

Chapter 12

User Interfaces

Chris V. Jones

*Faculty of Business Administration, Simon Fraser University, Burnaby, BC V5A 1S6,
Canada*

1. Overview

The field of operations research (OR) has traditionally concentrated on normative techniques for optimizing, simulating, studying – in short – analyzing complex problems to provide better understanding. Yet the analysis techniques do not stand alone. Quite simply, data must be input and reports output and disseminated. The analysis techniques are packaged by a user interface. It is arguable that for many OR projects, the user interface has been primarily an afterthought – a necessary evil, but not given a great deal of attention. Yet different user interfaces for the same task produce significant, measurable differences in user performance [e.g., Card, Moran & Newell, 1980b]. If the techniques are designed to promote maximum understanding of the complexity of the underlying problem, the user interface should be as helpful and sophisticated as the underlying algorithm. Achieving such a helpful user interface is not easy. In this chapter, we present theories, guidelines and techniques to help design and build user interfaces for delivering OR applications.

But how exactly does a user interface contribute to the success or failure of the application of OR? Geoffrion [1976] suggested that the principal benefit of an OR project should be ‘insight, not numbers’; to paraphrase, an OR project should contribute insight, not just models. Models are just that, models or approximations of the underlying system. They should not be trusted, at least initially. They must be probed, tested and verified. Model developers frequently have trouble understanding the behavior of their own models: is the strange behavior caused by a programming error? is it caused by an error in the input data? by an error in the model formulation? If model developers experience difficulty understanding the behavior of their own models, why should a manager be expected to accept the model? As Geoffrion [1987] stated, ‘... MS/OR practitioners and their work often are incomprehensible to non-specialists.’

We assert that understanding model behavior can depend less on the actual

models than on how the models are packaged. Developing insight on model behavior is ultimately a process of discovery, of finding trends, surprising behaviors, and comparing the behavior of the model to what is expected or observed in the 'real' system. How well does it match? Where does it differ? Do those changes correspond to what is expected? If yes, why? If not, why not? In one sense developing insight involves recognizing patterns. How does the model respond to changes in parameters? What trends can be detected? How do the trends compare? Pattern recognition can be performed formally with detailed statistical analysis, of course, or on the back of an envelope. Yet pattern recognition is also within the province of packaging of the model, i.e., the user interface. Since the user interface presents the output of the model to the user/modeler/decision maker/manager, it is the primary window onto the actual behavior of the model. Obviously it should be as clear as possible; not so obvious, however, is what is meant by 'as clear as possible'.

Understanding model behavior is only a part of the story, however. The user interface not only presents the output of the model to the user, but also the medium through which the user creates the model. Creating a correct and complete model remains a daunting task. For example, creating a correct linear programming model using MPSX is quite difficult. Matrix generators and modeling languages, by providing a more convenient medium for expressing models, can greatly speed the effective time required to obtain useful results, even though they do not speed the actual execution of a linear programming algorithm in any way. Modeling languages and matrix generators simply provide a more productive interface between the modeler and the OR technique.

This discussion of the importance of user interfaces takes place against a background of increasingly sophisticated hardware and software such as computer graphics, speech generation, mice, light pens, multimedia, and windowing systems. These developments have made possible vastly better user interfaces than those that were possible using punched cards, or paper-fed terminals. As many authors have noted [Bodily, 1986; Vazsonyi, 1982; Papageorgiou, 1983; Wynne, 1984; Geoffrion, 1983], these technologies provide a promising vehicle for delivering OR techniques to a wider audience.

OR technology packaged in a well-designed user interface can be very successful. In a forestry application [Lemberski & Chi, 1984], which won the OR practice prize, a traditional dynamic programming algorithm was developed to determine optimal cutting patterns for logs. Although the dynamic program determined the 'optimal' cutting pattern, Lemberski and Chi stated: 'The challenge in implementing MS/OR is often not in developing a technically sound model and solution algorithm, but in generating credibility in the minds of potential users.' Although the computer was not brought into the forest (as of 1984), operators used the system to hone their decision making skills and to design a revised set of operational procedures for cutting logs. In this and other applications, the packaging of the OR technique proved to be one of the critical factors to the success of the project.

As further evidence of the importance of the user interface, users of