Computational Modeling : Confessions of a Computer Scientist

Terry Jones
Santa Fe Institute
1660 Old Pecos Trail, Suite A
Santa Fe, NM 87505
terry@santafe.edu

May 10, 1994

1 Introduction

When I first arrived at SFI, literally overflowing with naive enthusiasm, I knew very little about modeling. I was to implement John Holland's Echo system for UNIX, C and X windows, thereby bringing the world out of the dark ages and ushering in a new era of enlightenment.

Now, almost two years later, it seems like a case of a fool rushing in where angels would fear to tread. My untrammeled enthusiasm seems at times to to have matured into a somewhat weary cynicism. My eagerness to jump on new problems has been replaced with an almost paralyzing reluctance to get involved with anything that doesn't seem to have every i dotted and every t crossed before the project has begun. In fact, my outlook is far from wholly negative, but I do have many serious concerns and questions about how one should proceed having made a decision to build a computer model.

I have discussed many of these ideas with members of the SFI community. Although much of what I write is the product of my own thinking and experience with Echo, almost all of it can be traced to hundreds of discussions at SFI, both with the more-or-less permanent members and with those visitors who have wandered (or been pushed) into my office. In addition, in March I had a chance to "interview" Marc Feldman, Chris Langton, Harold Morowitz and Alan Perelson about some of the topics below. I have tried to incorporate some of what we discussed in what follows.

2 Fundamentals And Assumptions

At the outset, I wanted to try to establish a common framework for discussion of my concerns. Although I will not attempt to define exactly what an "adaptive

agent based system" might be, I do want to be fairly explicit about the type of modeling I am concerned with.

The following premise is admittedly a narrow view of modeling, but it is simple to define, is representative of many modeling efforts, and at least provides a starting point for discussion. This simple premise is sufficient for me to discuss many concerns which will be among those that we should be addressing when we adopt a more sophisticated view of the modeling process.

I feel that I should stress the fact that the following discussion is intended to address perceived difficulties with a particular type of computer model. I am not concerned with cognitive models that we use to deal with the world, mathematical models used by physicists to explain the movement of the planets, or differential equations used by immunologists to describe the dynamics of the immune system. I know even less about these models than I do about computer models of systems with many agents, and would not presume to comment on them. A common humorous sentiment at SFI would deem that such ignorance is no reason not to write at least a few papers on the subject, but I shall try to refrain.

The view of modeling I will adopt in this article is the following. Some phenomenon, which I shall call $\mathcal{X}$, is observed in the real world. $\mathcal{X}$ is of interest, but is not well understood, and, in an effort to increase understanding about $\mathcal{X}$, a model of it, $M(\mathcal{X})$, is constructed. This simple view of a modeling process is more general than I need. My concerns apply to phenomena that involve large numbers of interactions between large numbers of agents, possibly over long periods of time. The particular types of $M(\mathcal{X})$ of concern are computer programs that attempt to model $\mathcal{X}$ via the construction of idealized agents within the computer, which are involved in idealized interactions in some idealized computational world.

A most important aspect of these assumptions is that the model is constructed *with the explicit aim of increasing our understanding of* $\mathcal{X}$. I am not concerned with models whose aim is otherwise.

Given this, I will take a further liberty, and make the assumption that $M(\mathcal{X})$ is constructed so as to be a *simplification* of $\mathcal{X}$. This seems like a reasonable step, as one is unlikely to (knowingly) construct a model which proves less open to understanding than the original phenomenon. This assumption does not say that a seemingly complex problem cannot be more easily solved via embedding it in an apparently even more complex framework whose solution can be obtained. This technique is very common in many fields[1]. The basis of the assumption is the word "understanding". That is, we construct $M(\mathcal{X})$, the simplification of $\mathcal{X}$, to be more understandable than $\mathcal{X}$. For now, I am deliberately ignoring the question of exactly what understanding might be. I will return to this subject.

In summary, we have $\mathcal{X}$, a model of it, $M(\mathcal{X})$, and the assumptions that the purpose of constructing $M(\mathcal{X})$ is to increase understanding, and that $M(\mathcal{X})$

[1] An example is the mathematician who knows how to make a cup of coffee from scratch, but when handed a kettle of boiling water, begins by pouring it out so as to create a problem whose solution is already known.

is a simplification of $\mathcal{X}$. All of which leads to a final assumption and begs an important question.

The assumption is that the success and worth of a model should be judged by the extent to which it succeeds in our stated goal of increasing our understanding of $\mathcal{X}$. If the model is very realistic and captures minute details of $\mathcal{X}$ but is, as a result, equally opaque, then the model, by these standards, should be judged a failure. Admittedly, this is a narrow-minded view of success. It would be a fantastic computational achievement to build a completely realistic virtual ecosystem (no matter how few the number of represented agents or species), but if it is as complex as the original, then we may gain less in the way of understanding than a simpler model might provide.

I am not claiming that complex computational models of complex systems will not, in general, increase understanding. I am merely establishing a yardstick by which to judge what we build. If I walk away from the model with the feeling that I haven't learned anything about the original system, then I will judge that model to be worthless.

The question begged by our assumption that model construction involves simplification is taken up the the next section.

3 Issues In Model Building

This section discusses one very important question, and then briefly touches on several other issues that should be considered.

The assumption that a model is a simplification raises the following question: What aspects of $\mathcal{X}$ should (at least initially) be omitted in the construction of $M(\mathcal{X})$? This, to me, is the fundamental question of model building. To maximize the potential for understanding the model you construct (and therefore, hopefully, for understanding the original phenomenon of interest), it seems reasonable to try to include in $M(\mathcal{X})$ those aspects of $\mathcal{X}$ (and no others) that are necessary to create those qualities of $\mathcal{X}$ that you were originally interested in understanding.

In a trivial sense, one could argue that successful model building is, in some cases, as difficult as understanding the original problem. If you need to know exactly which components of some complex system are important in order to build a successful model, then the process of identifying these components in order to construct a successful model may provide all the understanding of the original system that you initially desired. A similar situation occurs when evolutionary algorithms are constructed to search for objects with desirable properties. Here, one needs a so-called *fitness function* that can identify the most desirable objects, but the knowledge about the system necessary for the construction of such a function (in a computer) might be comparable to the knowledge needed to solve the original problem directly.

Having made the decision to build a model, one must address this question (what aspects of $\mathcal{X}$ to omit). Considering that the model might take a year to implement and cost thousands of dollars, these decisions are not something

that should be taken lightly. This is not to suggest that a model is necessarily something set in stone – far from it. In most cases models will be refined, enhanced, redesigned, and perhaps reimplemented. In an ideal world, one could repeat this cycle indefinitely, without cost, until the ideal model was constructed and its truth would be self-evident.

But in the more mundane world of academia, we are faced with issues of unreliable funding, publication pressure, migratory graduate student programmers, and, dare I say it, self-promotion. These have the unfortunate dual effects of making refinement, redesign and even abandonment far less likely and, simultaneously, hurrying into print the first sniff of anything that might be construed as results. In a self-perpetuating cycle, others involved in this brave new research paradigm hungrily gobble up these tasty tidbits, all the while applauding, and thus enhancing the credibility of their own endeavors.

Probably I am being too extreme. I believe what I have just written, but it does not preclude the existence of good modelers and good models. I have heard that good judgement comes from experience and that experience comes from bad judgement. If so, there is hope that inexperienced modelers (of which I am surely one) may become good modelers.

A final comment on the issue of what to omit and what to include is the following generalization. There seems to be a tradeoff between how faithful your model is to details of the real world and the extent to which you can use it to increase understanding. If you decide to model the dynamics of the trading floor of the New York Stock Exchange using a one-dimensional cellular automata with ten binary sites, you may construct a model whose dynamics you understand fairly well, but you're probably not going to come away with an increased understanding of the stock exchange. At the other extreme, if you choose to model the brain by constructing a model with 10^{13} completely realistic artificial neurons each simulated on a SPARC in a fantastically complicated network of 10^{10} CM5's, you may wind up with an exceptionally intelligent program, but you probably will not understand much about what a mind is or how the brain thinks. Of course, you'll have encyclopedic low-level knowledge of the brain, but you had that before you built the model.

Of course these examples are somewhat ridiculous, and could be dismissed if it weren't so easy to make choices whose effects will be similar. As with most of the issues I will raise, these are questions I have considered in the light of the Echo model. Echo is very simple compared to the fanciful brain model described above, yet I often feel the same way when trying to understand why the model behaves as it does. Even answering the simplest questions I can formulate about the behavior of Echo can take considerable legwork. Some people have pointed out, with justification, that this could be eased if I weren't so lazy and actually wrote some code to automate part of this search, instead of moping around bemoaning the situation.

The next issue I would like to briefly address concerns why one would want to build a model. I have met and talked with many people who are very taken with the idea of building a model of the kind of system they deal with. This interest is a good thing in general, and will undoubtedly push the development

of good models (via the development of bad ones). The field as a whole can be expected to profit. But what of the individuals who, through no fault of their own, will be the guinea pigs in this development? I think it is important, when considering the construction of a model, to ask oneself *exactly* what one hopes to learn from the model. What would be satisfactory? When can you stop? How will you know if you have succeeded or failed? How, exactly, will you construct the model? What will be in the model and what will not? These questions are very obvious, but I think they are neglected. The eagerness to build a model, the widespread availability of computers and programmers, and lack of experience combine to produce projects that are begun prematurely. In some cases, the best conclusion might be that one should *not* build a model, that valuable research dollars would be better directed elsewhere.

I am also very concerned about the interface between the expert and the programmer (assuming these two are not one and the same). If you are considering employing someone to build a model, and have not thought about how you will ever know whether your model contains bugs that effect its performance, you should. No programmer is perfect, and there are probably very few programs less than a year old of, say, ten thousand lines that are completely bug free. Assuming this is true, there is a high probability that if you write papers describing the results of such a model, you will not know what you're talking about. My Echo code (as opposed to John Holland's) is approximately fifteen thousand lines of C code, and is approximately eighteen months old. The last serious bug I found was probably about six months ago. Before that I had made numerous presentations, showed graphs and offered explanations of Echo phenomena that were all potentially badly flawed – it is not even clear in some cases to what extent the results were affected.

Unfortunately, problems like this are a fact of life in computer science. But by making sensible decisions in planning the construction of a model, they can be reduced. For example, each choice you make when deciding what should be in a model and what should not, directly affects what needs to be implemented and what does not. If you make a set of choices that require the construction of a very difficult computer program, you are simultaneously decreasing the probability that the model can be constructed without serious error. Not to mention making something that takes longer to write and is less understandable to and harder to maintain for the average programmer. Part of the high level decision-making process about what should be in a model should be informed by what each decision will mean at the programming level.

If you are in this position, you should also be aware that unless your model is very simple, the programmer will probably be faced with (and make) choices whose outcomes can dramatically alter the program's behavior. Some of these decisions will be trivial and may only affect, for example, running time and precision. Others will be less so. A subtle example is the choice of a random number generator – different generators have different characteristics. There are many tests for randomness, and some pass certain tests and fail others. This will produce different results even for generators that are considered reliable. The existence of incredibly poor generators is almost folkloric amongst computer

scientists, yet many people are unaware of this problem, and it is by no means a thing of the past. Two years ago I was shown a random sequence produced by a leading personal computer company – it generated about thirty different numbers before beginning the same cycle, which it then produced indefinitely!

As a not-so-subtle example, when implementing a population of Echo agents at a site, I used a variable length array representation. When an agent self-reproduced, I added the new copy of the agent to the end of the array. This was simplicity in itself and was of course very fast. During a presentation, Bob Axelrod asked a question that made me realize that I had implicitly made an important assumption. Surely the new agent should be placed next to its parent? I at once implemented this, completely at a loss as to how I had overlooked it in the first place. Later I considered that this was perhaps not the way it should be, and that some situations would be better modeled under the first scenario. I do know that the choice has a very great effect on the dynamics of Echo. This example is still somewhat subtle as it represents an error that was made when I failed to realize that I had a choice.

When you realize that a choice needs to be made, it is often the case that one of the possibilities will appear to "push" the system in a certain direction. If one is interested in seeing the emergence of some phenomenon, it is important to know to what extent what you get is a result of how far you pushed the system in that direction. Some decisions that will influence the extent to which the system is so pushed will be made by the programmer – quite possibly unbeknownst to anyone.

Lastly, it is important to consider what sort of information you want to extract from your model, and how you want to display it. My experience has been that one of the hardest things is knowing how to display information effectively. Getting it is a minor annoyance compared to the task of displaying it in a way that allows one to see at a glance some interesting behavior. In addition to seeing interesting behavior, it is helpful if you can explain it – at least in terms of the model. More on that below.

4 Using Your New Model

Having built a model, and convinced yourself that it is more or less a correct implementation, it would be a shame not to take it for a test drive. You start the program, create some agents and let the system run for some time. Perhaps you have colorful graphics which displays how some features of your model are unfolding. What happens next?

Two of the main uses of models are explanation and prediction. In the first, we want to know why a certain system (either $\mathcal{X}$ or $M(\mathcal{X})$) behaved in some way. We want to be able to explain past events such as stock market crashes, extinctions, oscillations, migratory patterns, the persistence of structures and so on. In the second, we would like to tackle the (usually) harder problem of using $M(\mathcal{X})$ to make predictions about $\mathcal{X}$. Not content with why the stock market crashed, we'd like to know when the next crash will take place.

My experience has taught me that both types of questions are very hard to answer. Let's start with explanation and, of course, an example from experiences with Echo. A relatively simple Echo world that we have experimented with contains one site with three types of agents, which we call ants, flies and caterpillars. These are idealized versions of those animals and, at least initially, the Echo world contains an idealized triangular relationship among them. Looking at many runs of the same world, it is common to see one agent doing well and for this to be followed by a reversal of fates wherein the agent that was doing well becomes low in numbers and some other agent's numbers grow quickly until it is as successful as the original agent was.

A seemingly simple question is: Why? Why did the flies (say) suddenly fall rapidly in numbers and why did the ants (say) begin to do so well? There was a time when, if asked a question like this, I would try to explain what had happened. It is reasonably simple to come up with an entirely plausible hypothesis that nicely explains the observed dynamics. Unfortunately, when one looks a little closer, the hypothesis is very often proved completely false. After briefly scratching my head, I could deliver a second even more plausible account of what had transpired. Confirming that this was also incorrect would take a little longer and require deeper digging into the system. Eventually, after perhaps eight hours, I would have ten hypotheses, every one of which sounded perfectly reasonable and every one of which was absolutely wrong. Generating new hypotheses becomes increasingly hard.

This brings me back to a point on visualization. There can be no doubt that pictorial representations can instantly show structure that could not be detected in other representations. I am a great believer in tools for visualization (not that I have written any particularly good ones – let's say that I believe in it in theory), but visual presentations will naturally lead to interpretation and hypothesis formation. The hypotheses so formulated could easily be wrong. I agree that a picture paints a thousand words, but in our case, it would be nice to know what the actual words are.

If it is so hard to explain the performance of an Echo run, what hope have we for explaining events in the systems that it is intended to model? The model is vastly idealized, we have perfect knowledge of every interaction in the system, we can test hypotheses, we can stop the run, we can run it backwards, we can introduce or remove agents, edit their genomes, undo mutations, play with fifty parameters, and yet it takes hours to answer practically the simplest question one could ask. What is wrong here? Are we asking the wrong questions? I think the answer to this question is possibly "Yes."

On the few occasions when I have traced every detail of an Echo run to get an answer to a question such as this, the result has always been a set of chance events that slightly altered the probability of some set of potential interactions. For example, a mutant ant is created that has a 0.8 probability of beating a "normal" fly in combat rather than the 0.6 probability a "normal" ant would have. Or a mutation in the trading gene of an agent allows it to trade with a copy of itself. As a result, over a relatively long period, this agent does well and eventually creates an offspring that is exceedingly lucky to survive but which

leaves an offspring that takes over the world (after itself passing through the eye of a needle).

This is the kind of explanation I can give, and it's not very satisfying to that part of our cultural heritage which would prefer to hear simple and elegant high-level explanations that are typically among the first hypotheses to be rejected when one starts digging. We want to hear that the ants drove the flies to extinction as a result of some instantaneous and ingenious evolutionary adaptation. The truth, at least in Echo, which I guess will not surprise any biologist, is that a collection of tiny accidents and chance encounters combined to produce what we saw. Moreover, this set of events is simply one among countless other sets that could have transpired.

As I said, this observation about Echo is not too surprising to some people, but it does clash with our societal desire for the nice causal explanations that allow us to deal with the world so successfully. A simple explanation of what happened does not exist. The only point of all this is that we tend to look for and expect simple answers to simple questions. If your aims in model building are to uncover pithy truths, you may come away disappointed.

All of which leads me to the second hoped-for use of modeling: prediction. Given the above description of the search for reason in a single time series, it should be apparent that prediction in such highly stochastic models as going to be difficult at best. The extent to which they will allow us to actually complete the modeling loop and make predictions about the real world is another question altogether. The further we try to extend such predictions (time-wise) the less likely they are to eventuate.

The apparent solution to this problem is to aim to make probabilistic statements. Having established that the model is a reasonable reflection of the original system, a large number of runs might indicate the probability that a certain phenomenon will be observed. This is a far more modest aim, but if even it could be achieved, it would be very important and useful in real systems. This kind of approach could also be used to develop a feel for the sensitivity of various parameters in the system. These aims are common to many of the people involved in modeling efforts at SFI.

5 Conclusions

This section contains some high level thoughts about modeling, explains my position (which is not as negative as some think) and generally tries to end on a happy and optimistic note.

Firstly, I think it is very important to question the extent to which we can regard modeling as a scientific process. Harold Morowitz has raised some of these issues with me on a couple of occasions. He has talked about such things as falsifiability (e.g. on what grounds should you throw a model out) and scientific method in general. I would like to think that we will see the development of something akin to the scientific method employed in other fields. This would give us (at least) four important things.

Firstly, it would instruct the modeler. The standards in other scientific disciplines lay down rules that must be followed. For example a fundamental idea is that of control experiments. Children learn about this in high school and beyond. We are taught to recognize a well controlled experiment, and when we do our own experiments, we understand the need for a control and, more generally, for sticking with accepted scientific methods.

Secondly, if such methods existed and were widely accepted and in widespread use, it would allow people to have confidence in each other's work. At the moment I don't even trust myself, and it consequently feels like an enormous leap of faith to read a paper or attend a seminar on modeling and to accept it at face value. I cannot see any reason why I should particularly believe anything that is said to me about the properties of some complicated model.

Thirdly, if a more-or-less rigorous scientific method for modeling existed, we would make progress towards repeatability. It is a common and important practice in science to repeat experiments performed in one laboratory in a laboratory elsewhere. I don't think anyone would deny this. However, in the field of computational modeling of adaptive agent based systems, repeatability is the exception rather than the norm. The lack of rigor in our field can be seen for example in evolutionary algorithms. It is very common for one person to carry out some research and for another to diligently attempt to reproduce it and obtain different results. This is an ongoing theme in genetic algorithms, and is the source of seemingly never-ending debates about the virtues and necessities of various components of those algorithms.

Fourthly, we would perhaps, in time, be led to a theory of falsifiability. There is currently no way to decide between whether a model is completely wrong, partially wrong or incredibly close to being right, but with a single variable critically wrong. So we tinker and play with what we have, adrift in an infinite space of possibilities – without even any tools for saying where we have been or what we were close to. It is simpler to tinker a little more with a model that has received some number of years of development, funding, attention and expectation than to discard it completely. The academic world does not reward those sort of "results." Fortunately, SFI is relatively free from many of these pressures, making it an ideal place for work that may involve such conclusions.

These methods do not, as yet, exist. We resemble the "scientists" of centuries gone by. They were determined to be scientists (even before the word existed), but had no idea how to construct a rigorous experiment. In many cases, they were completely ignorant of causes, but nevertheless were happy to propose grandly named theories of just about everything. The use of normative language in describing experiments and naming principles and theories then, as now, is a continual source of amazement to me.

I think it is important to recognize that model building is itself an evolutionary process. This is especially true, as mentioned above, in cases where getting the model right can be as difficult as understanding the original problem. If model building is, as Alan suggests, an art, then we should expect mistakes and false starts. I believe that Echo has several shortcomings (a couple are removed in John's later model) that could or should be removed. These would cost the

system in generality, but would provide for less opaque resultant models. The important thing about making mistakes, if that is what has been done, is that we have learned many things along the way, and that we apply them. I know I certainly have and will.

George Cowan once spoke to me after I had given a presentation where I was probably being somewhat negative about modeling. He told me that he had seen many examples of bad models that had been presented, that everyone knew they were wrong, but that they were still useful as they made people think, and argue and eventually construct better models.

Stephanie Forrest pointed out to me that we shouldn't expect every single model to be wildly successful and that a collection of models, each of which was not particularly useful, might nevertheless produce a great effect. The best example of such an effect is probably in chess playing programs of artificial intelligence. The incremental steps that have been taken towards producing better and better chess machines have typically been interesting only as examples of faster computation, bigger lookup tables, slightly deeper search, expensive hardware, occasional clever algorithms and speed-ups and so on. There is nothing that I would consider particularly relevant to intelligence in these programs. Yet they have wrought an enormous change. Forty years ago, literally no-one thought a computer would ever play good chess, that was a task which definitely required intelligence (whatever that was). Today, we have conveniently adjusted our impression of just what intelligence might be. You no longer have to be intelligent to be a grand master, though it probably helps. It seems clear from this example that collections of models might do things that none of them can achieve individually. However this is small solace to the individual model builder.

There are many ways to view modeling and many ways to use and envision using models. I am beginning to suspect that everyone's viewpoint is different. The premise on which I based much of the preceding discussion is a very narrow one. Chris Langton, for example, surprised me when talking about models and modeling. His ideas seem distant cousins to mine, and he doesn't seem to suffer from the same anxieties, perhaps because his goals and expectations are more realistic. For example, he said "I think the point of modeling is to generate questions, not to generate answers. Ultimately, we want those questions to give answers, but right now we're trying to figure out the right questions to ask." Chris is interested, amongst other things, in generating a wide range of models of a particular $\mathcal{X}$ and asking what these models have in common. These views and interests lie far outside the framework under which I built the above discussion, though I believe that many of the same issues will arise.

There is a large literature that concerns itself with the construction of models and with modeling in general. There is currently a very active discussion of some of these issues in relation to the 2050 project. I make no claim to any deep understanding of the modeling field, or to any great experience. In fact, I am far happier to claim exactly the opposite (which is much closer to the truth), that I know next to nothing of the details of all this work. This ignorance can be excused I hope. I am merely writing down things that have occurred to me,

things that others will undoubtedly arrive at SFI without having considered, things that I would be happy to be educated about, and, questions that I feel are incredibly important given the nature of what we do at SFI. It is also important to me that SFI not be seen as the kind of place where people blindly create fantastic models without reason or thought.

Here are some final suggestions to those who may be embarking on the modeling journey.

- Build the simplest model you can. If it proves to be too simple, that's great. Deliberately design a model whose behavior you will have some reasonable hope of understanding. While making these decisions do not ignore what they imply for the implementation of the model, and for how you hope to be able to examine the results of running the program.
- Introduce as few variables as you can. Three may well be too many.
- Try to decide exactly what you want to do *before* you try to do it. What, exactly, is the question you wish to answer? How will you know when you are done?
- Try to design experiments that have some semblance of a scientific experiment. Can you construct something falsifiable? Can you design good controls? Will your model be simple enough that someone else will be able to repeat your work easily?
- Report successes, failures and parameter sensitivities.
- Use sensible language when describing your model – for example, don't call something a species unless you are *sure* you know what a species is. Don't say that your agents have "discovered" or "learned" or "evolved" tool usage when you filled your world with hammers and nails and provided your agents with opposable thumbs and convenient subroutines called PICK_UP_OBJECT() and HIT_OBJECT().