EXPERT SYSTEMS AND KNOWLEDGE BASE - PART 1

TONY STARFIELD: We have talked a lot about trying to develop mind tools. Modeling tools that help you to think through problems and help you to communicate the way you think through problems. And in this segment, we're going to talk about another mind tool that is just a little bit different. It is something called an "expert system." It is more of an operational tool in the sense of telling people how to use information than it is a modeling tool in the sense of the dynamics of a system. But, nevertheless, as I hope you're going to see, it can be a potentially very useful mind tool.

Let me tell you a little bit about the history of expert systems. Way back when computer science, as a subject, was very young, and I was very young too, the really exciting part of any computer science conference was the section on artificial intelligence. Artificial intelligence was where all the zany ideas were presented. And the basic idea behind artificial intelligence was, "Could a computer perform a human activity as well as a human?" And the acid test would be that if you didn't know whether you were interacting with a computer or a human being, then that showed you that the computer was acting as well as a human. And this is where some of the original chess-playing ide-

as developed, and it's where robotics developed.

But somewhere along the line in that chain of thought was the hypothesis that a human being was using his or her brain to the maximum when they were acting as an expert in a particular discipline. And so the people working in artificial intelligence came up with the idea of, "Can we capture, in a computer, the expertise of an expert and deliver it to others in such a way that the person interacting with the computer wouldn't know that they were dealing with a computer?" The computer would be as good as, or almost as good as, the expert.

And this was a pretty esoteric branch of artificial intelligence until sometime in the early 1980s some people at Stanford figured out that there was potentially a commercial application for this. And so one had a whole lot of dot-com-type companies booming to deliver expert systems and develop expert systems. And there was a boom, and then there was a bust. And, as always with developments in computer science, the boom followed by the bust leaves a residue of something that's really useful. And you are probably interacting with expert systems or people using expert systems all the time now without realizing it.

Okay. So the idea of a practical expert system is something that

can deliver expertise to a user that is really useful to the user. So how does this work?

I want you to imagine your computer here. And what you do is you load into your computer something called an "expert system shell." Now what is a shell? Think, for example, of when you double-click on Excel because you want to develop a spreadsheet. As soon as you've loaded Excel, you have a shell in your computer. An Excel shell is one that is capable of doing certain things, like arithmetic and developing formulae and so on. That shell has got nothing to perform on until you either feed into it a spreadsheet model that you had developed previously, or you start a new spreadsheet model from scratch.

An expert system shell is similar to that. It provides the computer with a capability to communicate with a user in some kind of way to absorb the results of that communication, to process some logic, and to come back to the user with advice. But it can't do anything, any of that stuff at all, until it actually has an example developed in the shell or fed into the shell.

And just as you would feed in an Excel model into an Excel spreadsheet, so you feed into an expert system shell something called a "knowledge base." And that knowledge base is written in

the style that is appropriate to that shell and, basically, captures the knowledge that you want to deliver to the user.

So what do we have so far? We have your computer. You fed in a shell, which is kind of like an empty brain. And that empty brain is dying to know something, so you feed in a knowledge base on a particular subject, say, a knowledge base on diagnosing tropical diseases. And now what you have in your computer is that expertise on tropical diseases, ready to deliver to somebody who wants that information.

So the next step, once the knowledge base is loaded, along comes a user, perhaps a doctor who has a patient--a doctor in the states who has a patient who has been traveling in Africa and has come back ill, and that doctor doesn't know much about tropical diseases and needs a little bit of help. So the user comes along and says, "Help. Give me advice." And what the computer then does is to ask questions and provide a limited set of answers to each of the questions. The user replies to the questions; in other words, chooses one of the answers. And the computer then decides what to ask next on the basis of what it already knows from the user. And so it has this conversation with the user, and, eventually, at the end, out comes a decision. The computer says, "Your patient has malaria."

Now if you think about it, that is not a good interaction between a human being and a computer. Remember, in all of modeling, a computer is there to help you think. It is not there to think for you. So if I were a doctor, I would be very dissatisfied if I had an interaction with a computer program that eventually led to a decision, and I didn't understand how the computer got to that decision. I might not know a lot about tropical diseases, but I should be able to know enough about medicine to understand why the computer came up with a decision about malaria. So what is missing from that interaction is an explanation feature on behalf of the computer.

So what would be a really good interaction? The user comes along and says, "Help. I need advice." The computer asks a question, and the user says, "Why are you asking me that question?" And the computer has an explanation feature. It gives you a little bit of background on the question, and it tries to explain the way it's thinking. It's trying to say, "I'm asking this question because I'm thinking that--I'm trying to test whether your patient has something horrible, like Ebola. And this is the best way for me to figure out whether or not that might be the case."

And so you go back and forth answering the questions, and the computer asks a question. You ask, "Why?" You answer. And, even-

tually, the computer comes out with a decision. And at that point, you again say, "Why?" and the computer tries to tell you why it reached that particular conclusion. And now you have a really powerful mind tool.

Well, how did I get interested in expert systems? And following up from that, and after you've seen one, we'll talk about where I think they can be used in conservation and resource management.

I first got interested because they were just such a glamorous subject. It was kind of fun to read about these things. But then one day, I read a paper that said, "In order to capture the expertise of an expert, in order to be useful, a knowledge base has to be very, very large." Human expertise is complicated, and so the knowledge base, which is a model of that expertise also has to be complicated.

And, as you've probably realized, every time somebody tells me that a computer model needs to be complicated, my initial reaction is to say, "How useful is a simple model? Is it useful to develop a quick prototype? What can one do with something simple?"

So I got interested in the idea of whether it would be possible to develop a really simple knowledge base that is, nevertheless, useful. And with that idea in mind, I found a really good computer science student who developed a small expert system shell, and that small expert system shell eventually evolved into something called "WINEXP," a Windows Expert Systems Shell, which you're going to see in a moment. And I started working with people in conservation to try and create small but potentially useful knowledge bases. And I'm going to show you an example of one in a moment. But before we actually look at the computer, I need to give you a bit of a background to this example.

We're going to go back to the Kruger National Park again. And you might recall from previous segments that the Kruger National Park has an area of something like 7,000 square miles. Some of you may recall or have heard about the devastating fire that occurred in Yellowstone at some stage where a huge percentage of the Yellowstone Park went up into flames. Well, it turned out that the Kruger Park had had its Yellowstone experience, a huge uncontrolled fire back in the 1950s. And, as I recall, something like a third of the park was actually burnt. And management was so horrified by the specter of the whole of the Kruger Park going up into flames that they decided they had to make sure that this could never happen again.

And so what they developed was a grid of five-mile-by-five-mile fire breaks throughout the Kruger Park. In other words, the whole park was divided up into 25 square-mile blocks. And the idea was that if a fire broke out in a block, the fire breaks would probably contain it or, alternatively, it would slow it down enough so that people could get out there to put it out. And they were very proud of the fact that they were now controlling fire in the Kruger Park.

But there was a catch. And the catch was that African savannahs had co-evolved with fire. And if you took fire out of the system, you were fundamentally changing the ecosystem. And so after a few years of applying this policy of putting out all fires, they recognized that things were happening. The vegetation was changing in ways that were really bad for the animals that lived in the Kruger Park. And they were now faced with the fact that they had taken fire out. Could they put it back in, in a controlled way?

So they came up with a scheme for going in and burning these blocks in a organized fashion to try and simulate natural fire. And the experts on burning blocks would be able to go and look at a particular part of the park and say, "I think we should burn this, this year, and we should probably burn it at this

time of the year."

And the little expert system you're going to look at now was developed with the expert on fire in the Kruger Park. And at the back of your mind, you should have an idea of a novice, somebody new coming along to work in the Kruger Park, being sent out to some distant point with their laptop, and they have to decide whether to burn a particular part of the park and when to burn a particular part of the park. So they are carrying the expertise of the expert in their computer.

Let's go and have a look at how this might work.

[END OF VIDEO]

Document: blm0210.doc
Transcribed by: http://www.hiredhand.com