Rapid Prototyping Applied to Modeling

Tony Starfield

recorded: March, 2011

We ended the previous module on a bit of a down note because we had a long list of things that can go wrong in modeling. You might remember that list was buffeted by the two cardinal sins, starting off with a vague objective and ending up by misinterpreting your results because you confuse the **Real World** with the **Model World**.

What we're going to do in this section -- and I'm glad you've hung in because a lot of people give up on modeling because of all the things that can go wrong. What we're going to do in this section is try and show you how things could go right. What can you do to limit the chance that those things can go wrong?

If we look at the list, the first problem is not having a clear objective. And very often, people can put a lot of work into developing a model, which is, in fact, the wrong model. So the first way we can go about trying to correct that problem is by doing something that I call 'rapid prototyping'.

Rapid prototyping is the idea that you try to get an answer to your problem as quickly as possible, not in the sense of building a perfect model but in the sense of doing something quickly and then improving on it.

So if you go back to the diagram you have that represents the real world and the model world and is, essentially, a model of a model. The idea in rapid prototyping is to think of that rather like a race track. And the idea is to start in the real world and shoot around the race track as quickly as possible to try and get back to the real world.

Now how does that help you if you've got a vague objective? Well by taking whatever objective you have and developing a simple model, when you get back to the real world and present some results from the simple model, it's going to be pretty obvious if you're off in the wrong direction. People are going to say, "That's not quite what we wanted," or, "Your model tells us that we weren't quite thinking about this problem properly, and maybe we need to change our objective." So a rapid prototype helps you to develop a quick solution to the problem, which might indicate that, in fact, you're solving the wrong problem.

Now, again, if you look at your diagram, the next step in going from the real world to the model world is to design your model world. And the problem here is that people tend to put in excessive detail.

Now when people are designing a model world to build a perfect model, there's this pressure always to put in more detail in the model rather than less detail. People are terrified that they might leave something out. So the kind of heuristic that they often use is, "If in doubt, put it in." We don't want somebody to blame us because we left it out.

But, in fact, if you're developing a rapid prototype, you have the opportunity to turn that on its side, upside down and say, "If in doubt, leave it out." Leave it out because we can always decide to put it in later in a second prototype or even a third prototype. So if there's any doubt at all about putting something into the model world, don't put it in, but make a list of what you've left out. In a sense, you're making a **list of assumptions**, which you can review later.

Okay. So suppose now you've designed your minimalist model world. The next step is to design your model, or develop your model. And here, the problem was what we call **complex mathematics**. Not in the sense of having anything against mathematicians, but there's just the danger that one jumps into some complicated mathematics, which, again, might not really fit the problem that you're trying to solve. So the advantage of doing a rapid prototype is if somebody comes along and says, "Let's use differential equations or partial differential equations," you can say, "Maybe we'll use that later, but for our purposes here, let's just do something on a spreadsheet."

The other problem, the next on the list is **data deficiency**. You don't have the data you want. Well the beauty about a rapid prototype is you can guess at the data because you can say, "This is just a prototype. Let's guess at the data, and let's see how it works." And in fact, if you know that your data are within specific limits, you could say, "Let's guess low. Let's guess high." And we'll come back to that later.

The next item in the list is **too much expert input**. And the idea there, perhaps, is it covers some of the things we've talked about like excessive detail and complex mathematics. But somebody wants you to use some fancy computer code. And, again, if you're doing a rapid prototype, you can say, "Keep it simple. Let's do something on an Excel spreadsheet." It might be slow. It might be inadequate for the actual model, but we can always come back and code it in something better later.

And then, finally, **confusing the model and real world** is absolutely solved by doing a rapid prototype because you are aware that this is just a prototype. You know that it is a simple model, and so you're going to think really hard about how to interpret it when you get back to the real world.

So that's rapid prototyping. The idea is that you start with the current objectives. You build a model as simply as possible. If in doubt, you leave things out, but keep a list. And, eventually, you come around to some solutions. And notice how this involves the people who -- stakeholders, perhaps, or managers that you're working with. Because instead of going away and coming back six months later with your model, you are going to try and develop a prototype within a few days or a week. And you're going to come back at them and keep them involved in what you're doing.

But rapid prototyping, by itself, isn't sufficient. Because, in a sense, when you develop a rapid prototype, you are building a castle in the air. You're taking a problem. You're stripping it down to its essentials, and you're saying, "Okay, that's our model." But is it a good model? Is it appropriate? How do you improve it? And how do you get better data when you need better data?

And to resolve those issues, we need to do two more things, other than just a rapid prototype.

And the first of those is to do a thorough **sensitivity analysis** of our simple model. Every single parameter that you've put into your model, you're going to ask the question, "What difference does it make"? And not necessarily just "What difference does it make to the answer? Is it going to change the answer by 2 percent or 4 percent or 10 percent." But if the problem is a problem in the context of making a decision, you're going to ask, "What difference does it make to the decision?" Do you really need to know that number, precisely, in order to make a good decision? Or, to look at that in a slightly different way, does the uncertainty in that data make any difference with respect to the decision that we tried to make? So a thorough sensitivity analysis goes hand-in-hand with rapid prototyping in helping you to determine which of your data you need to improve for your next prototype.

And, finally, the third thing you need to do is a thorough **assumption analysis**. Now some people think of an assumption analysis as part of a sensitivity analysis. But I've coined the words "assumption analysis" because I want people to think hard about that part of what they're doing.

And, remember, when you were designing your very, very simple model world, you were keeping a list of assumptions. And it might be very useful to go through that list of assumptions and prioritize it so that you can say, “I think this is the most serious assumption. I think that is the second-most serious.” And then one by one, you need to think very creatively about what difference that assumption might make to your answer without developing a much more complicated model. In other words, you need to say, “What might be the worst case or the best case”?

Well what are some examples of assumptions you might make? If you’ve been doing any economic analysis, maybe you made the assumption that house prices never go down. Well it would be really useful to say, “What difference would it make if we allowed house prices to rise and then drop?” And let’s put that into our model. It’s almost like doing a different run of the model with a slightly different numbers in it and see what difference that makes.

Or maybe one of your assumptions is that you’ve ignored the possible effect of climate change. So you could stop and say, “Where, in our model would climate change have the maximum input, and how could we represent that very, very simply, again, to see what difference it makes”?

So what does assumption analysis do for you? It really tells you how to design your next prototype, because it tells you which of the items that you left out of your model world you need to put back into your model world. And so the trio of rapid prototyping, sensitivity analysis and assumption analysis is a very, very powerful way to deal with the problems that one has with all the things that can go wrong in modeling.

If you think about it, you’re really setting yourself up for a win-win situation. When you do your sensitivity analysis, if your data are inadequate, your sensitivity analysis is going to tell you a lot about what you really need for those data. So, for example, instead of saying, “I need to get that number precisely,” you might know, from your sensitivity analysis, what really matters is whether that number is bigger than three or less than three, for example.

On the other hand, if the sensitivity analysis makes no difference to your decision or no appreciable difference to your model, you know you don’t have to go and collect those data. Same with the assumption analysis. If people are clamoring for you to make your model more complicated, your assumption analysis shows them where you do need to make it more

complicated or why you don't need to put that into your model in order to give them a good answer.

I sometimes feel I'm cheating a bit when I talk about rapid prototyping. Because if I'm, for example, trying to develop a model with a group of stakeholders, I might very politely say to somebody - an expert in the group, "What you've told me is very interesting, and it could be very important, but remember, this is just the first prototype. So I would prefer to leave it out in the first prototype. Let's put it on our list, and we'll come back to it later." When I say that, my experience tells me that very often, you don't need to come back to a second or third prototype. Because the first prototype gives you enough of a sense of what you need to do to move on to some different aspect of the problem or, perhaps, to give management or a group of stakeholders the answers they need. You very seldom need to go beyond a second prototype.

To sum up then, use rapid prototyping as the first step. And this is a good heuristic, not just in modeling. There are lots of situations in the real world where you're doing other things than modeling where rapid prototyping can be important.

I once had an insight from a student in a class who said, "I've realized that I use rapid prototyping just about everywhere. If I'm asked to write an essay, I do a rapid prototype. I've never thought of using rapid prototyping in modeling."

Rapid prototyping, sensitivity analysis--and notice, because you've kept your model simple in the rapid prototype, it's much, much easier to do that sensitivity analysis--and then, finally, assumption analysis to tell you where you need to make your model more complicated.

< 00:15:11 END >