Short Notes

Comparing Conceptual Models and Data Flow Diagrams

In 'Towards a tool kit for the systems analyst', Benyon and Skidmore discuss various approaches to systems analysis and include a comparison of De Marco's data flow diagrams and Checkland's conceptual models. This note compares the two in greater detail and concludes that the underlying methodologies are much further apart than Benyon and Skidmore suggest.

Received November 1987

1. Introduction

A recent article by Benyon and Skidmore[1] described various methods and techniques of systems analysis, using the metaphor of a tool kit. As part of this, they compared Data Flow Diagrams (DFDs), as developed by De Marco,[2] and Conceptual Models (CM), part of Checkland's Soft Systems Methodology (SSM)[4] in the following terms. 'The DFD and the activity model of the soft systems approach have enough common ground for the techniques to be linked together' (p. 3).[1] and concluded 'The similarities between the approaches... enable[s] 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' (p. 3).[1]

There are, however, significant differences between the two, and this note is intended to outline both similarities and differences in a more detailed way and to dispute the claim that a DFD is to be preferred to a conceptual model. To make the contrast sharp, the note deals strictly with the style of DFD as described by De Marco, although it is recognised that in practice DFDs may have developed in different ways.

An example of a DFD, taken from De Marco's book, is shown in Fig. 1 together with what is called its context diagram. A CM based on the same situation is shown in Fig. 2, although it is rather artificial to create a CM without a concept, as expressed in a root definition, from which to derive it.

2. Data Flow Diagrams

De Marco's work is mainly concerned with the analysis phase of a project, and here DFDs play a central role. Analysis is seen by De Marco as part of an overall methodology which, in overview, involves first a survey or feasibility study of the possible benefits and constraints of the project, secondly a structured analysis of the existing information system and changes to it. Thirdly, a more detailed structured design of the system together with a study of the necessary hardware and finally, implementation of the system (p. 26).[2]

Looking in more detail at the analysis phase, it involves the following. First, the present situation is studied and depicted in a Current Physical DFD which describes *how* the information system works currently. This DFD will include, for example, individual departmental or personal names as processors of information. The Physical DFD is then converted into the Current Logical DFD which shows *what* is done, stripped of the particular way it is done. It is one of these which is shown in Fig. 1. Any new requirements, identified in the survey stage, are then incorporated to produce a New Logical DFD which can then be used to create a number of possible realisations of this in different New Physical DFDs. The best of these is then selected and used to form a structured specification which is the basis of the design stage (pp. 27–31).[2]

Looking in more detail at DFDs, we can see

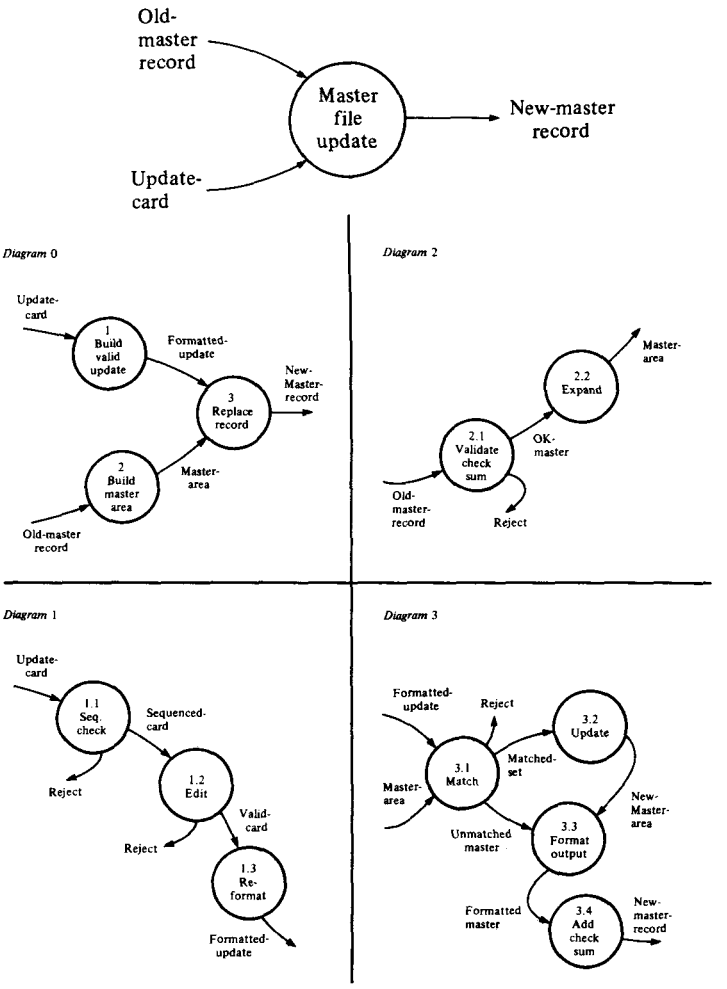

Figure 1. Data Flow Diagram. (This diagram is reproduced from Tom De Marco, Structured Analysis and System Specification © 1979, p. 73, reprinted by permission of Prentice Hall, Inc.)

Downloaded from https://academic.oup.com/comjnl/article/31/4/376/380591 by guest on 09 April 2024

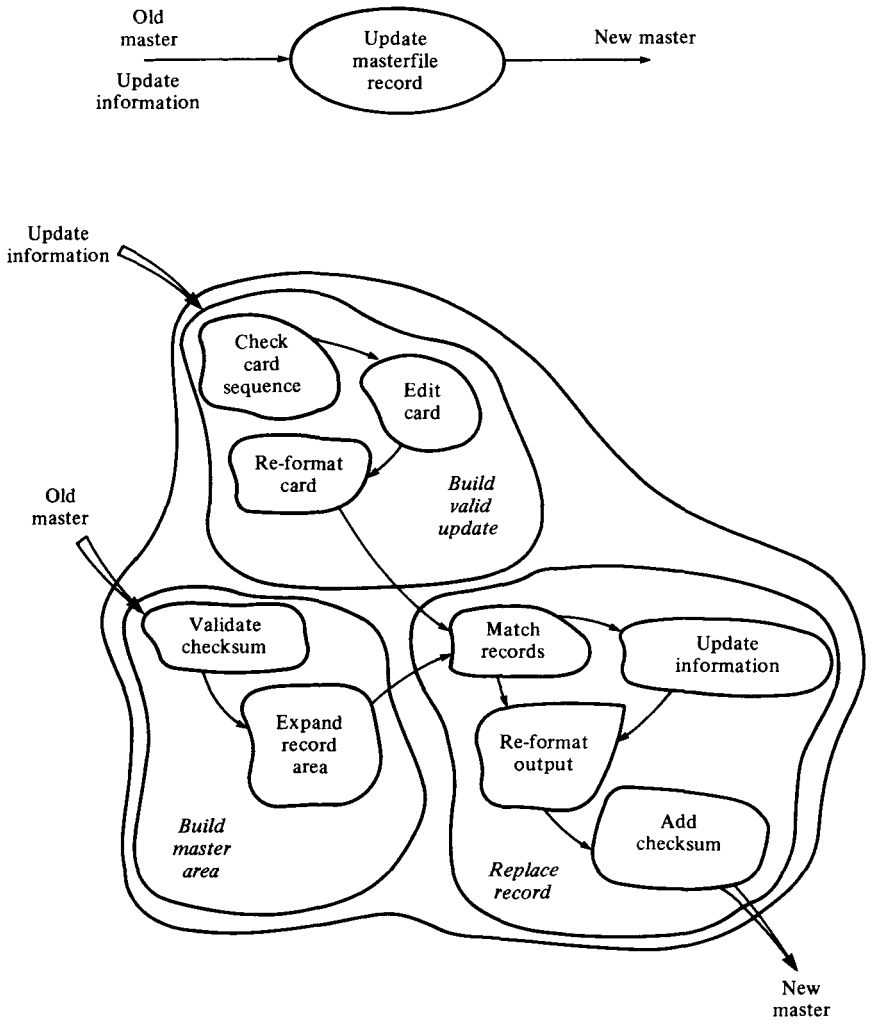

Figure 2. Conceptual Model.

from diagram 0 (Fig. 1) that a DFD consists of blobs and arrows – the arrows showing flows of data and the blobs the processes which convert the input flow into the output flow. Two other elements can be shown in a DFD although they do not appear in this example – files which are seen as temporary stores of data, and sources/sinks which are the origin and destination of the data.

The processes, being activities, should all be verbs and De Marco stresses that both the flows and the processes should be accurately named. Each process can then be decomposed into a lower-level DFD showing that particular process in more detail – so diagrams 1, 2, 3 show processes 1, 2, 3 respectively. The whole forms what is called a levelled DFD. Consistency between the levels is emphasised as in, for example, a correspondence between their inputs and outputs. The context diagram is really the top-level diagram showing the overall input(s), transformation and output(s) of the system.

In constructing a DFD, primary importance is placed on the data flows rather than the processes – it should depict the situation *from the data's point of view*. The arrows are seen as pipes through which packets of data flow, and these are laid out first. The processes are then the transformations necessary to convert the input into the output. DFDs are not supposed to show *control* as in a flow chart, nor any operations of the people processing the data, merely the possible paths taken by an item of data.

3. Conceptual models

The SSM[4] is a general methodology for improving problem situations in which the CM plays a central role. Briefly, after developing a rich picture of the situation, various ideas or concepts of potential relevance are explored by way of root definitions (RDs) and CMs. Whilst these are born out of the actual situation, their exploration is done in an essentially logical or conceptual way, removed from reality, in order to form a contrast with the real world. This contrast forms the basis for a discussion, among the actors involved, of possible beneficial changes.

The RD is a concise but detailed description of a system relevant to the problem. This definition is then expanded into a list of the activities which such a system must *necessarily* perform. The activities will all be verbs describing *what* must be done, not *how* it could be or is implemented. The list of the minimum necessary activities and the logical relations between them are drawn in a diagram. These relations show only the logical ordering of the activities – i.e. which go in sequence and which may carry on in parallel. They do not particularly show flows of materials, information, control, etc., although any of these may be the reason why one activity must be completed before another.

Having produced a first-level CM showing the minimum necessary set of activities, further levels of resolution can be obtained by treating each of the activities in turn as a named system and generating an RD and set of activities. These can be shown on the same diagram, as in Fig. 2.

4. Similarities

Clearly there are many obvious parallels.

(i) First, there is an overall diagram showing the basic input, transformation and output of the system, and both emphasise that this must be consistent – the input must logically be capable of being converted into the output. De Marco does not consider this diagram vital, merely saying:

'Drawing a context diagram may seem like a trivial formality. It serves only one purpose ... to delineate the domain of our study... I propose, however, that you take the extra five minutes to draw the (admittedly redundant) Context Diagram' (p. 75).[2]

The SSM would place more importance on this diagram, as it shows the central activity of the conceptual system being developed.

(ii) In the main diagrams, the blobs are

Downloaded from https://academic.oup.com/comjnl/article/31/4/376/380591 by guest on 09 April 2024

activities or processes of transformation and so must be described by verbs. Both emphasise that the logical consistency of the diagram is important. *All* the blobs should be activities described by verbs, unlike many systems diagrams which indiscriminately mix all manner of categories, and it is important that the names accurately reflect the transformation of input to output – difficulty in naming activities or flows usually indicates a poorly conceived model.

(iii) In a DFD, the arrows represent flows of data as if they were actual pipes with chunks of data flowing through them. They are named 'to represent not only the data which moves over the pipeline, but also what we know about the data' (p. 54).[2] In a CM, the arrows are more general, representing the *logical relations* between activities – i.e. the dependence of one activity on others. Such dependence might, of course, be because one activity requires the information produced by another, so that a CM could be made specific to a set of information-processing activities and their flows. Both diagrams, however, show logical flow, *not* a flow of control as in a flow chart. 'A data flow is not a representation of a flow of control... another thing a data flow is not is an activator of a process' (p. 54).[2]

(iv) Both involve the same top-down decomposition approach. Beginning with the top-level diagram, each bubble can be split up into its own set of activities and flows. These can be shown either in different diagrams or the same one. In the SSM this process of splitting up an activity is helped by defining a root definition for each higher-level activity and then expanding this into its own set of minimum necessary activities. There is no equivalent approach in DFD, although guidelines are offered (p. 82).[2] Both stress that the lower-level diagram must match up logically with the rest of the higher-level diagram. Thus the inputs and outputs of a lower-level diagram must correspond exactly with its parent activity.

(v) Finally, both emphasise developing models which depict *what* happens logically, rather than *how* it happens to be realised in a particular situation. In some ways this is the heart of the SSM – the logical development of a concept away from the actuality of the situation in order to provide a contrast or comparison. As Checkland states:

'In order to service this debate [about potential changes – JM] it should be clear that the "below the line" models of some human activity systems are *not* models of "what is" in the real world. Real world manifestations of human activity are of a quite extraordinary richness and complexity. "Below the line" a model will contain a structured set of activities which expresses with great purity a particular view of the system named in the Root Definition' (p. 42).[6]

Similarly, after describing the current situation in a physical DFD, De Marco's next step is to

'"logicalize" our model of the current environment... we remove the physical checkpoint items, replacing each one with its logical equivalent... A particular implementation of policy is replaced by a representation of the policy itself. The underlying objectives... are divorced from the methods of carrying out those objectives' (p. 28).[2]

5. Differences

Despite the undoubted similarities, there are fundamental differences, most of which stem from the methodologies as a whole and their respective philosophies.

(i) In De Marco's DFD the flows of data have primacy, the activities are merely those processes which convert the input flow into the output flow. In SSM, in contrast, it is the activities which play the central role. The development of a CM begins by specifying a list of the activities necessary for the named system to be as described. If information is involved, it would be that produced by, or necessary to support, the activities. This seems much better as a means of justifying the particular data flows. It must surely be the case that the activities of a system come first – it is, after all, the activities which it carries out that are the whole reason for the organisation's existence. A company exists to make and sell products, not to transform form PT 11 into form OY 76. To the extent that this is lost sight of in actual organisations, problems develop, and to formalise this displacement of ends by means in a methodology is surely to demand trouble.

(ii) The DFD, and De Marco's whole approach, is based on a description of the current situation *as it is* even though it is expressed in logical rather than physical terms. It appears to take for granted that it is largely the current system (warts and all) that is to be reproduced, albeit with some extra facilities. Indeed, it explicitly limits itself to the process of *analysis* and has very little to say about the *design* of systems or the generation of agreed objectives. It offers no help in addressing questions such as: Why does the system exist as it does? Is it the best system? Is there agreement about what the system should be doing? What if this is an entirely new development with no existing system? It may be argued that logical DFD *could* be entirely new, but there is little guidance or help in De Marco's method for developing one.

A CM, on the other hand, is not intended as a model of what exists, nor even a model of what *ought* to exist, but a model of a concept or notional system that may be helpful in agreeing beneficial changes. It could be that the concept *is* intended to match what is there (Wilson's approach[3] recommends developing an agreed model of the 'primary task' of the organisation or department) but it could equally well be something radically new. What is important, however, is that it is the result of the development of a single concept from a relevant system through an RD to a CM, and as such is likely to provide a much more coherent and consistent model than the reproduction of the inevitable vagiaries of the historical contingencies of the existing system.

(iii) DFDs take a strongly objectivist stance towards the data. In common with the data analysis approach in general, it assumes that there is only one way of looking at or organising the data. Indeed, that the data contains its own unique and unambiguous structure. In contrast to this, it can be argued that *data by itself* is intrinsically meaningless. It only becomes *information* when it is invested with meaning by people. Information is not therefore an objective fact, but dependent on the meanings and practices of the people, organisations and societies which use it. This is a deep philosophical issue beyond the scope of this article (for a useful discussion see Klein and Hirschheim),[5] but it is the case that the basic philosophy of the SSM is one that takes this subjectivist viewpoint seriously.

(iv) Most generally, the whole SSAD approach essentially restricts itself to defining problems as 'how' problems and then producing well-structured technical solutions. It is a hard as opposed to a soft systems approach, assuming a well-defined problem, perceived in the same way, and agreed on by all involved – including the operators of and users of the system – which merely requires an efficient solution. Whilst it is admitted that the politics and culture of the organisation are important, De Marco explicitly (and honestly) recognises that they are ignored in this approach.

'Of course, analysis is an intensly political subject... But most political problems do not lend themselves to simple solutions... Political problems aren't going to go away and they won't be "solved". The most we can hope for is to limit the effect of disruption due to politics' (p. 13).[2]

It is precisely this which the SSM recognises and is designed for... to attempt to bring rationality to bear on politics (in its broadest sense). As is being realised, computer systems exist as part of human activity systems in which a complex and often contradictory pattern of perceptions and meanings have to be structured and organised in order to achieve successful change. A failure to recognise this explains the lack of success of many computer systems.

6. Conclusions

DFDs and conceptual models do have a number of strong resemblances both in what they portray and in how they portray it. However, there are also major differences, stemming mainly from the different philosophies of the methodologies in which they are embedded. The points outlined above suggest that the Conceptual Model, developed within the SSM, provides a much more flexible, consistent and realistic approach to the analysis of computer systems. Benyon's conclusions that they are more or less the same, with the DFD to be preferred because it is diagrammatic, seem to stem from some mistaken ideas. First that CMs are merely lists rather than diagrams. In fact a CM is most centrally a diagram, as a look at the published work would show.[6,7] Secondly, a lack of awareness of the gulf separating the philosophies behind the two approaches.

Having said that, however, DFDs are more specialised towards computer systems, with for example files, sources/sinks and the links into data dictionaries and structured English. This can be very valuable in the more detailed stages of analysis and design, and so a well-constructed and detailed Conceptual Model might usefully and easily be converted into a DFD.

J. MINGERS

School of Industrial and Business Studies, University of Warwick, Coventry CV4 7AL.

References

1 D. Benyon and S. Skidmore, Towards a tool kit for the systems analyst. *The Computer Journal* **30** (1), 2–7 (1987).

Downloaded from https://academic.oup.com/comjnl/article/31/4/376/380591 by guest on 09 April 2024

2 T. De Marco, *Structured Analysis and System Specification*. Yourdon, New York (1980).
3 B. Wilson, *Systems: Concepts, Methodologies and Applications*. Wiley, New York (1984).
4 P. Checkland, *Systems Thinking, Systems Practice*. Wiley, New York (1981).
5 H. Klein and R. Hirschheim, A comparative framework of data modelling paradigms and approaches. *The Computer Journal* **30** (1), 8–15 (1987).
6 P. Checkland, Techniques in soft systems practice, part 2: building conceptual models. *Journal of Applied Systems Analysis* **6**, 41–49 (1979).
7 P. Checkland, Achieving 'desirable and feasible' change: an application of soft systems methodology. *Journal of the Operational Research Society* **36** (9), 821–831 (1985).

Information Systems Development: A tool kit is not enough

In the paper 'Towards a tool kit for the systems analyst', Benyon and Skidmore[3] make a useful contribution to the present debate concerning information systems methodologies.

Received October 1987

The paper develops the taxonomy work in Wood-Harper and Fitzgerald,[23] incorporating the important later work of Mumford[15,16] on participation and Dearnley and Mayhew[7] and others on prototyping. Benyon and Skidmore later suggest that the systems analyst, rather than follow a particular methodology, in effect produces a unique method for every project. This is an interpretation of information systems development which is independently supported by our own action research, and argued, with reservations, in Avison and Wood-Harper[2] and evidenced in Wood-Harper.[21] The flexibility provided by a 'cut and paste' approach to developing information systems has many attractions which Benyon and Skidmore do well to emphasise. However, there are issues that ought to be raised in the debate, and in particular we consider a tool kit an insufficient basis on which to develop information systems.

The classification of five approaches. It is particularly difficult to classify information systems development methodologies (see Ref. 1 for a discussion of the problem). However, Benyon and Skidmore's classification omits general systems theory, expert systems, planning approaches and formal methods altogether, and prototyping is not regarded as a separate approach. The paper of Cookson[6] provides support for the inclusion of general systems theory. However, even if it is conceded that general systems theory may not be considered as a separate approach (it is frequently argued that it provides only a philosophical basis for a methodology) its influence is pervading in information systems methodologies. It is perhaps too early to suggest expert systems as an approach to developing information systems, even if methodologies which are expert systems-based are likely to be evident in the next few years. However, planning approaches such as Business Information Analysis and Integration Technique (BIAIT), described in Burnstine,[5] IBM's Business Systems Planning (BSP), described in Martin,[12] and ends/means analysis[19] are important because they emphasise top management needs in information systems development. The omission of formal methods as an information systems methodology is hardly unusual, indeed the present authors would readily admit to avoidance measures here. The area is not one where practitioners of systems analysis adapt easily. However, extensions of Vienna Development Method[11] and Rigorous Method[4] could broaden its present sphere of influence wider than that of software development. There is also a good case for looking at prototyping and also the related theme of automation as a separate approach or separate approaches. The MetaSystems product,[18] for instance, is much more viable than the original ISDOS products of the 1960s. Further, although Dearnley and Mayhew perceive prototyping in their 1983 paper as being an added sophistication to the traditional life-cycle approach,[7] and Benyon and Skidmore regard it as part of the participative approach,[3] it is also being used as a separate methodology (with or without participation), if that is the 'acid test'. The later paper of Mayhew and Dearnley[13] discusses many alternative prototyping approaches. Should prototyping not then have at least equal status to that of participation, which *can* be used as a separate approach (as in ETHICS),[16] but in our view is an important aspect of any sensible approach? People ought to influence their own working lives and should influence decisions made, whatever approach to information systems development is adopted.

The necessity to consider the underlying philosophies. Benyon and Skidmore, in their analysis of various approaches,[3] do not consider the underlying assumptions and philosophies which we believe are important. The five approaches discussed in Benyon and Skidmore, and other approaches to information systems development, are (or should be) more than mere sets of techniques (or tools and techniques). Indeed, it was a discussion of the various viewpoints embodied in the approaches that enabled the six approaches in Wood-Harper and Fitzgerald,[23] three within a science (or reductionist) paradigm and three within a systems paradigm, to be identified. The philosophy may be implicit rather than explicit, but it represents an important aspect of each approach. We are particularly surprised that Checkland's work is featured in a discussion of structured systems analysis and design. The philosophies of the two approaches are in our view very different. Aspects of both may be included in the same framework, but not at the same stage, and they are not equivalent. Conceptual models and data flow diagrams are described in Benyon and Skidmore as being equivalent. They certainly have resemblances but, as Mingers has shown,[14] they are very different because of their underlying philosophies. Adjectives such as 'technical', 'computer systems orientated', 'data and data flow orientated' and 'claims to be objective' could be used to describe data flow diagrams. Adjectives such as 'conceptual', 'systems orientated', 'activity orientated' and 'overtly subjective', respectively, are more appropriate to conceptual models. It is particularly interesting that Benyon and Skidmore's paper in *The Computer Journal* is followed by one that does look at these fundamentals in the context of the data-modelling approach (Klein and Hirschheim)[10] and this describes, through a discussion of the underlying philosophy, the appropriateness of various data-modelling views.

Weaknesses of tool kit approach. The idea of a tool kit for the systems analyst is a useful contribution and does counteract the 'one best way' which Benyon and Skidmore argue leads to 'elaborate and bureaucratic methodologies'. But there are problems with this approach, many of which are readily identified by management services people in the 'real world'. Benyon and Skidmore's 'D.I.Y. analysts', as with the 'tradition and common sense analysts' (described in Veryard)[17] that operated before, are likely to produce idiosyncratic and unmaintainable systems of variable value. The tools themselves are useful, and they might be appropriate in different situations, but choosing which tools are appropriate (and when they are appropriate) is a very skilled job, and those with these skills are few and far between. The Multiview approach recorded in Ref. 22, which Benyon and Skidmore mention, is a contingency approach offering alternatives, but the tools and techniques described have to hang together within a framework, in this case the Multiview framework. As Wood asserts,[20] we must establish 'general principles by which we can choose and develop tools, techniques and methods for information systems development'. He emphasises the need for an epistemological and methodological foundation – we return to the philosophical aspects! Perhaps we will have to wait for the development of an expert systems approach, which *might* eventually be sophisticated enough to dictate when and which tools are appropriate. In the meantime, we suggest that the tool kit has to be used within a framework such as those described in Ref. 8 or within a rather looser framework such as the 'explorative framework' described in Ref. 2. A reflective practitioner, who may not want to follow a rigid methodology, still needs to use a *coherent* approach.

Despite the above comments and criticisms of the paper, the work of Benyon and Skidmore belongs to the movement in information systems which reflects fourth-generation languages, prototyping and participation and which will, perhaps, break systems analysis away from the rigid strait-jacket with which many of the present-day methodologies restrict systems analysts. This limits both the practitioners and the usefulness of the information systems developed. Every situation is different, and analysts should have the opportunity to explore and create a unique method for each situation, but this can only be achievable within an information systems definition framework, with the analysts aware of the underlying philosophies and assumptions.

D. E. AVISON*, G. FITZGERALD† AND A. T. WOOD-HARPER‡
*Department of Computer Science, Aston University, Birmingham, B4 7ET; †School of

Downloaded from https://academic.oup.com/comjnl/article/31/4/376/380591 by guest on 09 April 2024