NICE TO KNOW

Using Models Designed by Other People

Tony Starfield

recorded: 2005

I sometimes phone calls from people who are working on a model, and they ask me if I would be willing to help them design it or develop it. I know that as soon as I've said yes, within a day or two, I'm going to get another phone call. And the phone call conversation will go something like this, “Remember we asked you to help us design a model? Well, we were speaking to somebody the other day, and they said, why don't you use this code? Or, why don't you use that code?” In other words, use somebody else's model instead of developing your own. Let's suppose this code was called ‘Magic’. They all have catchy names. So I get this phone call that says, "This is what happened. What do you think?" "Do you think we should use Magic?"

Of course, I don't know the details of these models. So the response that I've developed is one of trying to help people think through for themselves whether or not it is useful to use somebody else's model. And to think this through, we really need to go back to basics.

Let's look at our real world model world diagram. When you are going to build a model, you have to design your model world. Remember, a model is a purposeful tool for solving a problem. Therefore, your objectives in the real world are going to determine what you put in and leave out of your model world.

Now, you need to do that for yourself. But whoever has developed Magic has presumably done it for themselves. So the first thing you need to do is to compare your model world with their model world.

So if we are going to look at a process of deciding whether or not you use somebody else's model, the very first step is **compare the two model worlds**.

Well, it might be that you don't know what their model world is. Perhaps it hasn't been spelled out carefully. If that's the case, the only advice I can give is run. You don't want to have anything to do with any model where you don't fully understand what the model world is. However, if they have described their model, let's think of some of the key things you need to look at in comparing your model world with their model world.

The first point, of course, would be objectives. What are the objectives of your model? What are the objectives of their model? It could be that the objectives are different and you can still use their model, but if the objectives are not the same, that is a heads-up to be wary.

The next thing you would want to do is to check spatial and temporal scales. A model built at a particular spatial or temporal scale might not see things that you need to see at a different scale. It might be too detailed or it might be too broad.

And then you need to check assumptions. You need to be sure of what assumptions you are willing to make. And you need to check to see what assumptions they have made. And remember, maybe they haven't spelled out all of their assumptions.

Suppose, though, we pass this test. Suppose there is sufficient overlap between your model world and their model world for you to seriously consider using the model.

The next step, then, would be to **consider various operational issues**. Now, what do I mean by operational issues? The first would be, is the model easy to use? If it is difficult to use, that might put you off. Is it easy to import and export files? What are the input requirements? Perhaps more importantly, what kind of output does it produce? Maybe it is a stochastic model and the output is averages and variance. Whereas what you really want is probability of being above or below a threshold. If that's the case, can you go into the code and adjust it easily? You might have the perfect model but it might not give you the output you want. Does it provide options? Does it explain how to use those options? Does it have routines or options which you might want to omit? For example, it might go into great detail in a particular part of the model which is irrelevant for your purposes. You cannot take the shortcut of saying, "Oh, well, I don't mind it if it does that detail. I'll just give it some input values and let it do its thing." I am never happy using a computer model which is doing its thing and I don't understand what its thing is. So if it has these routines, I want to have the ability to cripple those routines that I don't want to use.

Suppose we pass all these operational issues. Then what's the next step? Well, now you're getting serious about using the model. That means you need to **test it thoroughly**. What do I mean by testing it thoroughly? There are a number of things one could do.

For a start, I would perhaps build my own first prototype spreadsheet model, just a very simple model. And compare results with the code just to see that the two models are in the right ballpark. Then I would design some simple test runs of the model, runs where I can anticipate the kind of output I would likely get. I would try to break the model. I would think of its limits of operation and see how well it performs as it hits the limits of operations. I would, most importantly of all, try to do a thorough sensitivity analysis. There is nothing like doing a thorough sensitivity analysis of a model, to be able to understand the model, and to know what it can and can't do. It's the best way to understand its performance.

Okay. So if we pass all of those three steps, then we are ready to use somebody else's model. But we've done most of the modeling work. There's no shortcut to using somebody else's model. If you haven't thought through carefully what you are doing, and now you have the additional job of thinking through and understanding carefully what they were doing, if you

haven't done that, you are using that model at your own peril. So there's really more work in getting there than in just designing your own model. So what do you save? Why would you want to do this?

Well, perhaps the most important thing you save is you don't have to do the coding. And there are really good advantages to not having to do the coding. Because if this is a code that other people have used, the chances are that the bugs have been ironed out, and that is a major advantage. If you watch the "Antiques Roadshow" you would say that the model has ‘provenance’.

The other thing it has is brand recognition, in the sense that when you write a paper or write a report and say ‘I used Magic’, then everybody who has ever used Magic will know what you were doing. It makes it easier to describe your modeling effort. And the other thing that is not to be underrated is that it might have some really neat output. And save you a lot of hassles in terms of preparing your output.

So there are good reasons for using other people's model. But those good reasons can never outperform the possibility that you're using the wrong model. They can never trump the idea that you are using the wrong model. So however beautiful Magic happens to be, you have to be very, very sure that it is the model that you want to use.

< 00:09:31 END >