STRATEGIC DECISION SUPPORT SYSTEMS: AN EXPERIENCE-BASED APPROACH

SUWANNA RASMEQUAN CHRIS ROE STEVE RUSS

{suwanna,croe,sbr}@dcs.warwick.ac.uk

Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

Abstract

The paper describes the application of a novel, experience-based approach to modelling, known as Empirical Modelling (EM), to the challenging task of building effective strategic decision support systems (SDSS). As a case study a restaurant management model is described which illustrates how EM can be applied to issues which are characteristic of SDSS such as imprecise problems, the need for end-user development and 'negotiation modelling'. The three concepts of observable, dependency and agency are fundamental to EM. They lead to a distinctive approach to computers as instruments capable of embodying our knowledge of a domain and giving us experiential feedback from models which can be directly compared with the results of interaction with the world. This is the basis for claiming that EM offers a new quality of human-computer interaction which in turn allows for a much enhanced openness and flexibility in systems such as SDSS where the collaboration of human and computer-based processes is crucial.

Keywords: Decision support systems, human-computer interaction, modelling, experience

1.0 INTRODUCTION

Computer-based systems to support decision making abound in elaborate functionality but they are often difficult to use effectively, and are therefore often not used. In this paper we address this problem indirectly by first describing an unconventional approach to modelling being developed with some success at the University of Warwick. Previous work has often focussed on areas rich in interaction (design, games, graphics etc). But it is becoming clear that it is the quality of interaction itself, that is promoted by our approach, that is very significant and renders our approach promising for applications which depend on the close integration of human and computer processes. We introduce here briefly the methods to be used and the target application.

What is Empirical Modelling about ?

All programming is, in some sense, modelling. The data and algorithms of a program represent entities of some kind, whether abstract or concrete, and ways of processing those entities. But the converse is not true. Much modelling – for example, mathematical or physical modelling – does not amount to programming in any conventional sense. When an engineer uses a wind tunnel, or a motorist consults a road map, they are appealing to the reliability of the model to inform their expectations about future interactions with the world. Such expectations point to the familiar mental, or cognitive, modelling that is characteristic of human consciousness and again this internal, personal model-making seems far removed from programming. The approach to modelling described in this paper is unusual in being close to human, mental modelling and yet being computer-based. It is more primitive than conventional programming but in principle can be used to extract conventional 'programmed' systems from our models once they are validated from our experience as reliable and satisfying some requirement. The approach is based on the concepts of observable, dependency and agency, but instead of referring by these terms to public, objective entities these notions are regarded initially *from the modeller's own perspective and interpretation.* The view of the domain being modelled is therefore personal, subjective and provisional. The process of model development depends upon comparison of the model observables with the real-world observables, through extensive interactions with the model and with the world. It is this emphasis on observation and experiment that gave rise to the name 'Empirical Modelling' (EM) for this style of using the computer. It has broadened into a wider understanding of computation. A 'computer' in EM's sense refers to any reliable, interpretable, state-changing device. With this meaning in mind, EM takes the context in which a computer is operating to have an equal significance to the computer itself. 'Programming' in EM's sense is broadened to include not only algorithms and data but also the configuration of the system of users, devices and processors. EM is a novel approach to computation that has been developed by Dr Meurig Beynon and his collaborators at the University of Warwick, UK [1] since 1983. The principles of EM have philosophical roots that have much in common with the work of William James [2].

The EM approach offers a distinct concept in modelling. With a traditional approach, the modeller has to preconceive what are going to be the inputs and outputs before starting the construction of the model in order to prescribe the processes or behaviour required. That is, the boundary of the proposed system must be determined in advance. If there is a need for a new input or output, e.g. an additional system feature is required, then the whole system may have to be revised and re-designed at substantial cost. With the use of EM, any new observable or property of the model can be added in at any point during the modelling process through re-definition without the need to revise, or even re-start, the whole process. Part of the reason for this flexibility is that in EM we are focussing initially on the *state* of a model or domain, rather than on the behaviours we wish to produce. Changes of state are regarded as resulting either from 'law-like' causes integral to the domain (e.g. 'whenever I move my shadow moves'; 'I've won the game because I have three X's in a row'), or from the actions of some agent (e.g. 'I decide to move now'; 'she puts an X in that square'). In the former case, the relationship is described as a *dependency* and is expressed in a definition which, like a spreadsheet formula, is updated automatically by the modelling tools that we have developed. In the latter case we must identify an *agent* with associated *observables* (state) and privileges for actions to change state by re-definitions.

During the course of model development it is likely that new observables, dependencies and agents will arise apart from those initially expected. These new perceptions are the product of experimentation with the model and give insights to the user, or they may introduce the user to a new view or idea of the situation or problem being considered. In this way the process of model building proceeds in tandem with the enrichment of the user's own conceptual model of the domain. The scripts we build may contain explicit propositional knowledge but the rich interaction possible with the artefact represented by such a script offers the engaged user experiential and tacit knowledge of the domain.

We have envisaged the potential need for 'negotiation modelling' where many people may interact with a model and with each other. There is now a distributed version of our main modelling tool that allows for various modes of communication between users. Using this tool we have successfully modelled the collaboration between train drivers and signalmen in the context of an historic railway accident [3]. This could support the widespread practice these days of the collaboration of different staff in an organisation from different parts of the world; this is considered to give a major competitive advantage to the firm. The combination of conceptual and computer-based modelling that our methods offer would make such collaborations even more advantageous.

What are Decision Support Systems about ?

Decision making is a central part of business practice at every level of management. In a classic text by Simon [4] the three phases of decision making are identified as intelligence, design and choice. He points out that these are closely related to the phases of problem solving described by the philosopher John Dewey: What is the problem? What are the alternatives? Which is best? In neither case are the phases necessarily distinct, and a decision support system (DSS), from the earliest work on such systems in the late 1960s, sought to address all three phases. The very term decision support system emphasises the collaboration expected between human and computer processes. An early pioneer of DSS, Scott-Morton, writes that the DSS approach calls for a strategy 'for meshing the analytic power and data processing capabilities of the computer with the manager's problem-solving processes and needs' [5]. A more recent definition of DSS has a similarly wide scope and assumes human involvement and integration:

'Decision Support Systems are computer-based systems that bring together information from a variety of sources, assist in the organisation and analysis of information, and facilitate the evaluation of assumptions underlying the use of specific models.' [6]

A DSS is thus a system in which the human user plays an essential, interactive role.

Sutherland [7] classifies three levels of decision making: strategic, tactical and operational. The strategic level has to do with formulating and choosing among alternatives that are different in kind ('qualitatively disparate'). This makes the integration of the human and computer-based processes, and the qualitative nature, of an EM approach particularly suited to a strategic decision support system.

Strategic decision support systems (SDSS) present three additional challenges of their own:

(i) strategic problems are typically imprecisely described and qualitative;

(ii) strategic decisions are likely to be taken by the most senior managers and so ease of use, or the possibility of end-user development, is highly desirable;

(iii) strategic decisions are likely to be shared among managers and so require a distributed system (i.e. a group DSS).

In addition, according to [5], 'A DSS is more a service than a product. Since the problem can only partially be structured and since managers grow in their understanding and needs over time, a DSS must constantly grow and evolve as the user adapts and learns.' Models built in an EM environment are well-suited to having this quality of continuous adaptation. The remaining sections give further details to support our proposal that the EM approach offers a promising new direction for the development of SDSS.

2.0 A RESTAURANT MANAGEMENT MODEL

Empirical Modelling is a collection of principles, tools and techniques for an alternative approach to system development. EM offers an unusual environment where the way in which a system is developed begins with a very open-ended analysis of a domain. This involves building an artefact that enables a high degree of interaction and the visualisation of many features of the domain thought likely to be of interest to the modeller or a potential user. During extensive experimentation the artefact may be refined, and certain behaviours circumscribed and automated, to optimise the usefulness of the resulting 'system'. Typically, there will be many ways in which human users may continue to intervene and modify the system to adapt to unforeseen, changing requirements. The way in which the initial artefact is developed follows closely the way in which people naturally make sense of their surroundings and applications. We suggest that the ideas of observable, dependency and agency are, in some form, fundamental to all such 'sense making' activities.

By way of illustration we sketch here the development and features of a model relating to restaurant management. The example was prompted by the first-hand experience of the first author in a particular restaurant. It is not intended to support full strategic decision making for restaurant managers (whether to open new branches, change cuisine offered, etc). The purpose here is only to indicate briefly how EM methods support the properties required of a SDSS, namely the capacity to cope with imprecise, qualitative problems, to be amenable to end-user development and to offer distributed access to several users.

An important daily decision making activity for the restaurant manager, which we focus upon here, is the preliminary allocation of bookings to tables, followed by the real-time adaptation of this allocation in response to customer preferences, telephoned booking requests and cancellations, and walk-in arrivals. Relevant observables clearly include the number of bookings for a particular evening, number of people for the booking, time of booking, customer preferences for seating (window seat, privacy alcove etc), profile of occupancy, experience of previous similar evenings. There is a dynamic timetabling problem here in which both the requirements and the resources are changing over time. There are numerous dependencies involved which become even more complicated if tables may be joined together in some circumstances to accommodate larger groups. Customers (and staff) will be acting as agents in unpredictable and unforeseeable ways.

There is a decision making model in the manager's mind which more or less consciously determines the decisions on table allocation. This model has four main components: the level of income (correlating approximately with maximising occupancy over an evening), the level of customer satisfaction, depending on such factors as quality of food, service and atmosphere, in relation to cost, the level of staff morale and performance, and the past experience of the manager in successfully balancing the optimisation of these levels. The aim of the EM model described here is to complement and support the manager's own conceptual modelling.

The model developed shows a simple visualisation of the layout of the restaurant with a number of different-sized tables (see Figure 1). Below the table layout is a schedule showing the booking information for the restaurant for a particular evening. Each slot is labelled with a customer name. To the right are forms that can be edited to simulate customers booking or cancelling a table. These

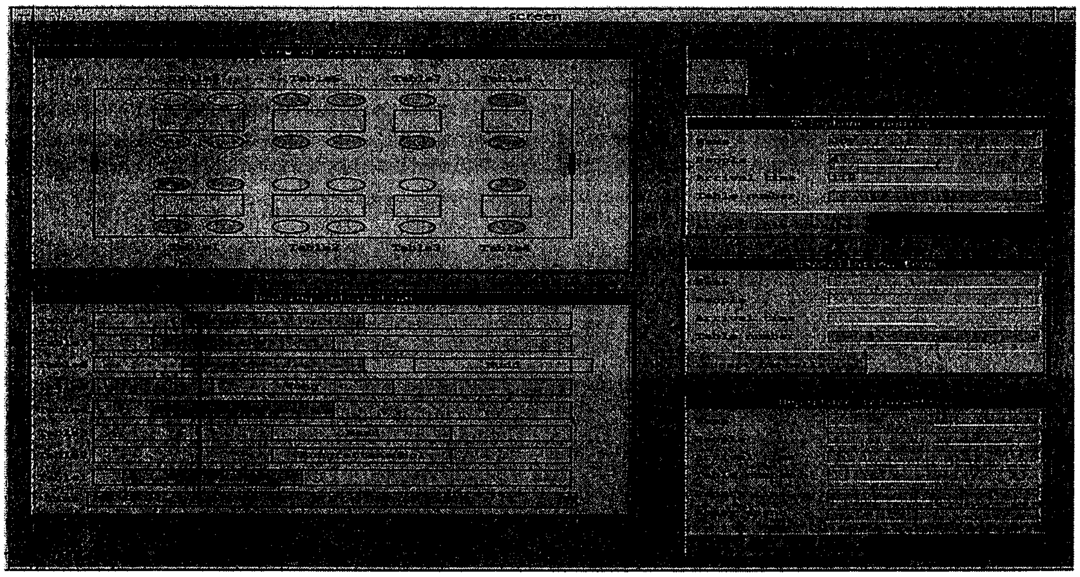

FIGURE 1. Interface to the Restaurant Management Model

can be entered manually and the resultant changes will be effected on the schedule. In the top right-hand corner is a clock so that the simulation can be animated. When it is turned on the events of the evening progress with a bar moving across the schedule to show the current time. Each time a customer enters the restaurant their slot highlights on the schedule and their group is highlighted arriving at their table. When they leave the restaurant, their slot returns to normal and the highlight dims from their table. The data from their visit, including arrival time, length of stay and amount of money spent can be recorded for later use. At random times in the simulation customers may enter the restaurant, or telephone for a booking or cancellation. The frequencies of such events can be altered to create busy or quiet evenings as required. When an enquiry is received the user can allocate a table for the customer, reject them, or let the computer choose an appropriate table, using a re-definable algorithm.

We now consider in the following paragraphs the extent to which the model displays the desirable properties for SDSS of dealing with imprecise, qualitative problems, being suitable for end-user development and supporting several users as stakeholders in decision making.

The raw data for an evening's bookings are quantitative. But the manager is likely to use the data to imagine some scenarios when there are quiet times, and times when there is likely to be congestion. The visualisation offered, though crude, supports such an overview of the occupancy pattern since the changes over (say) five hours can be surveyed in animated fashion over (say) twenty seconds. While each booking has only numerical data the overall profile of occupancy and movement takes on a qualitative aspect which is reliably maintained in such a computer model and supports human imagination. In particular, the flexibility of the visualisation allows for rapid interaction and experimentation with different allocations of tables. (Many parameters are defined by dependencies, so if major features are altered, such as the length of the working evening, length of booking slot etc, the integrity of the display is maintained.) It may be necessary to make many allocations, and booking decisions, 'on the fly'. Where there are larger groups concerned and a variety of ways tables may be joined to accommodate them, the model may give the manager support in exploring the combinatorial possibilities and in reducing 'wasted time' between bookings at a particular table, while optimising occupancy by maintaining flexible availability for different group sizes.

Our modelling tools are still basically research tools – they illustrate our principles but do not have the robustness and consistency required for a commercial product. Nonetheless the potential for end-user development is suggested, (a) by the fact that this model was constructed (by the second author) in only a matter of days, (b) by its unusual open-endedness, and (c) the close engagement required from the user in development, and the principles themselves, make for a high degree of comprehensibility. We have developed here only the dynamic timetabling aspect of managing bookings. Even in this area, we would expect to be able to deal with unforeseen issues – such as addition of new tables, some tables going out of use, areas of the restaurant being unusable due to re-decoration, introducing e-mail bookings, etc – by relatively simple re-definitions. There are many additional aspects of restaurant management – management of staffing rotas and payroll, menus, suppliers, marketing etc – that we believe could be developed with EM tools, in principle though not yet in practice, by a manager.

The problems for end-users of integrating programs which perform specific functions (e.g. a statistical package and word-processor) are well known. Our experience so far suggests it is relatively easy in our framework to integrate models built for quite different purposes as and when the need arises. A few weeks prior to building the restaurant model a tool for the dynamic visualisation of data sets in very flexible ways was built by the second author. (See Figure 2. The tool is modelled on Attribute Explorer developed by IBM UK after an idea due to Prof. Bob Spence of Imperial College, London.) It allows a user to enter a data set (say, restaurant customers in the last six months) and display different fields (say, amount spent, length of stay, arrival time and number in group) in separate windows. The distribution of each field is represented as a bar graph composed of blocks each corresponding to a data item. The tool allows the user to set constraints on one or more fields and those items satisfying all the constraints are highlighted. We can interactively edit the constraints through the use of scrollbars. The speed of feedback during interaction offers a qualitative sense of the (possible) relationships between the quantitative data. The manager could explore whether a group twice the size of another usually spends twice as much. Do groups arriving earlier in the evening stay longer, and spend more? Does this depend on the group size? Such issues may be part of the folklore of experienced managers. Using reliable data with these rapid visualisation techniques may give rise to suggested correlations that could then be checked using statistical packages. There are innumerable possibilities for exploring data in this way. The point here is that we cannot *foresee* the tools a manager might find helpful. EM offers an environment where it is reasonable to expect that any useful components from one model can be integrated with another model. That is a major benefit for end-user development.

The first requirement for any corporate strategic decision making is a thorough understanding of the present performance and the strengths and weaknesses of the current business processes. We believe a good way of achieving that, and allowing discrepancies and conflicts in viewpoint to emerge, is to build as faithful a model of current practice as possible. In such a model the different human agents should retain their identity and partake in their normal human role as an agent within a distributed computing environment. This way we preserve their insight into what actions and judgements they make, while avoiding the immense difficulties of 'automating' people. We have had some experience of modelling co-operative work in the context of developing requirements for a warehouse management system [8], where most of the human agents retained their human role in the model. We have also experimented with a classroom model [9] where the user was a teacher exploring effective scenarios for praising and punishing (automated) pupils with certain crudely modelled characteristics. We have not extended the restaurant model in this way yet but can envisage doing so with a view to better mutual understanding of current processes. Among the uses of such a model might be exploration of different protocols for the allocation of waiting staff to tables, for the taking of orders and making of bills etc, and the training of staff and managers.

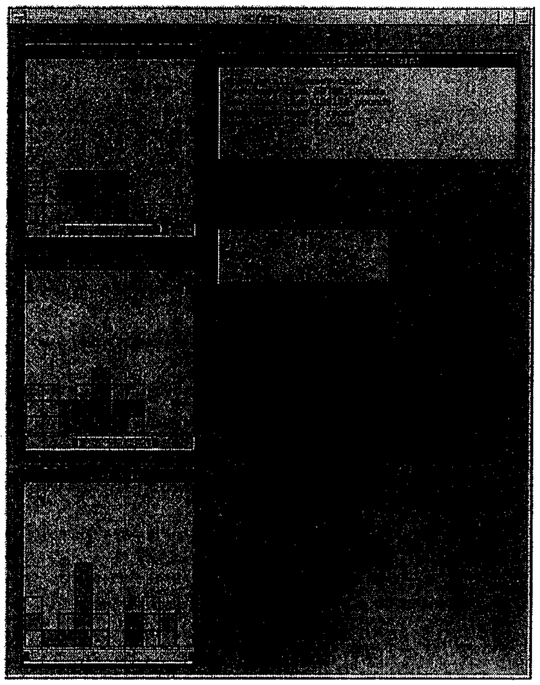
FIGURE 2. EM version of Attribute Explorer

We have outlined here the development of part of a simulation model for restaurant management, namely that part concerned with bookings and their allocation to tables. We emphasise it is the nature of the development process that matters, not any details of functionality in the product so far. This process is fundamentally different from the process of writing a conventional piece of software, due to the quality and role of interaction with the model and the cumulative accrual of experience gained through refinement of it.

A good analogy might be to compare the EM approach to modelling to that of a crafts-person working with a physical medium. The work is very open-ended, engaging and enjoyable, guided by the medium itself, the worker's own vision and the application in mind. Many conventional approaches have more of the pre-conceived quality of a production line item in a factory. The production process is tightly regulated and human involvement is limited. Unexpected behaviour is likely to cause expensive crashes and there is no scope to take advantage of unforeseen opportunities.

3.0 THE EM APPROACH IN PRACTICE

The way in which a model is constructed in the EM approach depends on construing a phenomenon through three basic perceptions: observables, dependencies and agents. These perceptions are based upon the modeller's observation and interpretation of the domain or, in other words, the modeller's personal experience of the domain. The initial account of a domain identifying what seem to be the relevant agents and observations is organised and expressed in a systematic but informal notation called LSD. This LSD account represents an initial analysis and understanding of the domain, it is revised and elaborated during the model development but it remains a textual record and guide to aid comprehension and construction.

It is not essential to give an LSD account of the domain to be modelled (it was not given explicitly in the restaurant model) but the observables and dependencies that belong to such an account are basic ingredients in the scripts of definitions that are interpreted by our main modelling tool. Each definition is either a value definition or a formula definition and these work as they do in a spreadsheet. So

$$x \text{ is } y-2z; \quad y=12; \quad z=5;$$

is a fragment of a definitive script in which the definition for x represents a dependency automatically maintained by the interpreter. At present x will have value 2, but this will change as y or z change. We think of the values of y and z, and the dependency of x on them, as the observed state from the point of view of the modeller, not simply as a set of three abstract values. Thus the 'observable' that Jason is a *potential_project_leader* may depend on all, or many, of a number of criteria being satisfied in the opinion of his manager. Such a dependency could become part of a definitive script concerning Jason. Changes of state occur in the model only through re-definitions or the addition of new definitions. Of course, a single re-definition may trigger the automatic updating of many other variables in a large script. Groups of re-definitions may be bundled together as an action of an agent, usually with some real-world semantics (such as the re-drawing and re-printing of the organisation charts after Jason is made project leader). The representation of the dependencies (on the right hand sides of definitions) is handled in our work by user-defined functions using a C-like language. These three language features – *definitions* with dependency maintenance, user-defined *functions*, and *actions* – are provided in a notation Eden and its associated interpreter together with a Tcl interface. Other definitive notations, for example, Donald for line drawing and Scout for window management, have been developed on top of Eden. There are some newer notations being developed based on Java but all these are purely research tools, sufficient to explore the principles of EM but far from having the robustness and efficiency required for large-scale applications.

Thus a typical step in model construction is composing a user-defined function f and introducing a definition of the form $x = f(a,b,c)$ into an existing script of definitions. From the modeller's point of view this is recording a dependency between the observables x, a, b and c. From our computing point of view it is like adding a new definition into a spreadsheet which has a visualisation attached to its cells. Because of the ease of revision of scripts and interaction with our models it is appropriate to be more relaxed about the planning and development of a script compared with the writing of a conventional program. In general, the order of definitions does not matter (though the order of re-definitions will usually matter a great deal). Throughout the construction the effects of certain interactions will show the need for new observables and reveal the need for, or refinement of, dependencies. This experimental, interactive development is depicted in Figure 3.

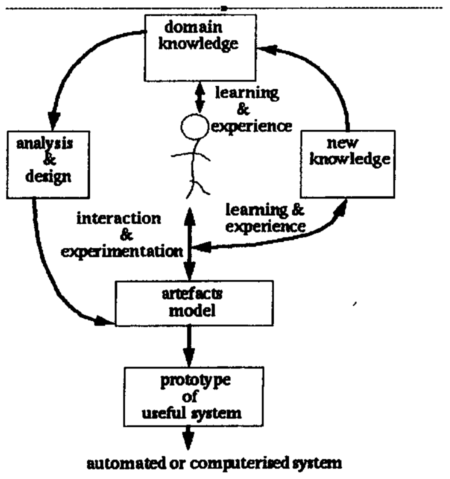

FIGURE 3. EM as a platform for developing DSS prototype

The Eden tool now has a distributed version that allows several participants in a development to interact with each other. Using a network the interaction of one participant can be sent to other participants' models and consequently affect their individual insights. All the models of participants can be connected together in such an environment (in a variety of controlled communication modes) thus making a powerful collaborative forum for the sharing of understanding and gaining of consensus.

While most of our research effort has gone into developing models in a very open-ended and exploratory fashion we recognise the need, in order to derive useful systems, for appropriate circumscription of our models once a desired functionality can be relied upon. We regard such systems as being derivable from our models in many ways. What is common to such derivations is that the boundary of the eventual system is not pre-conceived, but rather grows with the developing understanding of the modeller – both understanding of the domain and the requirements for the system. In a conventional system development the boundary must be planned in advance of development work, and the influence between developer and system is largely one-way: the functionality is planned and imposed on the artefact produced. In EM the influences between modeller and artefact are two-way, the modeller expects to gain new insights from the developing model and these will affect future development. The diagrams in Figures 4 and 5 illustrate this idea.

There are many technical resources (such as the Eden Handbook) and further details of the notations and numerous models available from our website. [1]

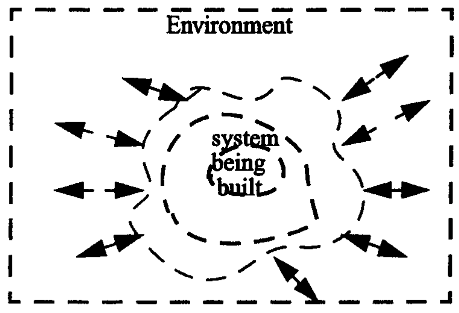

FIGURE 4. EM system development

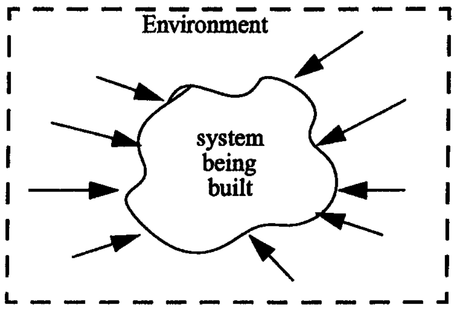

FIGURE 5. Conventional System Development

4.0 THE EM APPROACH FOR BUSINESS MODELLING

In section 2.0 above we have described a simple model which shows the potential of our tools for application to business modelling. Here we emphasise that the methods of EM arise from a real shift in perspective on the computer, not simply from additional ingenious techniques to add new functionality. We describe how EM generalises the spreadsheet principle and how this generalisation points to a view of the computer as an instrument or artefact by which we can render our understanding of a domain in an experiential fashion. We argue that this leads to an unusual quality of interaction that is well suited to business modelling and DSS.

The spreadsheet has been one of the primary business tools for decision support. EM generalises spreadsheets in a way that has significant practical implications. Spreadsheets have typically been applied to structured problems where experimental interaction is constrained to 'what if?' scenarios whose nature is largely pre-conceived. The development of EM artefacts is, in contrast, a more open-ended exploratory activity. This section explains how this generalisation can address the ill-structured problems that arise in business. It can also be interpreted as a paradigm shift from a mode of closed-world modelling in the scientific tradition, to a style of modelling better suited to the demand of soft computing and social science.

An EM model is more general than a spreadsheet in the following three major respects: presentation, underlying algebra and agency.

The presentation of an EM model is more elaborate in that the model can be presented using pictures, graphs and sound in whatever way best suits the application. For example, in the Vehicle Cruise Control Simulation (VCCS) depicted in [10], simple line drawings are used to represent how the positions of the accelerator and brake affect the speed of the vehicle. The core dependencies amongst the key observables (such as determine the relationship between the acceleration, speed and position of the vehicle) could easily be maintained in a spreadsheet. But, whereas the data presentation in a spreadsheet is confined to values, labels and formulae in the grid cells in the tabular interface, the drawing of the speedometer within the VCCS illustrates a more general kind of representation. Such pictorial presentation may be found more understandable by the user. For example, the state of the economy might be better represented to a child by a sad, or smiling, face than by a balance of payments chart.

The data types and operators that underlie the spreadsheet (its 'underlying algebra') are typically very simple in character. The underlying algebras of EM extend simple scalars and character strings by including recursive lists, and representations for 2-d and 3-d geometry and screen layout. This gives scope for much richer forms of dependency to suit the modeller's applications. For instance, the attributes of a window such as size, location and colour can be dependent on their content, and this content can be dynamically linked to geometric and textual data that is turn determined by other dependency relations.

The concept of agency in EM enables dependency relationships to be manipulated concurrently by several agents, both human and automated. In contrast, the user of a spreadsheet is typically the only agent that can change the data or formulae. Several existing EM models resemble distributed spreadsheets where different people working on different workstations can activate and change the components of the model. This could supply a suitable platform for Group Decision Support Systems.

Commercial software applications include substantial extensions to the simple spreadsheet concept. Such additional functionality as sophisticated graphical output, statistical tools, optimising tools and 'live feeds' to cells in the spreadsheet are now customary. But such extensions do not affect the underlying outlook of 'batch processing', i.e. pre-conceived input/output patterns. The key significant idea in spreadsheets – state change through dependency and agency – has not really been taken up seriously in conventional software, although there are signs that is affecting interface design. The EM approach does aspire to make this view of state change central and one consequence is to take the physicality of computers seriously. The computer is both a powerful mathematical machine and at the same time it is able, through its various peripheral devices, to embody and give experiential form to our models. A visual and auditory representation of flowing lava, for example, will make more immediate sense to a user trying to understand the behaviour of a volcano than will the numerical data generated from a set of mathematical equations, essential of these are for other purposes. While giving this sort of experience of the real-world is a feature of the physicality of the computer it is easily misunderstood. Its value in the modelling context is not so much for the entertainment, as for the engagement, of the user: it is to prompt the user into systematic, purposeful interactions with the model to compare the behaviour with the real-world domain and further their understanding and the faithfulness of the model.

The physicality and performance of modern computers, coupled with this sense of engagement by the user, allows for a quite new quality of interaction between user and computer. It is open-ended in the sense that the experience of the model may agree with, or contradict, the user's expectations in very detailed and subtle ways. It is like a learning process or a conversation, calling for continuous engagement, rather than the detachment with intermittent attention of conventional computer interaction. It is holistic in supporting and requiring a combination of graphics, visual effects, text, mathematical abstractions and dynamic interaction and in this way it matches human cognitive processes closely.

This quality of interaction, which is promoted and supported in the EM approach we have been describing, is particularly significant for dealing with areas, such as business modelling, where there is little known theory and where human agency plays a major role. What we said above about the enhanced functionality of spreadsheets also applies to DSS. As described in [11] a modern DSS comprises numerous components and elaborate functionality. But while the architecture for interaction is dominated by the batch-mode mentality the real difficulties in interaction cannot be addressed. The quotations in section 1.0 on DSS point to the fundamental need for better integration of human and computer processes. It is because we believe that EM represents an approach to computing that makes for a new kind of human-computer integration that we suggest it here as a promising framework for a new approach to DSS development. The conversational and cognitive qualities of interaction in EM make it particularly appropriate for addressing the properties detailed in section 1.0 concerning SDSS.

5.0 CONCLUSION

The EM approach which we have described here is broad and far-reaching in its scope. It seems to have considerable potential in the area of business applications, particularly for ill-structured problems where human intervention and opportunism will clearly add value compared with any conceivable fully automated system. Other active areas of application and research showing encouraging results are business process re-engineering, requirements engineering, financial modelling, timetabling, and various topics in the field of education. The potential for use of our tools and methods in distributed mode is huge, but the major limitation at present lies in the tools themselves. They are in need of overhaul and development in many ways but we have very limited resources for this work.

Acknowledgement : The authors are pleased to acknowledge the inspiration, insight, encouragement and wise advice received from Meurig Beynon at all stages of preparation of this paper.

6.0 REFERENCES

[1] http://www.dcs.warwick.ac.uk/modelling

[2] W. James, *Essays in Radical Empiricism* (London: Longmans, Green & Co, 1912)

[3] P.H. Sun, Distributed Empirical Modelling and its Application to Software System Development, *Ph. D. Thesis*, Department of Computer Science, University of Warwick, UK, 1999

[4] H.A. Simon, *The New Science of Management Decision* (NewYork: Harper & Row, 1960)

[5] P.G.W. Keen and M.S.S. Morton, Decision Support Systems: An Organizational Perspective (Reading: Addison-Wesley, 1978)

[6] V. L. Sauter, *Decision Support Systems: An Applied Managerial Approach* (NewYork: John Wiley & Sons, 1997)

[7] J.W. Sutherland, *Towards a Strategic Management and Decision Technology* (Dordrecht: Kluwer, 1989)

[8] P.H. Sun, Y.C. Chen, S.B. Russ, W.M. Beynon, Cultivating Requirements in a Situated Requirements Engineering Process, *Research Report 357*, Department of Computer Science, University of Warwick, UK, 1999

[9] E.L. Davis, Modelling Human Interaction, *Project Report*, May 1996, Department of Computer Science, University of Warwick, UK

[10] W.M. Beynon, S.B. Russ, Empirical Modelling for Requirements, *Research Report 277*, Department of Computer Science, University of Warwick, UK, 1995

[11] M.R. Klein, L.B. Methlie, Knowledge-based Decision Support Systems (2nd edition) (Chichester: John Wiley & Sons, 1995)