b) Time 2, (c) CAT T1 to T2

Figure 4: Visualization of an *in vitro* CAT (b) and corresponding dish burst activity (a).

Figure 5: Biological pathway under construction. Names of pathway components have been replaced by alphabet aliases. "??" indicate tentative relationships derived from the literature.

Figure 6: Mental simulation as the cognitive mechanism underlying building of models and thought experiments.

Figure 7: Subsumption of thought experiments within computational models in contemporary scientific practice.

References

Berson, J. A. 1992. “Discoveries missed, discoveries made: creativity, influence, and fame in chemistry.” *Tetrahedron* 48:3-17.

Brown, J. R. 1986. “Thought experiments since the scientific revolution.” *International Studies in the Philosophy of Science* 1, no. 1: 1-15.

Chandrasekharan, S. 2009. “ Building to discover: a common coding model.” *Cognitive Science* 33, no. 6: 1059-86.

Chandrasekharan, S., Nersessian, N.J. (In Press). “Building cognition: the construction of external representations for discovery”, Proceedings of 33rd annual meeting of the Cognitive Science Society, Boston, MA.

Di Paolo, E. A., Nobel, J., and Bullock, S. 2000. “Simulation models as opaque thought experiments.” In *Artificial Life VII: The Seventh International Conference on the Simulation and Synthesis of Living Systems*, ed. M. A. Bedau, J. S. McCaskill, N. H. Packard, & S. Rasmussen, [Cambridge, MA: MIT Press/Bradford Books, 2000.] 497—506.

Henderson, D. M., and Naish, D. 2010. “Predicting the buoyancy, equilibrium and potential swimming ability of giraffes by computational analysis.” *Journal of Theoretical Biology* 265: 151-9.

Lenhard, J. 2010. “When Experiments Start. Simulation experiments within simulation experiments.” Paper presented at the International Workshop on Thought Experiments and Computer Simulations, Paris.

Nersessian, N. 1992. “In the Theoretician's Laboratory: Thought Experimenting as Mental Modeling. ” PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association 2 : 291-301.

Nersessian, N. J. 2008. “*Creating Scientific Concepts*.” Cambridge, MA: MIT Press.

Nersessian, N. J., and Patton, C. 2009. “Model-based reasoning in interdisciplinary engineering.” In *Handbook of the Philosophy of Technology and Engineering Sciences,* edited by A. Meijers, 678-718. Springer.

Nersessian, N. J., and Chandrashekharan, S. 2009. “Hybrid analogies in conceptual innovation in science.” *Cognitive Systems Research Journal: Special Issue: Integrating Cognitive Abilities* 10: 178-88.

Osawa, E., Kroto, H. W., Fowler, P. W., and Wasserman, E. 1993. “The Evolution of the Football Structure for the C60 Molecule: A Retrospective [and Discussion]. ” *Philosophical Transactions: Physical Sciences and Engineering* 343: 1-8.

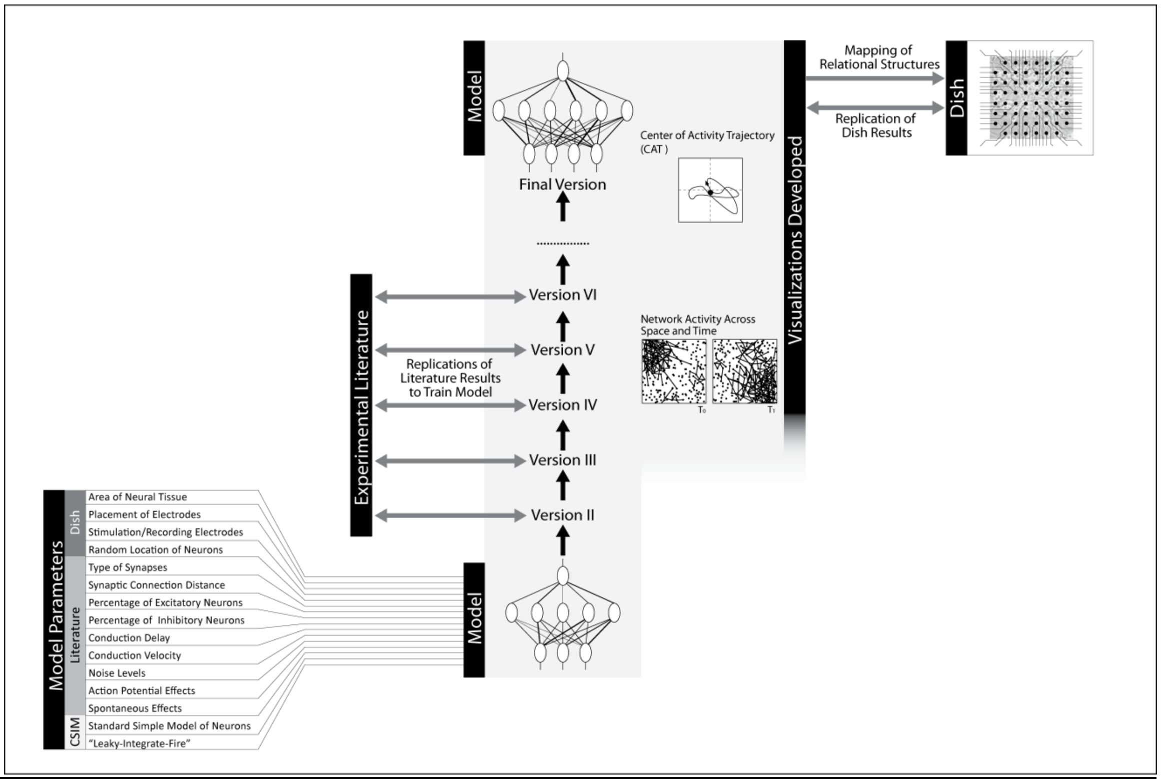
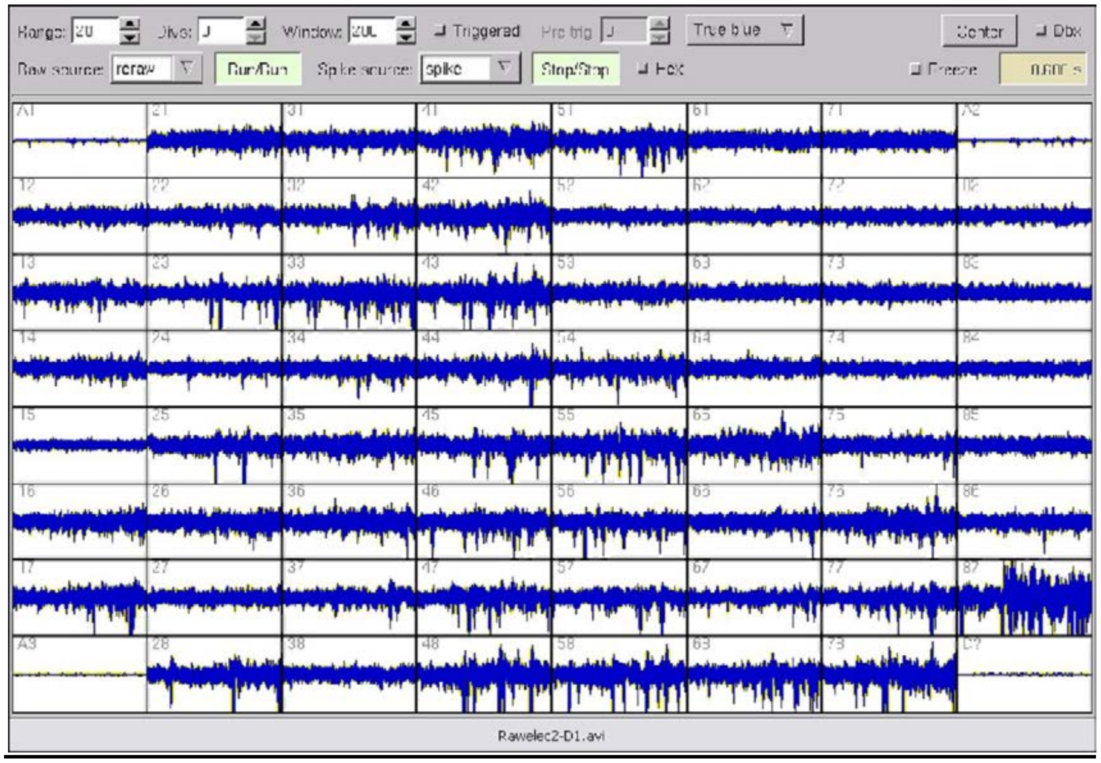

Model
Final Version
Center of Activity Trajectory (CAT)
Visualizations Developed
Mapping of Relational Structures
Replication of Dish Results
Dish
Network Activity Across Space and Time
T0
T1
Experimental Literature
Version VI
Version V
Replications of Literature Results to Train Model
Version IV
Version III
Version II
Model
Model Parameters
Dish
Literature
CSIM
Area of Neural Tissue
Placement of Electrodes
Stimulation/Recording Electrodes
Random Location of Neurons
Type of Synapses
Synaptic Connection Distance
Percentage of Excitatory Neurons
Percentage of Inhibitory Neurons
Conduction Delay
Conduction Velocity
Noise Levels
Action Potential Effects
Spontaneous Effects
Standard Simple Model of Neurons
"Leaky-Integrate-Fire"

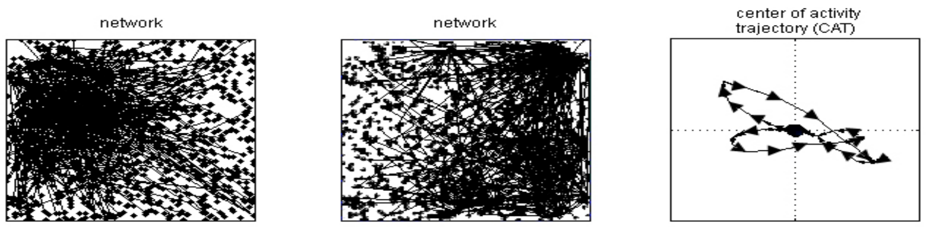
Figure 1


Range: 20 Divs: J Window: 20L Triggered Pre trig J True blue Center Dbx

Raw source: reraw Run/Run Spike source: spike Stop/Stop Hex Freeze 0.60F s

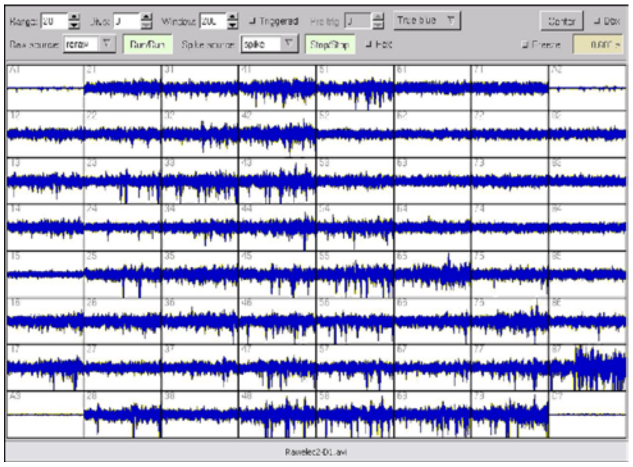
Rawelec2-D1.avi

Figure 2

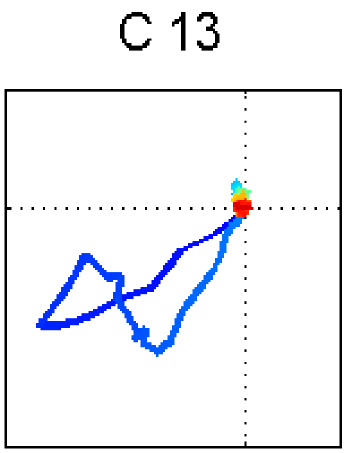


(a) (b) (c)

Figure 3



(a)



(b)

Figure 4

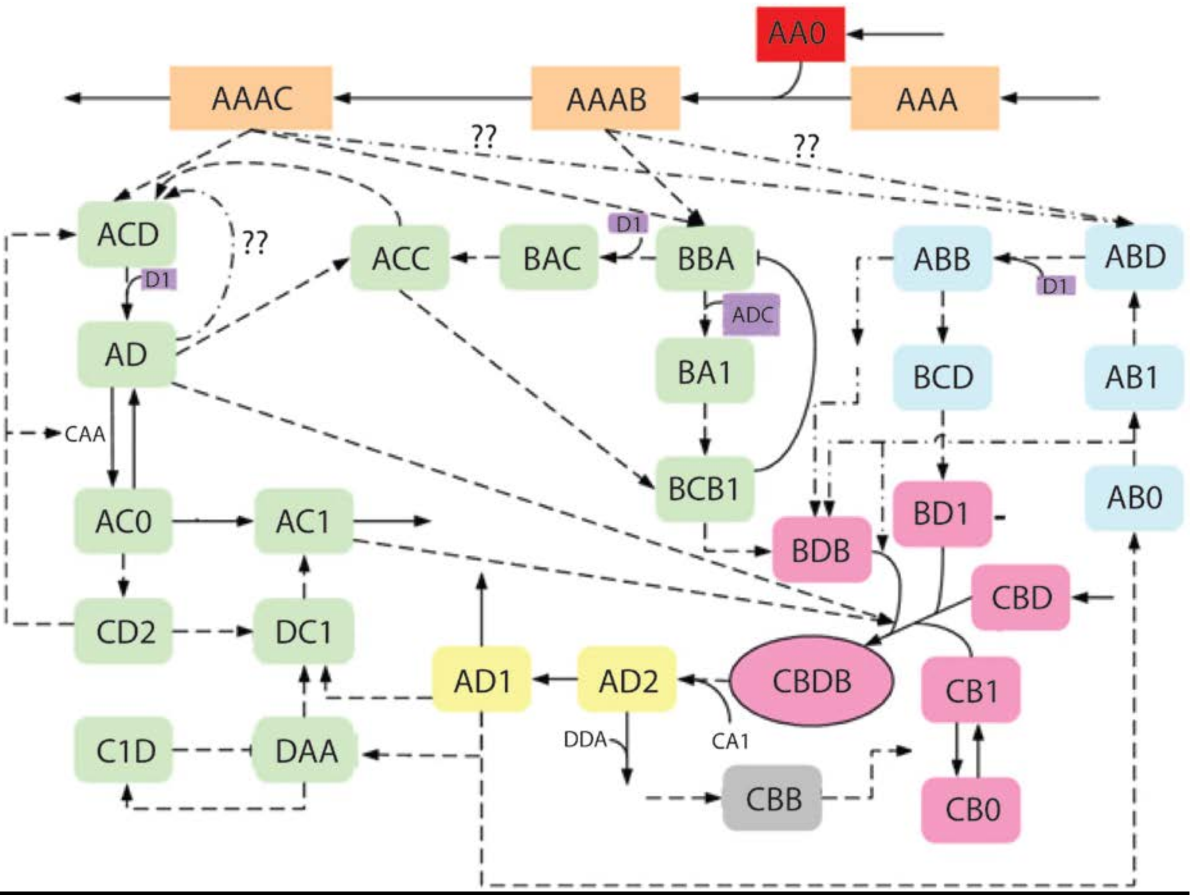

AA0
AAAC
AAAB
AAA
??
??
ACD
??
ACC
BAC
D1
BBA
ABB
ABD
D1
D1
ADC
AD
BA1
BCD
AB1
CAA
BCB1
AC0
AC1
BD1
AB0
BDB
CBD
CD2
DC1
AD1
AD2
CBDB
CB1
C1D
DAA
DDA
CA1
CBB
CB0

Figure 5

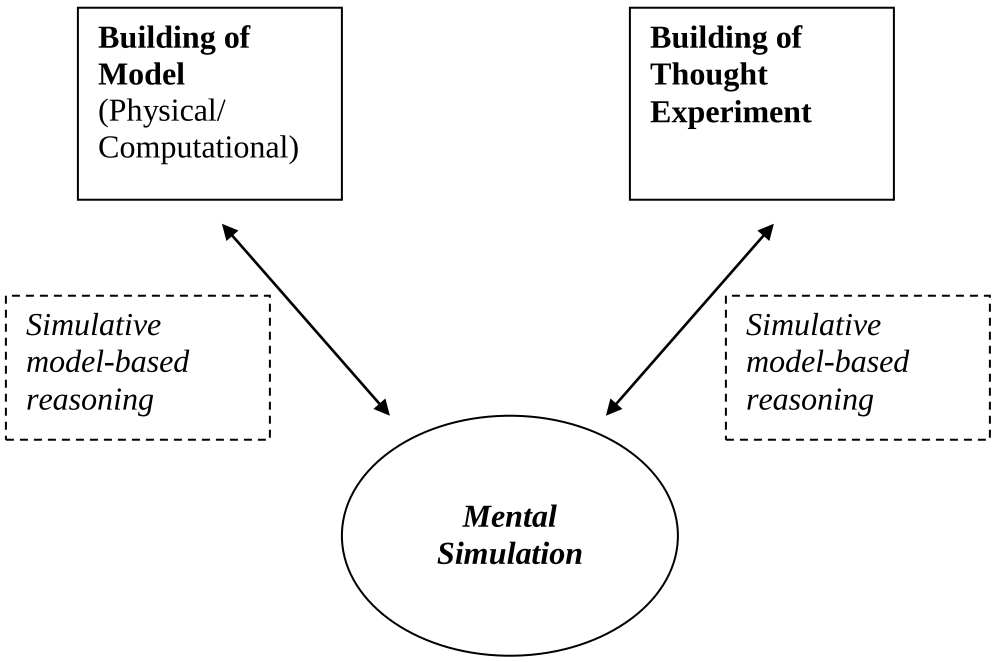

Building of Model (Physical/ Computational)
Building of Thought Experiment
Simulative model-based reasoning
Simulative model-based reasoning
Mental Simulation

Figure 6

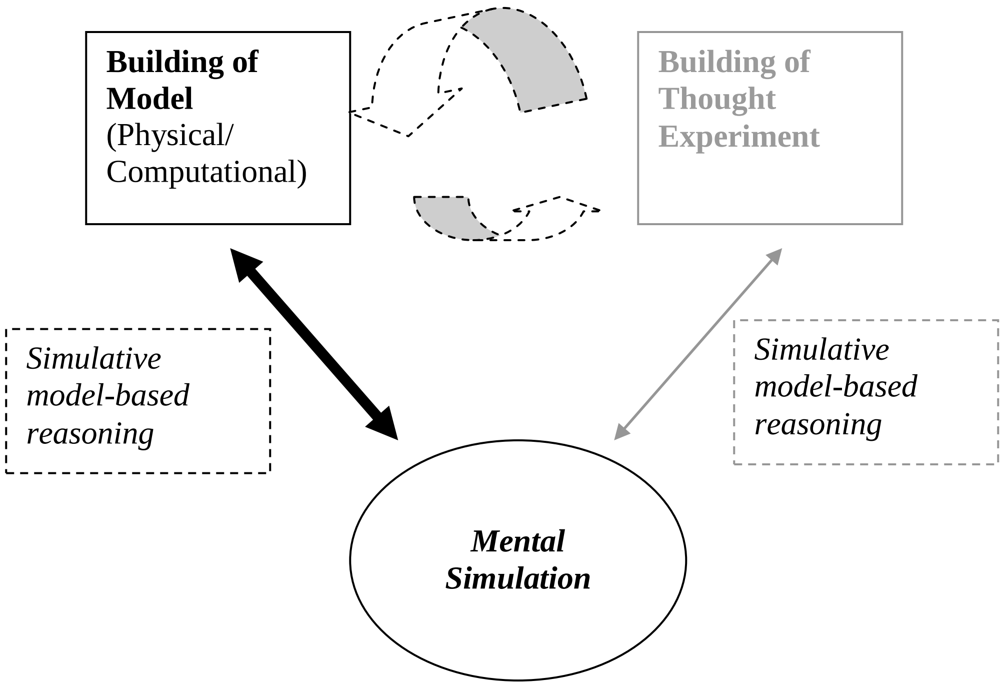

Building of Model (Physical/ Computational)
Building of Thought Experiment
Simulative model-based reasoning
Simulative model-based reasoning
Mental Simulation

Figure 7