Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

FIVE SIMPLE PRINCIPLES OF MODELLING

M. Pidd
Department of Management Science
The Management School
Lancaster University
Lancaster LA1 4YX
UK

ABSTRACT

As discrete simulation software gets easier to use it is tempting to assume that modelling also gets easier, unfortunately this is not the case. This paper suggests some simple principles of modelling that can be easily applied in discrete simulation, whether the model is implemented in a visual interactive modelling system or in a programming language. The modelling principles are used to suggest desirable features that might be included in simulation software.

INTRODUCTION

Since discrete simulation first became a practical proposition in the 1950s, it is probably fair to say that its development has gone hand in hand with general developments in computing. As computers have offered more bang per buck, so simulation models have been able to grow bigger and more complicated. In addition, as software has become easier to use, it has been possible for non-specialists to set about the task of building discrete simulation models.

The last decade has seen the development of powerful Visual Interactive Modelling Systems (VIMS) such as ProModel, Witness and many others. Using a graphical user interface, a modeller is able to both develop and run a simulation model in a point and click mode. Though VIMS are apparently targeted at non-specialists, they also make life much easier for specialist simulation analysts, who are able to develop straightforward models very quickly. In common parlance, this is 'simulation in the small'.

But some simulation models still need to be implemented as bespoke software, whether using programming systems such as MODSIM or in general purpose languages such as C++. This is the case for applications in which the logic of the model is tortuous and/or those in which there are severe computational demands due to model size or the need for fast experimentation. Such applications of 'simulation in the large', which may consist of programs with tens of thousands of lines of code or more, seem likely to persist whatever the development of VIMS.

This paper discusses a few practical principles which are relevant to simulation modelling, whether in the small or in the large. They are not intended to contradict the rigorous approaches suggested by various authors, most notably DEVS as developed by Zeigler (1976, 1984) and his colleagues. Instead they are intended to sit alongside them, nagging away at the modelling conscience of the analyst. Five such principles are discussed here, some of them could be subdivided, but that would break the 'seven, plus or minus two' rule of memorability. They assume that simulation modelling is a practical and yet intellectually hard task, one which would benefit from careful thought and planning (Pidd, 1996). They also assume that a simulation modeller will have basic technical competence in simulation modelling.

PRINCIPLE 1: MODEL SIMPLE, THINK COMPLICATED

In cybernetics, the science of control, is a tenet which is often known as the principle of requisite variety, due to Ashby (1956), which might seem to support the view that complex systems require complicated models. The principle of requisite variety can be stated in many ways, one of which is that 'variety must match variety'. Its origin lies in work in designing control systems that are to operate in complex environments. In its simplest form it suggests that, to be effective, a control system must be able to match the system which it is controlling. Thus, if a furnace can get too cool as well as too hot then the control system should include some way of detecting and responding to low temperatures as well as to high ones. Stated in this form, the principle of requisite variety is almost a truism. Ashby took it rather further than this commonsensical notion and developed a mathematical theory of its use.

What of its applicability to simulation modelling? Must a model be as complicated as the reality which is

being modelled? Thankfully, the answer is no - for a reason which may not be initially obvious, but that is illustrated in figure 1. This shows that models are not just built, they are also used - which might be rather obvious, but is vitally important. It is crucial that the variety of the model and the user(s) combined can match that of the system being modelled but this does not mean that either one of the two components (model and user) must separately be able to do so. Systems theorists (Checkland, 1981) speak of emergent behaviour, which is the properties of a system that are not apparent in its components. In the context of modelling it is important to realise that the model and the user form a system. Requisite variety is an emergent property of that human:model system and not of its component.

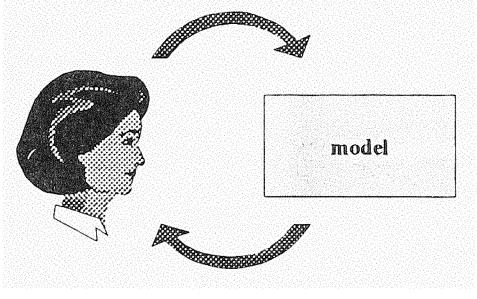

Figure 1: Requisite Variety in Modelling

The implication of this idea is captured in the aphorism 'model simple, think complicated'. That is, a simple model can be supplemented by highly critical thinking and rigorous argument and analysis. This can, of course, be taken to a ludicrous extreme. For example, a verbal model might be. "The world is hideously complicated and dangerous when things go wrong." No amount of critical thinking or analysis could take such a verbal model much further forward. At the opposite extreme, if a model itself does have requisite variety then there might be no need for much critical thinking. In such cases, the model could be built into a computer system which is used for day-to-day operation and control with no human intervention. Indeed, as Ashby pointed out in the principle of requisite variety, such complexity is essential if the model is to provide full and automatic control. Between these two extremes lies the realm within which most discrete simulation is conducted. The models are neither trivial nor fully requisite. Instead, they embody simplifications and aggregations.

The idea of simplicity needs to be linked with a suggestion from Little (1970) that a decision model should be easy to manipulate. As an analogy, there is a world of difference between the car driver and the skilled motor engineer. Most of us can drive cars whilst having only the vaguest of ideas about how the car works. We can do this because we have been trained to drive and also because the car itself gives us rapid feedback about our driving. We go off the road or hit other vehicles if we steer wrongly. A motor car is, after some training, easy to manipulate and it meets our preference for personal mobility. In a similar way, a model which is easy to manipulate (for example, its user interface might be made to resemble a spreadsheet) and which produces results which seem relevant, will be used. Thus simplicity has a second aspect, ease of use.

There are, of course, occasions when this metaphor collapses. In the world of travel, if we need to traverse the Pacific then driving across is not a viable option. Instead, we climb aboard a jet aircraft and surrender control to the aircraft crew with their own expertise. Similarly, there are some models which require the user to place their trust in the skills of the analysis team, because only they fully understand its workings. But users should only do so if the model produces results and insights which are relevant and appropriate to their situations. It is the joint responsibility of the analyst and the users to ensure that this is the case. Complicated models have no divine right of acceptance.

This idea that we might 'model simple, think complicated' brings us to the idea that models are 'tools for thinking'. It would be wrong to interpret that phrase as being 'tools to replace thinking'. Instead, they are tools to support and extend the power of thinking. Thus, a complicated model which is poorly employed may be worse than a simple model used as a tool for careful thought.

PRINCIPLE 2: BE PARSIMONIOUS, START SMALL AND ADD

The problem with the first principle of simplicity is knowing how simple or how complicated to be, and there is no general answer to this. Instead, like an army approaching a well-defended city at night, we use a little stealth and cunning. This is to employ the *Principle of Parsimony* which I have long found useful in computer simulation modelling (Pidd, 1984). In its more memorable form, this principle is sometimes known as *K.I.S.S.*, an acronym which stands for (*K*eep *I*t *S*imple, *S*tupid). The idea is that models should, ideally, be developed gradually; starting with simple assumptions and only adding complications as they become necessary. Instead of attempting, from the outset, a wonderful model which

embodies every aspect of the situation in a realistic form we begin with something manageable, which may have unrealistic assumptions. The intention being to learn what we can from this simple model and then to refine it gradually, wherever this is necessary. Powell (1995) calls the same approach 'prototyping', this carrying the idea that it is best to quickly develop a working model, even if it is imperfect. It can be refined, or even abandoned, later. Starfield et al (1990) provide some interesting insights into this approach as applied in general mathematical modelling.

An example of this approach is described by Miser and Quade (1990a) who discuss a model that might be used to estimate how much of a runway would be visible to pilots through partial cloud cover. This might be an important consideration for an aircraft coming in to land in cloudy conditions. There are various types of cloud, but at a height relevant to a plane seeking visual contact with an airport runway, only some need be considered. The first question is, how could these be represented in a model? As before, the principle of parsimony suggests that starting simple might be the best approach, and thus some simple geometric shape has much to commend it. One possibility is to treat discrete clouds as if they were circular disks. As Miser and Quade point out, using circular disks in this way is not ridiculous as the relevant types of cloud do tend to be compact. It is also a helpful simplification, since the geometry of circular disks is well understood.

Cloud cover is often measured as a percentage of the sky that is visible from points on the ground and its pattern may be specific to the location of the airport. For example, nearby hills or ocean may cause certain patterns to predominate. These patterns could then be modelled at different heights by the use of the simplified circles to represent the clouds, and in this way it should be possible to estimate the visibility of the runway from different points in the sky. Whether this model is close enough to the likely 'real' distribution of cloud cover is the crucial question. This could be at least partially assessed by having an aircraft fly past different points under known cloud conditions and then attempting to compare the actual visibility with that predicted by the model. If the model is found to be too simple, then it can be further refined by, for example, using shapes which are closer to the actual shape of clouds. For instance, each cloud could be modelled as a set of circles which overlap (producing shapes which resemble Mickey Mouse's face in one form). Again, this refinement might be chosen because the geometry of circles is simple and thus the model is tractable.

Hence the principle of parsimony requires us to develop models that are, initially, too simple for the task in hand. Ideally, the basic structure of the model will be correct, but even this need not be so. The modeller then gradually refines or replaces the model, all the time bearing in mind the first principle of simplicity. The idea being to add nothing unnecessary to the model.

PRINCIPLE 3: DIVIDE AND CONQUER, AVOID MEGA-MODELS

This is common advice given to anyone trying to understand how a complex system operates. Powell (1995) calls this decomposition, as well as divide and conquer. Raiffa (1982, quoted in Miser and Quade, 1990b) has the following to say.

"Beware of general purpose, grandiose models that try to incorporate practically everything. Such models are difficult to validate, to interpret, to calibrate statistically and, most importantly to explain. You may be better off not with one big model but with a set of simpler models.."

Raiffa's point is partially related to the first two principles above, but also relates to the need to build a model from components, each of which should be themselves developed parsimoniously.

This is, it should be noted, not the same as advising software developers to use modular designs in the their computer programs, laudable though this advice may be. The suggestion is, rather, that the best way to face up to complexity is to break it down into manageable chunks, all the time making sure that the inter-relationships between the sub-models are well-understood. In really large scale simulation modelling, which might involve a sizeable team of modellers, then this is the only way to proceed. It requires careful project management, to ensure that each participant is operating as part of the team, but it is a very fruitful way to proceed.

PRINCIPLE 4: DO NOT FALL IN LOVE WITH DATA

A common failing of students when learning about modelling, is to insist that progress cannot be made unless there is some (or more) data available. Their assumption being that examination of the data will provide some clues that will extend their understanding. This may well be a mistake, even though exploratory data analysis is a very valuable

technique. Modern statistical software or spreadsheets enable the rapid plotting and summary of large amounts of data and from this analysis, patterns may be quickly gleaned. Sometimes these patterns exist, but at other times they are, like beauty, in the eye of the beholder. Nevertheless, exploratory data analysis has much to commend it as an approach. It is, however, no substitute for careful thought and analysis.

Some of the dangers and pitfalls that await the unwary in their treatment of data are discussed below, but there is a fundamental point that should not be missed. This is that the model should drive the data collection and not vice versa. This means that the analyst should try to develop some ideas of the model and its parameters and from this should think about the type of data that might be needed. One of the problems with some case-style teaching is that the cases are often intended to be self-contained. That is, the students know that all the data they may need is available in the papers issued with the case. This is quite unlike real life in which data must be requested, justified and collected before it can be analysed. Data is not free, its collection has a cost as does its interpretation and analysis.

How then should the use of data be linked in to the parsimonious, gradual approach advocated earlier in this paper? If circumstances permit, then the best approach would be to develop a simple model and then to collect data to parameterise and test it. It may then be clear that the simple model is fine for the intended purpose or it may be that the model needs to be refined - which may need more data or different data. This new data will have a cost which should enable some rough cost:benefit calculation to check whether its collection and analysis will be worthwhile. And so on, until some point is reached at which the costs outweigh the benefits.

Of course, these ideal circumstances may not pertain and it may be necessary, especially when acting as a fee-charging external consultant to set up a complete data collection exercise at the start of the work. If this can be resisted then it would be a good idea to do so.

Within this fourth principle, there are a number of specific issues than can be addressed. These are almost principles in their own right, but are usefully gathered under this sceptical fourth principle.

1. *Data mining and data grubbing*: Data-mining is a term which has recently entered the language of statisticians and refers to attempts to develop statistical models from available data. Powerful computer packages are used to search for patterns in the data. A debasement of this is "data-grubbing", which some use as a term of abuse for approaches in which many different data series are unthinkingly collected and then read as data files by one of today's powerful statistical packages. These packages allow complicated analyses to be conducted on the data in a very short space of time. Regressions can be tried, linear and non-linear, other types of multivariate plastic surgery can be applied to the data and the original data series can be transformed by taking logarithms and the like. Within a simulation context, there are friendly packages to enable modellers to fit probability distributions to data that has been collected. This can all be done very quickly, by someone who knows very little about the statistical tools being used in the computer software. This can be like an attempt to bake a cake by collecting ingredients that look interesting, then mixing them together until boredom sets in, popping the resulting melange into the oven and waiting for the smoke to appear. Cakes cooked in this way should only be given to people with whom scores need to be settled.

This criticism of data-grubbing should not be interpreted as a call for a ban on friendly, powerful statistical packages. They are much too useful for that and they take the drudgery out of statistical modelling. However, they should not be a substitute for thought. Also, just because data is available it should not be assumed that it is useful.

2. *Data is useful in model building*: This criticism of data-grubbing and of a reliance on available data might be interpreted, wrongly, to imply that modelling is best carried out in an abstract way. It is certainly not the intention that data should be ignored in this way. It might, therefore, be helpful to divide data and information into three groups. First, there is preliminary or contextual data and information. One practical approach to problem structuring is to use the easy to remember questions of What, Why, When, Where, How and Who (Kipling's honest working men) as a useful guide in preliminary investigation. Clearly, the results of these questions may be qualitative or quantitative. In the latter case it may be necessary to conduct significant analyses on the data that is so produced. But this data is collected with a view to understanding more about the context of the problem, rather than the development of a detailed model. It may not be unusual for this preliminary analysis to reveal enough insights for there to be no real need to take a project any further.

The second type of data is that which might need to be collected and analysed in order to develop the model in some detail. This is model parameterisation, or model realisation (Willemain, 1995). But, as is

repeatedly stated here, the model structure should drive the data collection and analysis, not the other way round. The third type of data is discussed under the heading 'Avoid using the same data to build and to test a model'.

3. *Beware of data provided on a plate*: An old adage amongst Management Information Systems professionals is that information is data plus interpretation. One feature of modern organisations is that computer systems collect almost every conceivable type of data about what is happening - except the data which you really need for a particular model. For modelling purposes, data is best ordered a la carte rather than table d'hôte. For example, in attempting to develop models for production control, it may be necessary to produce sub-models of customer demands for the products being made. Most companies monitor their sales by using data produced by the sales order processing systems that are used to take orders and issue invoices. Thus, the obvious way to get hold of demand data might be to take it from the customer order files, but there are at least three reasons why this might be a mistake.

The first is that such systems often only record the actual despatches to customers and this may be as much a reflection of the available finished stock as it is of the actual demand from customers. They may request one thing, but the company may be unable to supply and they may thus go elsewhere or may accept a substitute. The second reason is that, if the customer suspects that a required item is out of stock (due to past experience) they may not even request the item. Finally, the whole idea of implementing a new production system might be to make the company more attractive to customers who normally order from other suppliers.

Hence, for all of these reasons, the data from a sales order processing system might be treated with some caution. In this and other cases there may be no substitute for proper and well-organised data collection if a useful model is to be constructed. It may also be possible to take existing data and massage it in such a way as to account for some of its shortcomings. Thus, a small scale data collection exercise might be used to reveal the discrepancies in the full, system-produced data which may then be modified to take account of this. However, it must be born in mind that such data massage implies that a model of the data itself is being used as part of the modification process.

4. *Data is just a sample*: It is also important to remember that, in the vast majority of cases, data is just a sample of what could be used or might be available. This is true in a number of dimensions. First, the time dimension, this being the simplest to understand. When data is being used to build or to test a model then that data will have been collected at a particular time and over a certain period. If we say that the data is believed to be representative then we are implying that it is representative of a larger population which displays some regularity through time. We are not expecting someone to come along later and surprise us with data which differs drastically from that which we have already obtained. Nevertheless this may happen and is always a risk which is clear when we realise that the results of a model may be used to extrapolate into the future. The future may simply differ from the past, that is the population from which the data sample comes may behave differently in the future. Data is also a set of observations and this is the second aspect to note in the realisation that data is a sample of what might be obtained given enough time and other resources.

5. *Avoid using the same data to build and to test the model*: Most models used in management science make use of data in one form of another. The model will have parameters that must be given values. This process of parameterisation is usually based on data, whether specially collected or generally available. The trap to avoid, if possible, is the use of the same data to parameterise the model and then to test it. As an example, suppose that a company wishes to understand the sales pattern for one of its products. Careful data collection proceeds and they eventually have a time series of their weekly sales over the last 2 years. A management scientist then develops a forecasting model which is intended to suggest how sales might behave, given certain assumptions, over the next few months. The idea being that the model might be used, say, monthly each time with an updated data series, to suggest likely future sales.

There are a number of types of forecasting model that could be employed, and they have in common the fact that they are based on the analysis of historical data. Under skilful hands, computer programs are used to estimate the parameters of equations which lead to a model which is a good fit to the recent historical data. The goodness of fit can be expressed in standard statistical terms. But this goodness of fit is an evaluation of the degree to which the model fits the historical data. It is also important, where possible, to test how well the model predicts what might happen in the future. A tempting short-cut is to quote the goodness of fit

statistic from the parameterisation exercise but this might be a mistake. A better approach, is shown in figure 2, in which the available data has been divided into two sets. The earlier data is used to parameterise the model and then the second set is used to test the model. This test set has not been used in the parameterisation but is being used as a surrogate future. Goodness of fit measures can be used to assess how well the model predicts this surrogate future.

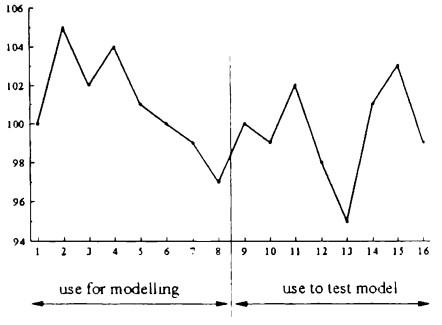

Figure 2: Use of Data in modelling

PRINCIPLE 5: MODEL BUILDING MAY FEEL LIKE MODELLING THROUGH

Because a simulation model is the result of an attempt to represent some part of reality so that action may be taken or understanding may be increased, it might be thought that model building is a linear and highly rational process. There have been few attempts to investigate this issue, as it relates to management science, but the evidence suggests that modelling is not a linear nor a smooth process. Instead, people seem to 'muddle through', making use of insights, perhaps taking time away from the modelling, trying to look at things from different perspectives and so on. This does not, of course, imply that modelling must be done this way, but it may well indicate how successful analysts actually operate.

A fascinating attempt to investigate this issue is reported by Willemain (1994). He gained the co-operation of a group of twelve experienced modellers, a mixture of practitioners and academics. All had completed graduate work in OR/MS and their average years of experience since finishing graduate school was over 15 years. They were not, in any sense novices. This group were asked to do two things. First they were given the chance to describe themselves, to express their views on their own approaches to modelling, to say what experience they had in modelling and to capture something important in a short, personal modelling story. Their self-descriptions and experience showed them to work mainly in areas in which they "pursue specific objectives towards fundamental changes in complex, existing systems". Also that they "develop a unique model for each problem, though all their models involve extensive computation". From this we can conclude that their work could in no way be described as 'airy-fairy', for their concerns seem down-to-earth.

Of their approach to actually building and developing models, Willemain (1994) summarises their responses thus. They "develop their models, not in one burst, but over an extended period of time marked by heavy client contact". Also, they are "guided by analogies, drawing and doodling, they develop more than one alternative model, in each case starting small and adding". Thus many of their claimed approaches are a good fit with the first four principles of modelling discussed in this paper.

In a second paper, Willemain (1995) reports an experiment with the same twelve people who were given modelling tasks and were asked to think aloud as they spent 60 minutes figuring out how best to develop suitable models. This is clearly an artificial task for two reasons. First it compresses their activity into just 60 minutes, when their expressed preference was to work "over an extended period of time". Secondly, the request to think aloud as they worked might distort their normal patterns of work. Despite these reservations, this thinking-aloud protocol reveals some interesting issues.

In analysing the tapes of their thinking aloud, Willemain (op cit) classifies their concerns under six headings.

- *The problem context*: which he relates to problem structuring as defined by Pidd and Woolley (1980). That is, the exercise which aims to gain a sufficient understanding of the problem to proceed to some form of formal modelling.
- *The model structure*: which he takes to be the process of deciding what category of model to use and of analysing data prior to actually building it.
- *Model realisation*: this is the process of parameter estimation for a model and/or calculation of results.
- *Model assessment*: which is deciding whether the model will be valid, useable and acceptable to a client.
- *Model implementation*: which is working with the client so as to gain some value from the model.

The tapes show that about 60% of the modellers' time was devoted to model structure, that is, what would be regarded as the core of model building.

About 30% of the time was divided equally between concerns about problem context and model assessment, with similar time spent on each issue. Just 10% was devoted to model realisation and almost none to questions of implementation.

As the study gave the modellers just 60 minutes to work on a problem it should be no surprise that so little time was devoted to model realisation or implementation. But what is significant is that so much time was spent on thinking about problem context and model assessment. What is also very important is the fact that the time spent on these three major concerns was scattered throughout the modelling session. The modeller kept picking up a concern for a while, dropping it, and then returning to it. Presumably this would be even more marked were it possible to follow how they operate over a much longer time period in their 'real' work.

Perhaps it is an exaggeration to say that modellers muddle through. However, it is equally an exaggeration to assert that modelling proceeds as a linear step-by-step process. It did not in Willemain's study and it probably does not in most of our experience. The other principles presented here should be used to provide some order to the muddling through. It is quite normal for a modeller to think in parallel whilst working on a model. Discussing ill-defined problem solving in general, Holyoak (1990) discusses how people tend to operate in parallel lines of thought and how they are continuously restructuring their ideas. A concern to understand the problem context goes hand in hand with a desire to produce a model which will be useful, as well as technically correct.

IMPLICATIONS FOR SIMULATION SOFTWARE

What then are the implications of these five principles for developers of simulation software? In answering that question, it must be re-emphasised that they assume that the modeller has some basic technical competence. Likewise, this brief final section assumes that the software is basically well designed and implemented, at least in technical terms. Over and above this basic competence, what help can software developers and vendors give to simulation modellers? Perhaps the best way address this is to look as at each of the five principles in turn.

1. *Model simple, think complicated:* This suggest something that is already well known to software developers, that modelling systems need to support the process of critical thinking and analysis as well as the process of model building. This might, perhaps, focus on the need to support experimentation, both formal and graphically based. At a basic level this implies support for common file formats to permit straightforward use of analysis packages. At a more sophisticated level it might mean the availability of database type support for analysis across a whole range of runs.

2. *Be parsimonious, start small and add*: This suggests the need for software that is properly layered, with top levels devoted to a VIMS sitting above a simulation code, which itself draws on libraries and may translate into C++ or similar. The reason for this being that the modeller needs some way of developing rapid prototypes that are either enhanced or discarded. Thus the VIMS may be used for this initial development, but the ability to translate the VIMS into a simulation code or into C++ means that a simple model can be enhanced in ways that are outside the scope of the VIMS. In addition, if the simulation code and its base language are object oriented, the early models could be code-based rather than in the VIMS and might draw on sensible class libraries.

3. *Divide and conquer, avoid mega-models*: If models are to be built up from smaller, relatively independent models then the modeller needs to operate within some paradigm that supports this. Two possibilities spring to mind, both of which have been found to be practically useful. DEVS and its extensions (Zeigler, 1976, 1984) was proposed with this need partly in mind. Object oriented systems provide an alternative way to attack this problem, as they require any model to be atomised into relatively independent components. Finally, the wide-spread use of windowing operating systems and the current vogue for 'network computers' means that it may, in future, be easier to develop communicating applets than has been the case until recently.

4. *Do not fall in love with data*: If it is easier for a modeller to develop rough-cut or prototype models then, possibly, an infatuation with data is less likely. Thus, the comments made above under the first three principles offer some support. But it seems unlikely that software developers can offer much more than this. It may even be that packages which ease (for example) the selection of input distributions may make things worse - though this most certainly need not be the case. It seems that thorough training is the only way to face up to this principle with all its variants.

5. *Model building may feel like modelling through*: The suggestions made above should help in this principle too, but another suggestion might also

do so. When someone is muddling through with their modelling, then ease of use and simple interface design seems very important as does software that is fun to use. If the modeller has also, as is commonly the case, several tasks and modelling projects underway at the same time, then the question of interface support seems very important. The interface needs to be standard, predictable, fast to use and satisfying in its response. Not too much to ask, surely?

REFERENCES

Ashby R. (1956) An introduction to cybernetics. Chapman and Hall, London.

Checkland P.B. (1981) Systems thinking, systems practice. John Wiley & Sons, Chichester.

Holyoak K.J. (1990) Problem solving. In Osherson D.N. and Smith E.S. (1990) An invitation to cognitive science, volume 3: Thinking. MIT Press, Cambridge, Mass.

Little J.D.C. (1970) Managers and models: the concept of a decision calculus. Management Science 16, B466-B485.

Miser H.J. and Quade E.S. (1990a) Validation. In Miser H.J. and Quade E.S. (1990) Handbook of systems analysis: craft issues and procedural choices. John Wiley & Sons Ltd, Chichester.

Miser H.J. and Quade E.S. (1990b) Analytic strategies and their components. In Miser H.J. and Quade E.S. (1990) Handbook of systems analysis: craft issues and procedural choices. John Wiley & Sons Ltd, Chichester.

Pidd M. (1996) Tools for thinking: modelling in management science. John Wiley & Sons Ltd, Chichester, forthcoming.

Pidd M. (1984) Computer simulation in management science (1st edition). John Wiley & Sons Ltd, Chichester.

Pidd M. and Woolley R.N. (1980) A pilot study of problem structuring. Jnl Opl Res Soc, Vol 31, pp1063-1069.

Powell S.G. (1995) The teacher's forum: six key modeling heuristics. Interfaces 25, 4, 114-125.

Raiffa H. (1982) Policy analysis: a checklist of concerns. PP-82-2. International Institute for Applied Systems Analysis, Laxenburg, Austria

Starfield A.M., Smith K.A. and Bleloch A.L. (1990) How to model it: problem solving for the computer age. McGraw Hill Publishing Company, New York.

Willemain T.R. (1994) Insights on modelling from a dozen experts. Ops Res 42, 2, 213-222.

Willemain T.R. (1995) Model formulation: what experts think about and when. Ops Res 43, 6, 916-932.

Zeigler B.P. (1976) Theory of modelling and simulation. John Wiley & Sons Inc.

Zeigler B.P. (1984) Multi-facetted modelling and discrete event simulation. Academic Press.

AUTHOR BIOGRAPHY

MIKE PIDD is Professor of Management Science in the Management School of Lancaster University in the UK. He has written two books on simulation, of which the best known is Computer Simulation in Management Science (John Wiley). He has just completed a new book for the same publisher (Tools for thinking: modelling in management science) which addresses the issues often raised by critics of rational modelling. His current interests in computer simulation include object oriented methods.