Towards a Tool Kit for the Systems Analyst

D. BENYON* AND S. SKIDMORE
School of Mathematics, Computing and Statistics, Leicester Polytechnic, P.O. Box 143, Leicester LE1 9BH

A review of the major systems analysis methodologies is undertaken in order to examine the aspects of information systems (IS) which they seem best able to represent. The importance of a collection of techniques – the analyst's tool kit – is stressed, and aspects of systems analysis poorly covered by existing methodologies are highlighted. The conclusion is drawn that one single methodology cannot cover the whole range of systems tasks.

Received July 1985, revised April 1986

1. INTRODUCTION

During the 1980s a number of system design methodologies have been promoted or adopted by an increasing number of organisations, reflecting an upsurge of activity in IS development. These methodologies have been comprehensively described, discussed and compared in a number of papers and publications (Refs 1–5). Wood-Harper and Fitzgerald[3] categorised the approaches according to the underlying paradigm ('scientific' or 'systems'), the conceptual models which they use and the objectives which they seek to meet. This analysis was useful in so far as it brought these different approaches together, but it did not put them sufficiently into their developmental context. Indeed, the categorisation distracted attention from the purpose of the approaches and concentrated on an argument about the relevance of these distinctions. This paper provides an alternative perspective on the use and usefulness of these approaches and presents an appreciation of their relationship to each other and to the problems of information systems analysis.

Despite the growth of these new methodologies, IS still broadly uses a third-generation approach to developing computer systems. This is characterised by a linear approach to systems analysis (the familiar phases of analysis, design, construction, implementation) coupled with a freezing of requirements at some point. This has been relatively successful in computerising well-defined, cyclical business systems, but appears unable to cope successfully with four major forces now evident in the market place.

- Cheap and versatile microcomputer hardware and software.
- Increasing diversity in applications with a marked tendency towards systems where information requirements are difficult to define.
- The development of new and powerful software, (particularly application generators).
- The increased sophistication and knowledge of end users.

The tools of the trade needed to deal with these changes in computer hardware, software and user characteristics need to be discussed and established.

2. MODELS

Fundamental to all methods of analysis and design is the model. This is a subjective representation of some aspects of the real (or realisable) world. Good modelling is essential to the effective understanding of problems in many disciplines, for example, Raphael[6] has shown the importance of choosing the appropriate representation in problem solving. If the essential features of a problem can be identified a solution may become readily apparent.

In the analysis and design of information systems there are many different aspects and interrelated problems which have to be tackled together. Information systems include people, data, information, procedures, hardware and software. This diversity leads to the need for many different representations of the same problem area in order to help determine what the fundamental features are. These different aspects can only be achieved by modelling the system from a variety of perspectives.

The importance of models in information systems has been recognised by several authors, particularly Wilson[14] and Martin.[15] Wilson stresses the importance of hierarchy and the use of models for exploration, communication and testing hypotheses, whilst Martin is particulary concerned with system development, documentation and maintenance. These are all important features but they are not comprehensive.

Beer[16] makes the point that models do not split things up. Rather, they simplify wholes whilst retaining their structure and in this way they highlight important features. A series of models is usually needed to represent the problem situation effectively at different levels of detail, and hence models need to be organised hierarchically. Martin[15] emphasises hierarchy because of the need to structure complex concepts, but it is also important because different features are more significant at different levels of abstraction. A book of road maps illustrates this. In planning a route, the traveller looks at the front of the book at a page which shows the major cities and page numbers where more details can be found. Turning to the relevant page, he finds the major trunk roads with towns represented as shaded areas and hence can select a suitable route. At the back of the book are often found detailed street plans, which enable him to pinpoint his destination more closely. Clearly there is a need for each type of representation to help solve different problems. In some circumstances one modelling level might suffice, but consider driving from Norwich to Cardiff with only a set of Ordnance Survey 1 : 25000 maps. It is possible, but it is likely to be unnecessarily difficult and inefficient.

The book of maps illustrates the importance of a hierarchy, appropriate simplicity (the maps only show relevant details) and consistency (motorways are always blue) in modelling, but it only illustrates one use of

* Now at Open University

models: communication. A further restriction of the road map analogy is that it assumes that the traveller knows where he is and where he wants to go. In information systems there is a need for more rich and descriptive models, a guide book to draw attention to interesting areas. Models also have uses beyond communication. They are invaluable for exploring ideas (an architect's initial drawings), for experimentation (the scale model car in a wind tunnel) and for making predictions (the computer models of the economy).

Thus models are used in a variety of ways and so different models and modelling techniques are appropriate at different times and for different tasks. The skills of choosing an appropriate model are rarely considered.

3. MODELS IN INFORMATION SYSTEMS ANALYSIS

In information systems analysis there are many modelling techniques available–from traditional paper and pencil flowchart to the interactive prototype system. Some have been put together into methodologies characterised by a particular model or tool. The different methodologies can be viewed as belonging to one of the five major approaches indentified by Hammersley[2] and adopted by Wood-Harper.[3] We will examine each of these approaches in terms of what they model and how useful they are at modelling that aspect.

3.1 Soft systems approach

Checkland[7] recognises the difficulty of defining problems and suggests that as much time as possible is spent exploring the richness of what he terms the problem situation. Checkland's methodology emphasises the subjectivity of systems analysis and the importance of an iterative approach to the activity. It stresses that there are many different, valid views of a problem situation and these should all be considered before arriving at the 'root definition' of a perceived problem. However, it is unlikely that the practising analyst can wander as far as Checkland suggests or recommend solutions as radical as may be desirable. Typically the relevant root definition is managements, the environment is a bounded part of the organisation and the elements of the system are the data items. Despite this, the approach provides a good framework for systems analysis and, most importantly, provides a language in which to discuss the purpose of the system. The recommendation to build conceptual models of the system 'from the verbs of English' may often be inadequate and to some extent this has been recognised by Checkland's colleague, Wilson.[14] He has extended the methodology by specifying more closely what is required to build conceptual models, moving from an 'issue-based analysis' through a 'primary task model' to the establishment of input, output and control mechanisms for each activity. He derives data models and organises all information processing into a matrix which he dubs the 'Maltese Cross'. The soft systems methodology is useful at the early stages of analysis, in establishing important sub-systems and the definitions of what these systems are. The development of conceptual models away from the real world is a vital activity, as it allows the analyst to concentrate on the logical functions of the business system. However, modelling those systems requires more sophisticated techniques than verbs. Wilson has discovered this, but by inventing his own tools he has ignored some useful existing ones, particulary those concerned with data analysis and information flow. The soft systems philosophy has also been made more accessible in Wood-Harper *et al.*'s multiview approach.[25] Indeed it represents the strongest and most convincing part of this method of information systems definition, and its exploration of rich pictures is particularly effective.

3.2 Structured Systems Analysis and Design (SSAD)

The dataflow diagram (DFD) of the SSAD approaches[8] is a good tool for modelling data flow irrespective of physical and organisational boundaries and the medium of that flow. It provides a mechanism for ensuring a consistent hierarchy ('levelling' procedures) and is a useful analysis tool. Used sensibly it can provide an immediate and understandable model of the essential inputs, outputs and processes of the system. It is also a good design model, permitting the production of alternative information flows and providing a focus for discussion about the location of the human–computer interface. The elements modelled – flows, processes and files–may also lead to their physical equivalent.

The DFD and the activity model of the soft systems approach have enough common ground for the techniques to be linked together.

Both emphasise the concept of hierarchy in developing models. DeMarco's top level is the 'context diagram', which 'shows only the net inputs and outputs'. He continues: 'it serves only one purpose...to delineate the domain of our study' and furthermore 'think of the context diagram as a transformation, a process that transforms the input data flows into the outputs'. DeMarco breaks the context diagram down into levels until he reaches 'functional primitives... bubbles which are not further decomposed into successively lower-level networks. Diagram 0...portrays the breakdown of an area into a network of components'.

Checkland constructs an activity model from the root definition 'Assemble the small number of verbs which describe the most fundamental activities necessary in the system described'. He emphasises modelling only essential flows. He continues: 'once the model has been built...it may be used as a source of...models expressing flows and/or possible structures'.

Both emphasise that the systems represent transformations. DeMarco claims that 'A process is a transformation of incoming data flows into outgoing data flows' and Checkland 'we may regard a system as an entity which...tranforms the inputs into the outputs'.

The similarities between the approaches–their emphasis on hierarchy, data or information flow and the transformation function of processes–enables SSAD to be used as the conceptual modelling tool once the system has been defined by the root definition. The visual attractiveness of the DFD makes it more effective than verbs.

3.3 Traditional approach

The traditional approach (see Ref. 9) is showing its age, and whilst it was appropriate for batch systems it is not

flexible enough to cover the range of modern systems. It focuses on functional analysis, 'fact finding' and the flow of control. It has already been criticised in Wood-Harper's paper,[3] and we generally agree with those criticisms. However, functional analysis and an understanding of the flow of control may be useful, particularly in documentation and the early stages of system understanding. Well-structured, columnar flowcharts can effectively model aspects of the system which are not reflected in the dataflow diagram. Indeed, there is far more structure in a well-drawn flowchart than there is in a DFD, and whilst this may reflect the (perhaps inappropriate) organisational structure of the firm, there are many instances where this structure will continue to exist. Even where it is to be changed, the flowchart can highlight features which would be overlooked by a purely dataflow analysis. Millington[15] attempts to exploit this aspect of the flowchart by combining its columnar structure with a dataflow view of the system. In doing so he loses the essence of both the models because the strengths of each are obscured by the other. For example, two strengths of the DFD are the conveying of information along the flowline and the lack of any arbitrary distinction between decisions and processes. Millington's diagrams fail to recognise this, and his amalgam does both 'parent' techniques a disservice.

3.4 Data-centered approach

Data analysis (e.g. Ref.10) looks at a static representation of the information content of a system. It is the only technique to do this, but at the cost of omitting other vital aspects such as personnel and data flow. It is impossible to classify data analysis as an analysis or design tool as it sits firmly on the borderline of the two. We cannot agree with Wood-Harper's comment that '...data analysis says very little about system design or problem solving'. In the first instance data analysis is central to competent Physical design, particularly if the target software is a DBMS. Secondly, the rigour of data analysis often uncovers important issues which prompt for further detailed systems analysis. Moreover, as we discovered in a recent implementation, the clarification of enterprise rules may lead to fundamental insights into the whole problem situation.

In recent years the data-centred approach has extended beyond simple data analysis. For example, an element of hierarchy has been introduced through the use of entity clusters.[18] To some extent this has become necessary because of the use of data analysis at a strategic level, resulting in very large data models. If the rules described by Miller[18] ensure consistency between the levels, entity clustering appears to be a useful addition to the data-centred approach. However, we have yet to be convinced that this is the case. The static representation of data analysis has been extended by the development of the Entity Life History (ELH).[19] The ELH considers the system from the point of view of the entities. Rather than asking how our entities are affected by transformations, we might ask what events occur that affect each of our entities.'[20]

Each entity is examined to see which events affect it and in what possible sequence. The processing required by each event is determined and the effect of sequence on that processing considered. A model of the life of an entity is created, charting the sequence and processes of events which may affect it during its time in the system. The ELH adds a dynamic dimension to the static model of data analysis. It also aids exception and error identification and handling and adopts a notation that fits in well with program development strategies.[21]

The data-centred approach offers a number of tools, but retains data analysis as its central model. It has powerful analysis and design possibilities, and like the dataflow diagram it is independent of technology and hence allows a variety of implementations. It can be supported, again like the DFD, by a comprehensive data dictionary. On the debit side, its emphasis on data, coupled with the apparent rigour of its techniques, can lead to the development of neat technical systems that ignore or contradict the political reality.

3.5 Participative approach

The participative approach is a useful heading under which to consider both the socio-technical system design of Mumford *et al.*[11] and the development of system prototypes.[13] The Land and Mumford School place emphasis on the successful management of change with specific techniques designed to minimise the uncertainty that change brings. The inclusion of such techniques as job satisfaction analysis and future analysis certainly makes it more than an 'implementation strategy'.[3] Its emphasis on the involvement of the user recognises the common mismatch between delivered and requested systems and also the 'people problems' that bedevil most projects. In contrast it has little to say on the technical aspects of system design, and doubts have been raised about the competence of users to contribute or comment on such matters. Its stress on formal participation also suggests that it may be out of step with the corporate policy of many companies, perhaps even with the current climate and expectations of the country itself. The underlying philosophy of the approach seems more at home in the mid-seventies than in the job-hungry, assertive eighties. However, despite such reservations, we must acknowledge that the methodology focuses on areas of system design that are seldom considered by other approaches and yet are known to be important in determining the success of a systems project.

Mumford has recently included many of the tools of the participative approach in a methodology for systems design called ETHICS.[24] Its stated aim is to 'ensure that users' specifications are accurate' and concentrates on techniques for achieving this. The general approach is at a similar level to Checkland's, but with the emphasis firmly on understanding by users of what they want and what is possible.

The prototype approach has attracted much attention and support. The case for this approach to system development was eloquently explained by Naumann and Jenkins[26] and has gained enthusiastic backing from others (e.g. Refs 26–28). It is an attempt to replace the life-cycle approach with a paradigm that responds well to complexity and uncertainty. This is essential if high-risk, high-value information systems are to be successfully developed. The life-cycle method may be useful in developing data-processing systems, but it is unlikely to be appropriate in the design and delivery of decision-support systems. The quick delivery of skeleton working

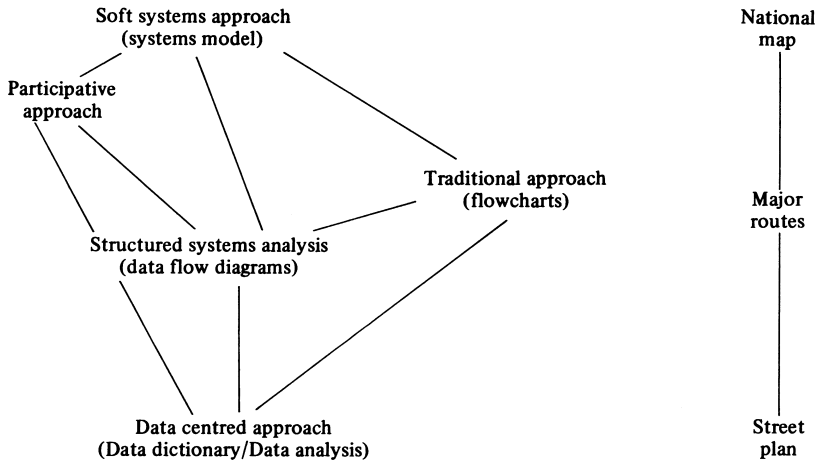

Fig. 1. Hierarchy of approaches and principal tools.

systems requires supporting software. This has been provided by the emergence of application generators designed to speed-up the development process (see Ref. 30 for a review of products). However, the existence of this software should not obscure the fact that prototyping is as much a management strategy as it is an approach to software development. One of the seminal papers on prototyping (Ref. 29) compared the specification and prototyping approaches with development teams both using Pascal – hardly a fourth generation tool! Similarly, there are instances of application generators being used in a traditional life-cycle/specification framework.

4. THE ANALYST'S TOOL KIT

Five major approaches to systems analysis were considered in the previous section. The soft systems approach emphasises the variety of legitimate views of a problem situation and uses conceptual models to contrast desired and actual states of a system. The dataflow diagram central to structured systems analysis provides a way of looking at the system from the point of view of the data and in doing so provides a powerful technique for determining the logical content of flows, processes and stores. The traditional approach emphasises the flow of control and the activity and arrangement of the current operational system. The data-centred techniques provide important analysis and design insights, particularly when linked to a data dictionary. Finally, both participative system design and prototyping stress the needs and role of system users, with the latter emphasising the quick delivery of an initial system.

We feel that the five methods are essentially complementary. Discussions about which is best seem rather fruitless, because the success of an approach is dependent upon so many external variables. It seems more useful to see the five approaches as comprising tools available to the analyst, who then chooses the correct tool or set of tools for a particular set of constraints and circumstances.

Thus it can be argued that the analyst should be skilled enough in all of these approaches to be able to select the tool most appropriate for the job in hand. The 'appropriateness' will vary with a number of circumstances. These are likely to include the following.

- The reason for using the model (exploration, communication, experimentation or prediction).
- The level of detail desired.
- The management style of the organisation.
- Organisational size, arrangements, hierarchy and norms.
- The nature of the system trigger.

At present we only have intuition to judge where and when each is suitable, and more empirical evidence is needed before any guidelines could be formulated. Our experience suggests that the models are most suitable for the purposes shown in Fig. 2 and at the level given in Fig. 1. We would, however, expect other practitioners to disagree. The map analogy is shown alongside to illustrate the level of abstraction of the approach and, as with the road traveller, the systems analyst will need to exploit the different approaches as appropriate. Fig. 1 does not offer a prescribed, topdown approach to systems development, merely a framework for understanding the tools at our disposal.

5. TOOLS REQUIRED

We have concentrated so far on five models current in either the practical or academic market place. However, we feel that there are still several areas of information systems activity which await adequate modelling.

5.1 Modelling the interface

The interface cannot be described simply in terms of inputs and outputs because the interface has many other characteristics (such as response time, medium, etc.). To imagine that the problem has been solved by establishing the systems (or processes at a lower level of description) is to ignore the most difficult part of the problem – how, when and where will they interface? Defining the data or information which has to flow between systems is necessary for a successful design, but not sufficient. We need a model–a language–for discussing the other attributes of interaction. A particularly difficult area is the style and location of the human–computer interface. The designer often has a bewildering range of options along a continuum, all with different technical requirements and implications for the system's users. Some opt

Tool	Primary uses of model
System model	Exploration Communication
Data flow diagram	Exploration Communication Limited experimentation
Flowcharts	Communication
Participation	Exploration Experimentation
Prototypes	Exploration Predictions Experimentation
Data analysis	Exploration Communication
Data dictionary	Communication Limited exploration

Fig. 2. Primary uses of models

to smother users, others to leave them to the vagaries of the system. In most instances the interface is determined by the designer's stereotype of the user or likely users. Most texts dedicated to examining the interface offer a checklist together with a description of the interface types – menu design, form-fill, question and answer. It is one thing to describe and develop these techniques, but it is another to use them correctly and appropriately. Some tools have been developed which help to design the dialogue. A data flow diagram can be used for this (Ref. 8, p. 117) and SYNICS[23] offers an automated tool for dialogue design. However, the interface is more than just the dialogue content. In the long run it may be that a computer-based tool offers the only effective model of the human–computer interface because it can capture the interactive nature of that area. But before this can be stated with any certainty, we need an effective diagramming tool which can capture the essence of any interface.

5.2 Controls

The data-centred view of the system includes two models–a data model and the data dictionary–which are useful in determining appropriate controls. For example, existence dependence is represented on the E–R model and referential integrity is a feature of the relational model. Data dictionaries can validate the range and format of data items. Many of these checks reflect the need to input and process data correctly, and it seems likely that future data dictionary systems will perform much of this function. More fundamental is the security philosophy of a system and the implementation of measures to control fraud and privacy and to accommodate system audits. These important facets are (if they are present at all) often crudely bolted on to the system and rarely considered as an integral part of the systems design. We are unaware of any tool to assist in security design other than a list of possible problems and solutions. A set of power tools for security design is badly needed.

5.3 Hardware and software selection

Advice in this area is usually limited to a checklist of things to look for and an unsatisfactory and general guide to quantifying costs and benefits. A number of firms have their own 'methodology' but the increasing number of clients willing to sue their advisers is exposing the cracks in the wall. Grindley and Humble[12] still remains one of the most coherent approaches, but its management aspects still appear stronger than its technical advice.

5.4 Documentation and user support

This concerns the evaluation and delivery of user support. Experience suggests that support should be tailored to individual users and not roughly aimed at a supposed 'user group'. Information systems professionals normally rely upon technical instruction and training, usually manifesting itself in voluminous manuals. However, as Stamper[22] pointed out over a decade ago, there are alternative, often richer ways of imparting information.

Similarly, on-line help facilities are often disappointing, frequently displaying an uncanny resemblance to the relevant pages in the manual. Learning modes seem a step in the right direction, but a lot still remains to be done in this area.

These four areas represent information problems which we feel are still poorly modelled by existing techniques. It is not a comprehensive list; we feel there are important gaps in at least three more areas–implementation, feasibility studies, small business computing–but it gives an indication of the work that still has to be done.

6. AUTOMATING SYSTEM DEVELOPMENT

It is not the purpose of this paper to discuss the long and relatively undistinguished history of computer-aided systems analysis and design. Suffice to say that we expect

parts of all of the five current model areas to have significant computer-based design aids within the next decade. Some already exist – decision table preprocessors, dataflow diagrammers, data dictionaries, etc., whilst others are undergoing development. The adoption of these tools will be another task for the systems analyst and designer. However, the fact that the tools may be automated does not invalidate our argument. In fact the over-use of inappropriate tools may be aggravated by the presence of automatic variants, which have to be used to justify the money spent on them. An excavator may dig a hole faster than a spade, but this does not help if the hole is in the wrong place.

7. CONCLUSIONS

Whilst we have some well-tried and proven tools for systems analysis, they do not cover the spectrum of tasks which the analyst has to undertake. It is not sufficient to provide a list of guidelines, we need effective modelling techniques which can be quickly learned and successfully applied. There is an increasing need for computer-assisted systems analysis and design, but before we can automate tools there must be a thorough understanding of the range of facilities which is required. The professional analyst should have an extensive tool kit at his disposal. The D.I.Y. analyst may make do with a subset of these, but he must have something effective.

In this paper we have provided a review which demonstrates that useful tools to deal with the analysis and design of databases, processes and dataflows already exist. What we lack are tools to handle the location and type of interfaces, controls and support. The problems of implementation are also poorly covered.

The recent trend of developing competing methodologies is hampering progress towards effective systems analysis. We feel that it is unlikely (if not impossible) that a single methodology could prescribe how to tackle the great variety of tasks and situations encountered by the systems analyst. The approach presented here of developing a tool kit for the analyst – using techniques where and when they are appropriate–provides much more flexibility for dealing with the diverse applications of computers to business and other problems. The training of the analyst should focus on the variety and selection of tools, so that appropriate methods may be used in different problem situations. The desire to produce 'one best way' is leading to elaborate and bureaucratic methodologies.

REFERENCES

1. R. N. Maddison, *Information System Methodologies*. Wiley Heyden, Chichester (1983).
2. P. Hammersley *et al.*, New approaches to systems analysis and design. *The Computer Journal* **23** (1) 2–33 (1980).
3. A. T. Wood-Harper and G. Fitzgerald, A taxonomy of current approaches to systems analysis. *The Computer Journal* **25** (1), 12–16
4. T. W. Olle *et al.*, *Information Systems Design Methodologies: A Comparative Review*. North-Holland, Amsterdam (1982).
5. T. W. Olle *et al.*, *Information Systems Design Methodologies: A Feature Analysis*. North-Holland, Amsterdam (1983).
6. B. Raphael, *The Thinking Computer: Mind Inside Matter*. Freeman, Oxford (1976).
7. P. Checkland *Systems Theory, Systems Practice*. Wiley, Chichester (1976).
8. T. DeMarco, *Structured Analysis: System Specification*. Yourdon (1980).
9. B. Lee, *Introducing Systems Analysis and Design*, vols 1 and 2. NCC publications (1979).
10. D. R. Howe, *Data Analysis for Data Base Design*. Edward Arnold, London (1984).
11. E. Mumford, F. Land and J. Hawgood, A participative approach to the design of computer systems. *Impact on Society* **25** (3) 235–253 (1978).
12. K. Grindley and J. Humble *The Effective Computer*. McGraw-Hill (1973).
13. P. Dearnley and P. Mayhew, In favour of system prototypes and their integration into the systems development cycle. *The Computer Journal* **26** (1), 36–42 (1983).
14. B. Wilson, *Systems: Concepts, Methodologies and Applications*. Wiley, Chichester (1984).
15. J. Martin and C. McClure, *Diagramming Techniques for Analysts and Programmers*, Prentice-Hall, Englewood Cliffs, N. J. (1985).
16. S. Beer, *The Brain of the Firm*. Wiley, Chichester (1981).
17. D. Millington, Structured systems analysis and design using standard flowcharting symbols. *The Computer Journal* **24** (4), 295–300 (1981).
18. D. Miller, Strategic data analysis. In *Data Analysis in Practice*, edited Simon Holloway. *Proc. BCS Database Spec. Group, Leeds* (1985).
19. C. C. J. Rosenquist, Entity life cycle models and their applicability to information systems development life cycles. *The Computer Journal* **25** (3), 307–315 (1982).
20. LBMS, *The Structured Development Strategies* (Course Notes) (1984).
21. M. Jackson, *System Development*, Prentice-Hall, Englewood Clifs, N. J. (1983).
22. R. Stamper, *Information*. Batsford, London (1973).
23. E. Edmonds and S. Guest, The 'SYNICS' user interface manager. In *Human–Computer Interaction – Interact' 84*, edited B. Shackel. North-Holland, Amsterdam (1984).
24. E. Mumford, Defining system requirements to meet business needs: a case study example. *The Computer Journal* **28** (2), 97–104 (1985).
25. A. T. Wood-Harper *et al.*, *Information Systems Definition: The Multiview Approach*. Blackwell, Oxford (1985).
26. J. D. Naumann and A. M. Jenkins, Prototyping: the new paradigm for systems development. *MIS Quarterly* **6** (3), 29–44 (1982).
27. J. M. Kraushaar *et al.* A prototyping method for applications development by end users and information systems specialists. *MIS Quarterly* **9** (3), 189–197 (1985).
28. J. Connel and L. Brice, Rapid prototyping *Datamation*, pp. 93–100 (15 August 1984).
29. B. Boehm *et al.*, Prototyping versus specifying: a multiproject experiment. *IEEE Transactions on Software Engineering* **SE–10** (3), 290–303 (1984).
30. R. F. Lobell, *Application Program Generators–A State of the Art Survey*. NCC (1984).