A New Paradigm for Computer-Based Decision Support

Meurig Beynon, Suwanna Rasmequan, Steve Russ

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

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

Abstract

We identify and address a fundamental general problem which we regard as crucial for the widespread, effective use of decision support systems (DSS) in the future: how can we substantially improve the quality of interaction, and the degree of flexible engagement, between humans and computers? Rather than seeking an answer in additional technical functionality we propose a new paradigm for computing that is human-centred and that adopts a novel, observation-oriented approach to data modelling. We report recent practical work (a timetabling instrument) showing an unusual degree of openness for interaction, and evidence that our models can significantly generalise expert systems.

Keywords: Decision Support Systems, Expert Systems, Modelling, Observation, Dependency, Agency

1.0 Introduction

The history of DSS over the past 30 years has seen the opportunistic exploitation of every kind of technology that could be brought into service: spreadsheets, databases, networks, hypermedia, expert systems, visual programming, intelligent agents, neural networks etc. Yet in spite of enormous technical advances over these years in storage, speed, functionality of tools and even interface design, the improvement to be seen in the effectiveness and extent of DSS usage has been relatively modest. There seems to be a significant pattern operating here. On the one hand, applications where useful interactions between human and computer are simple and they can largely be preconceived - such as word processing, accounting software, e-mail and web-browsing - have escalated in use and sophistication very rapidly. On the other hand, applications demanding a high quality of interaction because of human factors, the role of the physical context, or the unpredictability of the environment, for example, have made much slower progress. Such applications include those for engineering design, learning environments and requirements engineering in addition to business modelling or DSS. There is a *prima facie* case here that a fundamental problem facing computer applications is that the quality of interaction, and the degree of integration possible between human and computer activities, is affected very slowly, if at all, by technological advance. Many different approaches have been proposed to alleviate this problem [6,12,18].

In this paper we approach this fundamental problem from a computing perspective that is rather unconventional: we are not so much advocating new technological solutions but rather proposing the adoption of a different paradigm for the use of computing technology. Despite a number of signs of change, contemporary views on computers and programming still seem dominated by a reliance on logic and formal methods in constructing and interpreting applications. Such a logicist outlook affects thinking about data and models, and tacitly discourages human interaction. For example, in a recent work on DSS where the importance of model formulation for exploring solutions is emphasised, it is taken for granted that these will be 'logical, mathematical models' (p.140 of [8]). Logicism in computing at large may have been reinforced in earlier decades by an association with techniques for essential optimisation and efficiency. This has left a legacy that is now anachronistic. It is, therefore, ironic that today as much as when it was written in 1960, few would consciously doubt the remark made by Simon that, 'Many, perhaps most, of the problems that have to be handled at middle and high levels in management have not been made amenable to mathematical treatment, and probably never will.' (p.21 of [16]) The paradigm which we propose in this paper is human-centred and based on concepts of *observable*, *dependency* and *agency.* It is thus very different from a logicist outlook but much more primitive conceptually and capable of embracing formal concepts rather than challenging them.

2.0 A fundamental problem for Decision Support Systems

In a classic paper by Mintzberg et al. [11] the process of decision making is conceived in terms of three major components: problem identification, development of alternative solutions and selection among the alternatives. The computational paradigm that best matches this characterisation of decision making is that of logic programming [9]. Traditional expert systems are an example of this, where the problem to be solved is formulated as a sophisticated logical constraint to which possible solutions are generated by a systematic search procedure, and the results are filtered by applying suitable algorithmic criteria. In the development of the expert system concept, it has long been recognised that human problem solving activity presumes a fuzzier framework than precise quantitative data and exact reasoning supplies. This paper is rooted in more radical reservations about a logicist perspective on decision support. These stem from a perceived need to develop an approach to computer data representation fundamentally different from that underlying conventional programming paradigms. (See [2] for a more detailed discussion of these issues.)

The perspective of the human expert on the logicist framework for problem-solving provides helpful motivation for our approach. By way of illustration, consider the task of manually constructing a timetable for a class of students each of whom is to give an oral presentation to a specified group of staff. Though it is possible to give an abstract characterisation of a feasible timetable, and indeed to write a computer program to generate a timetable that meets every prosaic constraint concerning availability of staff and students, this is not what the human expert understands by solving the timetabling problem. The expert has many tacit expectations about what qualities the timetable can have, and these expectations will be tempered by a process of exploration that attempts to construct a solution with these qualities. If the expert chances upon many possible solutions, or is frustrated by a lack of solutions, her evaluation criteria may consequently be revised. Though the coverage of the solution space may be far less comprehensive than that offered by an automated procedure, there will be no counterpart for the expert's personal understanding of those parts of the solution space that have been visited. In constructing and contemplating possible solutions, the expert will have explored the relationship between the abstract assignment of symbols to slots and its real-world referent in ways that are highly situation-specific, and can neither be preconceived nor automated. Her evaluation may indeed reflect the subjective judgements of all the staff and students involved. In some circumstances, the expert's knowledge of the timetable will show itself in an ability to make adjustments in response to changes in requirements following unexpected events.

The issues raised by this illustration can be abstractly summarised as follows:

Problem identification: An expert will typically not know initially how to identify, or formulate, a problem in a circumscribed way. The problem will typically be presented in broad, imprecise terms pending exploration of the solution space, and the assessment from past experience of possible changes in the environment.

Developing alternative solutions: The expert may have no abstract or systematic method of finding solutions. They will recognise solutions when they have seen and explored them, rather than apprehend them as specified by a declarative constraint. What is judged to be a satisfactory solution will depend highly upon the specific problem-solving situation, and upon how skilful, fortunate and conscientious the expert has been in their exploration of the situation. They will know what to do next only when they find themselves in a situation. The problem-solving activity is typically guided by what is encountered, as it is encountered.

Selecting solutions: The expert may not have any explicit heuristic for evaluating and selecting solutions. The criteria applied in evaluation may be qualitative or impossible to preconceive. A good solution will be apprehended not as an abstract entity with intrinsic qualities, but in its relation to the real-world scenario to which it refers and to its neighbourhood in the solution space. It may also be conditioned by many different person's - possibly inconsistent - evaluation of these relations.

The issues raised by the expert's perspective concern a specific abstract model of computer use in decision support. This is not to deny the value of automation in problem solving activity. Even in manual solution of a task such as timetabling, an expert can exploit automation in several ways.

Automation is useful in imposing known constraints. For example, a computer interface can be constructed to prevent or inhibit constraint violation. The computer can perform routine calculation and inference that needs no reference to the real-world referent, as in the updating of a spreadsheet, or automatic execution of slot assignment steps that are obligatory consequences of a particular scheduling action. Automated search procedures can also address the routine aspects of exploring the solution space. Computer generation of potential solutions cannot provide the in-depth understanding of a region of the solution space such as the human interpreter acquires, but it typically leads to an examination of the search space that is more comprehensive, more systematic, and deals in a more uniform way with all potential solutions. Where quality criteria can be suitably framed, evaluation can also be automated. By recording, classifying and monitoring the results of searches and evaluation, the computer can also supply global knowledge about the solution space that is otherwise inaccessible.

The above discussion highlights the need for both human and automatic processing activity in decision support. The key issue is how these activities can be effectively integrated. Only the human observer can interpret the abstract solution in relation to its referent. Only the computer can deliver the processing power to support more efficient and comprehensive exploration and analysis. Introducing a wide range of preconceived ways of evaluating solutions (as in a timetabling expert system such as [23] that examines the quality of timetabling solutions with respect to criteria such as the distances between venues) does not attack the essence of this problem. The judgements made by the human expert concern a relationship between the abstract solution and its real-world referent; they typically involve observations that cannot be comprehensively preconceived and pre-programmed.

Strategies for integrating human and automatic activity in decision making are intimately connected with the way in which the three component activities are intertwined. Though the logicist perspective suggests a clear separation of concerns between problem identification, solution development and selection, this does not typically reflect the expert's experience (cf [20]). Problem formulation is influenced by exploration of the solution space; evaluation and selection of solutions is helpful in guiding the search; the quality and abundance of solutions obtained influences expectations about potential solutions and feasible constraints. Changing the balance between human and automatic activity in decision making is often associated with revising the interaction between identification, development and selection. This is relevant to the preliminary analysis and experiment that precedes the development of a classical expert system, where expert activity in which all these various components of decision making are conflated is rationalised and simplified so that it can be emulated by systematic and routine procedures.

The limitations of current paradigms for integrating human and computer activity in problem solving are most prominent in volatile environments. In a timetabling context, for instance, it is usually necessary to adapt to changes in requirement that arise unexpectedly, either during the process of compiling a timetable and or even whilst it is in operation. For such purposes, it is essential to adapt the existing solution rather than develop a new solution - a process for which there may be no automatic support if the existing solution has been generated automatically. Such adaptation may be carried out much more effectively where human insight has played a significant role in the development of a solution, especially if the change in requirements pertain to features of the solution space that have been thoroughly explored.

Compromise is an important element in human adaptation of solutions. This may mean relaxing what were at first considered to be absolute constraints. For instance, in a timetabling scenario, if there is no other recourse, it may be that a person is required to timeshare between two slots. Fuzzy approaches to logic and data attribute the brittleness of automatically generated solutions simply to the fact that logical and quantitative data must be precisely specified, but this may be to underestimate the real challenges of integrating human and automated activity.

In seeking to find better ways to provide computer support for decision making, this paper explores the prospects for applying a new paradigm for computer use. This involves a different stance on the problems that motivate fuzzy approaches. The need for fuzziness is viewed as symptomatic of a deeper problem - an essential distinction between the treatment of observables by humans and the treatment of data by computers. The evidence for this is clear in the sharp separation that differentiates the conception and construction of a program from its execution. In program development, human imagination and knowledge of the world is paramount. In program execution, human interaction and interpretation invokes knowledge of the world in preordained and preconceived ways. There is a fundamental mismatch between *abstract data that is interpreted by the human in direct association with its counterpart in the real-world referent and situation*, and *abstract data that is manipulated according to computational rules that can only take account of prespecified and preprogrammed features of this association*. The next section examines an approach to computer-based modelling that aims to address this issue.

3.0 An Experience-based Approach: Empirical Modelling

The application which has perhaps been the greatest single influence for enhancing the quality of human-computer interaction is the spreadsheet. The interaction the spreadsheet allows, and the cognitive engagement with the user that it enables, has no conceivable counterpart within the batch mode of computation of the 1960s. Despite this, the computational paradigms of today are in their essence unchanged. The computer is still represented as supplying functions to the user, and trading output for input according to a pre-specified protocol. The key significant idea of spreadsheets - state change through dependency and agency - has not really been taken up seriously in conventional software development. The approach adopted here takes the concepts of state, and of state change, as fundamental. We are using 'state' in a broad sense to encompass states of mind in subjective experience as well as the state of a computer artefact and the state of its real world referent. Our approach also leads to a broadening of the meanings of computer and programming. A 'computer' in our sense refers to any reliable, interpretable, state-changing device; and the context in which such a computer operates is as important for the interpretation of its behaviour as the computer itself. The concept of programming then includes not only algorithms and data but also the configuring of a whole collection of components, including human agents, into a provisional system. This is such a broad concept that we have usually referred to our artefact construction as *modelling* rather than programming. And because of the emphasis on experience (particularly on observation and experiment) we describe our research as Empirical Modelling (EM). EM is a novel approach to modelling developed by Dr Meurig Beynon and his collaborators at the University of Warwick, UK since 1983 [22].

The way in which we construct an EM model depends on construing a phenomenon through three basic perceptions: those of observables, dependencies and agents. These perceptions are based upon the modeller's viewpoint and interpretation of the domain or, in other words, the modeller's personal experience of the domain. The initial statement of perceptions is organised into an account written in a special purpose notation. This account consists of an identification of the relevant agents together with a classification of the observables into those which belong to particular agents (*states*), those which an agent can respond to (*oracles*), those which an agent may control (*handles*), and those which have specific dependencies associated with them (*derivates*). This account informs the construction of an artefact (i.e. a model) that reflects the privileges that each agent has for changing variables; these are described in the protocols for each agent. The contrast between such an account of a domain and a conventional system specification may be likened to that between an artist's preliminary sketch and an architect's final drawing. The former is spontaneous and personal, a soft boundary to be continuously developed into a piece of artwork, the latter is the culmination of earlier drafts, a hard boundary not intended for revision.

The EM approach offers a distinctive kind of modelling. With its focus on state, and lack of control structures, it is more primitive than conventional programming. State change is only achieved through automatically updated dependencies or the explicit action of an agent (who may be the modeller). With a conventional approach, the modeller must preconceive the inputs and outputs before starting the construction of the model in order to prescribe the processes involved. If there is a need for new inputs or outputs, e.g. an additional system feature is required, then the whole process may have to undergo costly revision and redesign. With the use of EM, revisions can in principle be made at any point during the modelling process with immediate effect. This is because the state of a model is captured in a script of definitions that resemble spreadsheet formulae. Each definition is either a value definition or a formula definition. For example

product_variable_cost (x) **is** *production_cost - fixed_cost;*

production_cost (y) **is** *12*;

fixed_cost (z) **is** *7*;

is a fragment of a definitive script where the definition for *x* records a dependency that is automatically maintained, so that while x is currently *5*, it will change as *y* or *z* change. The '**is**' here is a keyword of the notation indicating a dependency rather than an assignment. The state of the model can be viewed as the current values of *x*, *y* and *z*, or (with more insight) as the values of *y* and *z*, together with the dependency relating *x* to *y* and *z*. We have in mind here that these values are the observed state from the point of view of the modeller, not the mathematical abstractions of numbers (though we must often resort to using such values when what we intend is something more like an analogue quantity). So the 'observable' that an employee Simon is *ready_for_promotion* may depend on all or many of a number of criteria being satisfied in the employer's opinion. Changes of state occur in the model only through re-definitions or the addition of new definitions. In a large script with many dependencies a single re-definition may trigger automatic updates to hundreds of other variables. Groups of re-definitions may be packaged 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 Simon's promotion). The representation of dependencies by definitions involves operations that include user-defined functions. These three language features - definitions with dependency maintenance, user-defined functions, and actions - are provided in a notation EDEN and its associated interpreter. EDEN incorporates a general purpose definitive notation and other definitive notations for line drawing and window management. These three notations can be used freely together for model construction. The environment for the users of EDEN offers some limited assistance for the management and debugging of scripts, but our notations are primarily research tools, sufficient to explore the principles of EM but far from having the robustness and efficiency required for large-scale applications.

In general, the order of definitions does not matter (though the order of re-definitions will usually matter a great deal). This fact, together with the ease of revision of scripts and interaction with our models allows us to be more relaxed about the planning and development of a script compared with the writing of a conventional program. Another consequence of the relatively loose structure of our scripts is that there is a potential for many streams of updating activity to proceed in parallel.

During the initial stages of script construction the effects of certain interactions will show the need for new observables and new, or refined, dependencies. The script, built up in this experimental and incremental fashion, represents the modeller's knowledge of the domain in a similar way to that in which a spreadsheet embodies some of the financial and operational knowledge of its user. At the same time the script corresponds to the state of an evolving computer model of the domain. Because of the origin of this model in personal experience it has a *situated* quality. That is, it is organically connected to its referent or domain through being continuously open to revision by comparisons between experience with the domain and experiences of interaction with the model. In recognition of this style of evolution we often call our models 'interactive situation models' (ISMs [3]).

There is now also a distributed version of EDEN that allows a variety of modes of communication between many users. This offers the potential to support collaborative modelling where many people with different viewpoints and knowledge may interact with a common model and work to achieve consensus on properties or objectives that are initially in conflict.

4.0 The principles of a new paradigm for computer support

An alternative framework for decision support

Empirical Modelling involves a paradigm shift in which experience becomes the primary and primitive ingredient. In keeping with the philosophical principles developed by William James in [7], experiences are to be interpreted through their relation to other experiences. The key idea in EM is to use computer-generated experiences as an alternative mode of knowledge representation. Such an approach is generally better suited than precise mathematical modelling to representing concepts in social science and business. The situated, experiential nature of most business problems and tasks means that they are not amenable to abstract formulation without simplifications which themselves remove the very difficulties we are seeking to address. The cognitive support for personal viewpoints that experience-based representations can provide is a particularly distinctive feature of EM. This can potentially play a significant role in giving computer-based support to the new forms of business collaboration that arise through globalisation.

The specific focus of this paper is upon substantial improvement in the quality of integration of human judgement with automated processing in decision making activity. Our thesis is that most paradigms for describing computation reflect a legacy of concern about optimisation and efficiency that has shaped and constrained their concept of computer data representation in a profound way. Significant as these concerns have been in the past, especially in the seminal early stages of computer development and in relation to the large-scale data intensive problems of decision support, the technology now exists to justify a paradigm shift that gives greater emphasis to the experiential aspects of computer state. The overall objective of this section is to show that by creating an artefact in the form of an ISM, in which human observation is closely aligned with computational state, it is possible to achieve more effective human-computer integration. The illustrative examples used for this purpose include a simple special-purpose timetabling application and a reworking of an expert elevator designer using EM tools. These show how dependency maintenance and visualisation can be used to integrate human and computer roles in problem identification, solution development and selection.

A timetabling application

The ideas behind the generation of ISMs for decision support will be illustrated with reference to a simple but authentic timetabling application. (For more details, see [5].) This application involves timetabling a week of oral presentations for final-year project students. Each presentation requires a half-hour timeslot between 9 am and 5 pm from Monday to Friday. The presentation is attended by the project supervisor, a second assessor, and a moderator whose job it is to chair the session and to arbitrate where there is a significant discrepancy between the marks awarded. In assigning staff to moderation and assessment roles, the need for staff to be suitably qualified as examiners for the specified project and the need to balance workloads are significant - and sometimes conflicting - considerations. There are generally two rooms available in each timeslot, so that two presentations can run in parallel. The number of project students is presently of the order of 120, but is currently increasing year-by-year. Our discussion of this application draws upon practical experience of two of the authors in the manual construction of a timetable.

The kind of decision making activity that is to be supported in the timetabling application has been informally described above. The timetabling agenda is much broader than that of simply deciding how given resources are to be deployed subject to specified abstract constraints. From the perspective of an expert human timetabler, there are many considerations that are appropriately addressed by human interaction, for a variety of reasons. For instance, human judgement may be essentially involved (e.g. can we make the sessions thematically coherent? is this member of staff well suited to examining this project?), or there may be specific issues that pertain to this particular timetabling problem (e.g. a key member of staff cannot be certain until Sunday whether they will be available on Monday rather than Tuesday, and the assignment of assessors to a project and moderators to students is affected by which). Though such issues may in principle admit automatic support, it is more cost effective to address them manually, especially when the advantages of gaining better understanding of the characteristics of the particular solution space are taken into account.

The key to applying EM principles to enable effective integration of human and computer activity in tackling this agenda is to identify the observables and dependencies belonging to each state encountered in the process of manual construction. For this purpose, we imagine being in the middle of constructing a timetable, having made certain allocations of students to slots, certain allocations of staff to examining roles and contemplating the next step in extending the current partial table. The impact of assigning a student to a slot will be to affect observables relating to the availability of the student, the slot and the staff involved. This effect will be indivisibly associated with the assignment action, and is an intrinsic part of what the timetabler regards as the meaning of the action. By setting up a definitive script, the way in which the effect of the action is mediated to other observables can be directly represented. Since this representation captures the functional dependency by which current staff availability is determined by the assignment of students to slots, and the assignment of staff to examining roles, the integrity of the state is preserved under other actions, such as revoking the scheduling action, or the changing the moderator for the session. A small extract from the script that can be formulated to capture these dependencies is given in Figure 1 below.

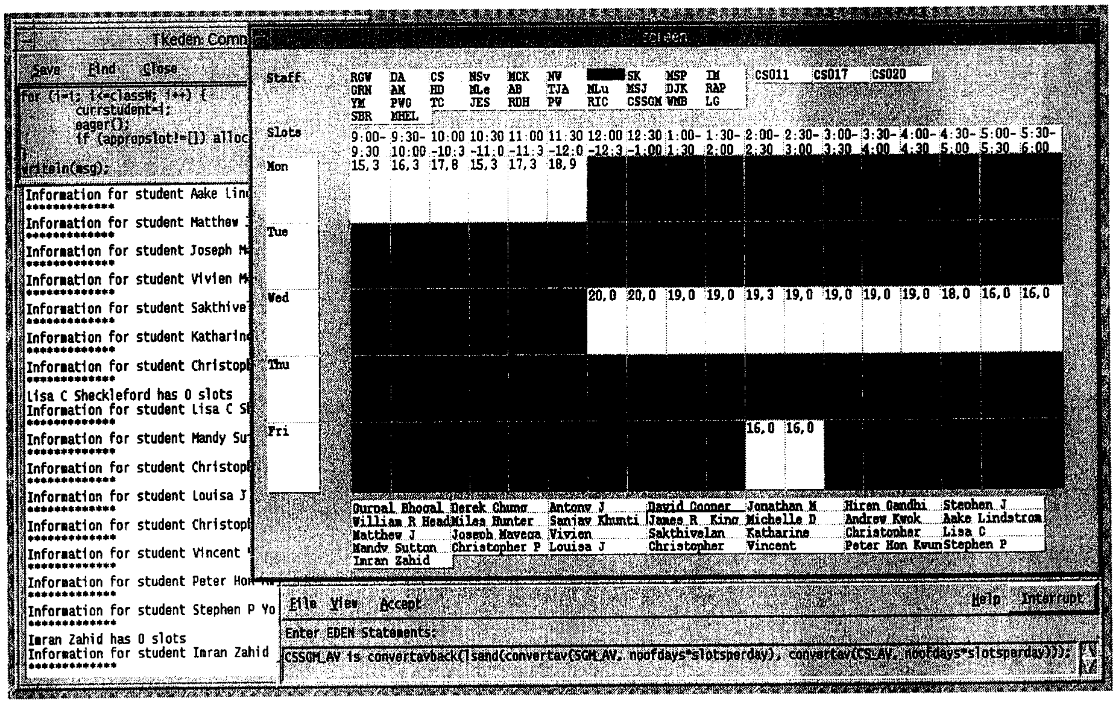
The simplest form of interaction with the script that can be used to construct a timetable involves assigning students to slots, monitoring the consequences and taking corrective action as and when this is deemed necessary. The openness of the activity that is framed in this way is in keeping with the exploratory nature of the modelling paradigm. An alternative strategy, more closely resembling the way in which a typical constraint-oriented program operates, is to revoke any assignment as soon as any conflict is introduced into the timetable. The virtue of allowing the timetabler to introduce conflicts into the evolving schedule is that this suits the human aspiration to seek solutions of a quality that pushes the constraints to their limit - a strategy that presumes negotiation between feasible solutions and imposed constraints.

The observables that concern the human timetabler are not adequately represented by prosaic abstract conditions and values internal to the computer. The perspective of the human interpreter is conceptually that of concurrently observing the state of the computer model and the corresponding state of its real-world referent. It is not the fact of maintaining up-to-date values in the computer model that - of itself - gives special support to this human viewpoint. What matters vitally is how the current state of the computer model is mediated to

```
staff is ["ABC", "OBJ", "PVC", ... , "GLC", ... ];
/* list of staff */

SAM is [["James Bond", "ABC", "PVC", "GLC"], .... ];
/* supervisor, assessor, moderator assignment */

TT is [["James Bond", 3], ... ];
/* (partial) timetable as list of (student name, scheduled slot) pairs */

ABC_AV is [1,3,4,6,8, .... ]
/* list of slots that ABC has declared as available */

avail is [ ABC_AV, OBJ_AV, PVC_AV, ..., GLC_AV, ... ];
/* declared availability of staff member */

assign is makestaffassign(TT, SAM);
/* slots so far assigned to staff members, as function of TT and SAM */

current_student is "James Bond";
/* student who is currently the focus of interest */

S is index(staff,supervisor(current_student, SAM));
A is index(staff,assessor(current_student, SAM));
M is index(staff,moderator(current_student, SAM));
/* S is identifying index for supervisor of current student etc. */

avSAM is MEET(avail[S]-assign[S], avail[A]-assign[A], avail[M]-assign[M]);
/* possible slots in which to schedule current_student */
```

FIGURE 1. Extract from a definitive script to maintain current availability of staff

the human interpreter. In adopting the term 'interactive situation model', the purpose is to focus attention on the computer model as something that contributes to the modeller's current experience, and is to be interpreted in the context of another experience - external to the computer model - to which it refers. A screenshot of the timetabling ISM is shown in Figure 2.



```
for (i=1; i<=class#; i++) {
        currstudent=i;
        eager();
        if (appropslot!=[]) alloc
}
writeln(msg);
```

```
Information for student Aake Lin
*************
Information for student Matthew J
*************
Information for student Joseph M
*************
Information for student Vivien M
*************
Information for student Sakthive
*************
Information for student Katharin
*************
Information for student Christop
*************
Lisa C Sheckleford has 0 slots
Information for student Lisa C S
*************
Information for student Mandy Su
*************
Information for student Christop
*************
Information for student Louisa J
*************
Information for student Christop
*************
Information for student Vincent
*************
Information for student Peter Ho
*************
Information for student Stephen P Yo
*************
Imran Zahid has 0 slots
Information for student Imran Zahid
*************
```

```
Staff   RGW  DA    CS  NSv  MCK  NW   SK    MSP  IM   CS011  CS017  CS020
        GRN  AM    HD  MLe  AB   TJA  MLu   MSJ  DJK  RAP
        YM   PWG   TC  JES  RDH  PW   RIC   CSSGM WMB LG
        SBR  MHEL
```

Slots	9:00-9:30	9:30-10:00	10:00-10:3	10:30-11:0	11:00-11:3	11:30-12:0	12:00-12:3	12:30-1:00	1:00-1:30	1:30-2:00	2:00-2:30	2:30-3:00	3:00-3:30	3:30-4:00	4:00-4:30	4:30-5:00	5:00-5:30	5:30-6:00
Mon	15,3	16,3	17,8	15,3	17,3	18,9												
Tue																		
Wed							20,0	20,0	19,0	19,0	19,3	19,0	19,0	19,0	19,0	18,0	16,0	16,0
Thu																		
Fri											16,0	16,0						

```
Gurpal Bhogal     Derek Chung        Antony J        David Cooper    Jonathan M   Hiren Gandhi     Stephen J
William R Head    Miles Hunter       Sanjay Khunti   James R  King   Michelle D   Andrew Kwok      Aake Lindstrom
Matthew J         Joseph Mavega      Vivien          Sakthivelan     Katharine    Christopher      Lisa C
Mandy Sutton      Christopher P      Louisa J        Christopher     Vincent      Peter Hon Kwun   Stephen P
Imran Zahid
```

```
Enter EDEN Statements:
CSSGM_AV is convertavback([]sand(convertav(SGM_AV, noofdays*slotsperday), convertav(CS_AV, noofdays*slotsperday)));
```

FIGURE 2. Timetabling screen shot

For the timetabler, the observed state of the computer model must convey information about the actual scenarios that will be experienced when the timetable is enacted. The way in which human mental processes operate is such that neither of these experiences need be entirely explicit. Where observation of the state of the computer model is concerned, the quality of the display and speed of the hardware are relevant, as is the intelligence and skill of the timetabler. It is not possible for the computer to display the current status of all staff and students on a single screen, even if the timetabler could assimilate such a volume of data all at once. It is enough that this current state can be interrogated, and that it will be presented in a way that is suited to direct apprehension by the timetabler. Similar considerations apply to the observation of the real-world referent. The timetabler can perfectly well envisage the activity that is involved in moving from one venue to another and need not enact it when planning the timetable. In so far as this should be observed in developing a good timetable, it is appropriate to incorporate some feature at the computer model interface that reflects distances between venues that must be visited consecutively. This could either involve a visual or scalar representation of distances accessible through interrogation, or be displayed on a map of the venues.

The timetable ISM is also enriched by the fact that each state of the script is open to *what if?* style interaction. In manual timetabling, each step towards the construction of the final timetable typically involves some experimentation aimed at identifying and satisfying criteria for optimisation, both qualitative and quantitative. For instance, we may ask whether it is possible to schedule all the hardware project orals on Monday morning, or whether it is possible to confine the commitments of a particular member of staff to morning sessions. Making small readjustments to each partial timetable created during the construction of the final schedule is a typical approach to tackling such issues. It is also the means to identify heuristics that can be used to speed up or refine the development process. Such heuristics can be introduced into the ISM to support both manual or automatic timetabling activity. This entails introducing new observables and dependencies into the script to monitor concerns such as how many as yet unscheduled orals can be placed in any given as yet unused slot. For automatic use of such heuristics, these observables can then be presented as oracles to agents that can carry out reassignments of slots autonomously, or at the discretion of the user. This approach to automation is significant in that it supplies computational activity that is intelligible to the timetabler; the automatically generated states of the ISM are just such as the timetabler might themselves encounter through suitable sequences of manual definition.

The primary advantage of using a definitive script to represent the current status of the timetabling activity is the exceptional richness of the state-transition models it can support. Where state information is concerned, the definitions in the script for the ISM record: all the data pertaining to scheduling; all the geometric, textual and communication elements associated with the visualisation and user interaction; and the status of all conditions that the timetabler wishes to monitor. The possible transitions between states include any introduction of a definition (or parallel introduction of several definitions) into the script. From the timetabler's perspective, the significance of such an action is entirely shaped by the relationship between the state of an abstract computer model and that of its real-world referent. A new definition may be introducing a new observable (e.g. the workload of a member of staff), making a correction to the data (e.g. interchanging the names of supervisor and assessor), or changing the state of the computer model to reflect a conceptually different state of the referent (e.g. revising the declared availability of a member of staff).

Appropriately, since many - indeed most - of the dependencies that can be introduced through definitions have no sensible external interpretation (e.g. account the contribution to the workload as negative if the student's surname begins with Q), part of the interaction with the ISM involves the negotiation of meanings. It is in this respect that the agenda that motivates fuzzy logic is addressed within our paradigm. It may be appropriate in exceptional circumstances to schedule the same member of staff in two concurrent slots, for instance, or convenient to add a dependency to link assignment in one slot to unavailability in another (e.g. to ensure that a part-time member of staff who attends an oral presentation at 9am is not deemed available after 3pm).

The management and concurrent interaction of human and automatic agents can also be addressed in the script. The state of the script can reflect the presence or absence of automatic agents, the observables that enable and provide the cues for their action, and the eagerness with which actions are performed, and express the ways in which these are interdependent and dependent on other elements of the current state. Concurrent state can be captured by distributing scripts in such a way as to reflect the viewpoints and modes of interaction of several different human agents. The state of the script can encompass observables that determine the configuration of concurrent state - for instance, whether a definition introduced into a script by one human agent is broadcast to others.

The framework that surrounds our key concepts of observation, dependency and agency is radically different from the logicist framework. This difference stems from the way in which the abstract structure of the computer model is developed in close association with experience of its referent. This establishes a relationship between 'form' and 'content' that, in keeping with Cantwell-Smith's *Two Lessons of Logic* [17], is quite unlike that presumed in the logicist perspective. Since any new paradigm for decision support should account for the success of classical expert systems, it is important to understand how such systems can be interpreted in EM terms. To this end, we now discuss how the expert elevator designer VT [10] and the subsequent development of a knowledge-based system ELVIS for elevator configuration [14] can be interpreted in terms of observation, dependency and agency.

An ISM for the elevator design problem

Elevator design (as instanced in what Rothenfluh et al. [14] identify as the Sisyphus-2 problem) is a prime example of a decision problem that is closely circumscribed. The decision making process is directed at relatively narrowly specified objectives: choosing the components from which to assemble an elevator system, whilst meeting certain user requirements, engineering constraints and safety regulations. The constraints essentially frame a routine problem in engineering design, in which there is no conceptual novelty, and decisions are concerned with making a suitable choice from pre-specified options (such as which of a given set of motors to use).

In comparing this elevator design problem with the timetabling problem discussed above, it is helpful to focus on the knowledge acquisition activity that was involved in developing VT, rather than the product itself. It is in the support of this activity, where the analysis of the heuristics used by an expert elevator designer was carried out in conjunction with the development of a computer model, that an observation-oriented perspective is most relevant. It is significant that ELVIS was subsequently derived from the original expert elevator designer VT, framed in a logicist idiom, through re-engineering that involved spreadsheet principles. Because of this prior translation, it proves conceptually quite straightforward to recast the ELVIS model into the form of an elevator design ISM implemented using the EDEN interpreter - a task first carried out by Paul Ness, Mike Joy and Simon Yung in 1995.

The script for the elevator design ISM consists of a large suite of definitions that is similar in character to the script for the timetabling ISM, as depicted in Figure 3. There are several hundred definitions that record the elevator users requirements, the dependencies between the characteristics of different engineering components, and the observables associated with monitoring whether constraints are violated. Each of the fixes for constraint violation - documented in [21] - can be interpreted as an automatic agent such as has been discussed above in connection with the timetable ISM. In the elevator design ISM, these fixes are implemented in such a way that the cues that trigger their invocation are maintained in the script, and the user is presented with a menu indicating precisely which fixes are currently commended for possible execution. This creates an environment in which the modeller has discretion over which fixes are triggered, and so can explore the space of possible solutions that is accessed

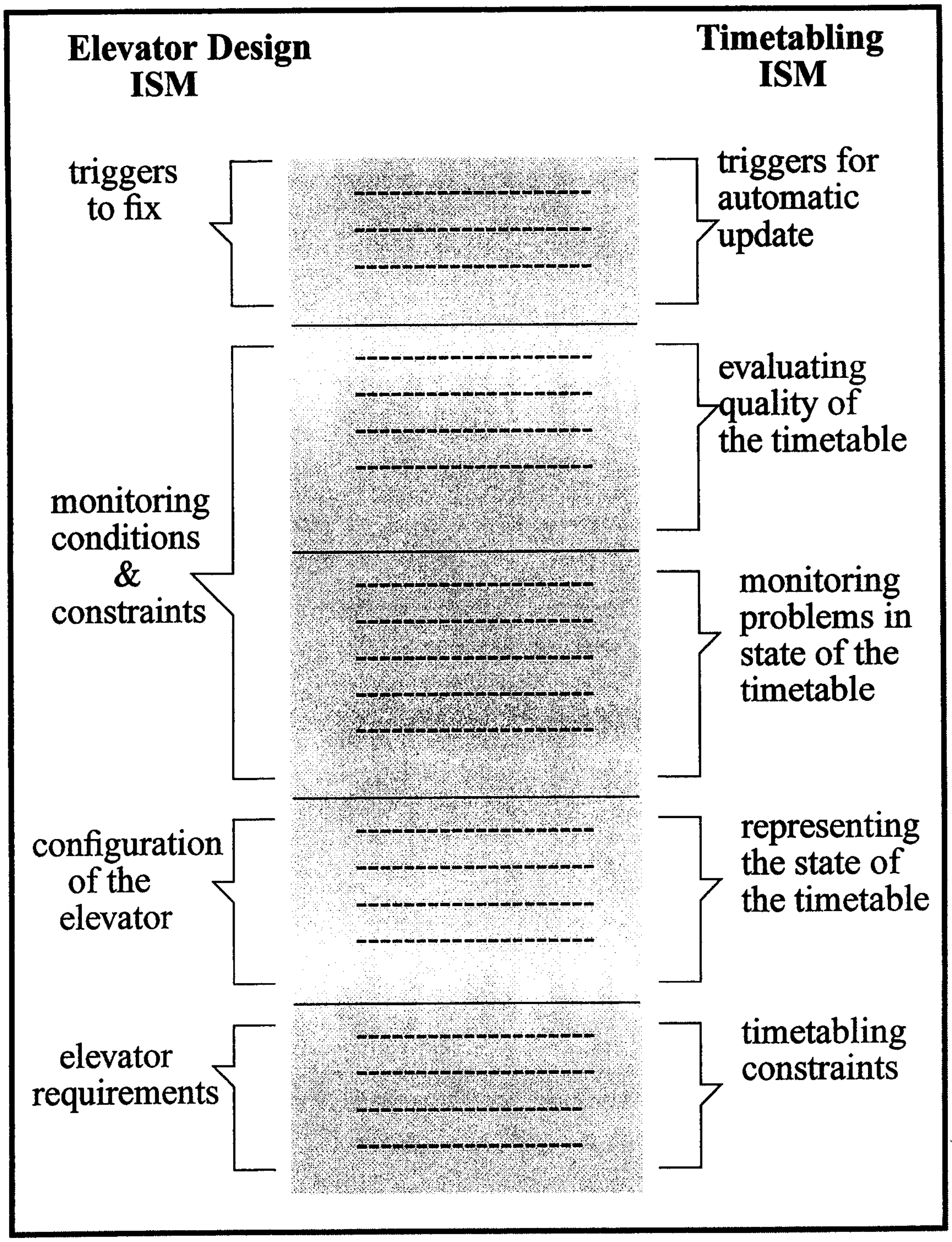

Elevator Design ISM
Timetabling ISM
triggers to fix
triggers for automatic update
evaluating quality of the timetable
monitoring conditions & constraints
monitoring problems in state of the timetable
configuration of the elevator
representing the state of the timetable
elevator requirements
timetabling constraints

FIGURE 3. Elevator Design vs Timetabling scripts

by such fixes in an experimental fashion. This is one of several ways in which the elevator design ISM can be used so as to combine automatic processing with human interaction. This interaction technique does not of course preclude the possibility of fully automatic execution.

Despite the structural similarity between the timetabling and the elevator design ISMs, there is a highly significant difference between their semantics. This stems directly from the nature of the relationship between the ISM and its referent. The elevator design ISM is the result of a thorough knowledge elicitation process, as a result of which the correspondence between the ISM and its real-world referent can be deemed to be no longer an interesting focus for experiment. Specifically, the elevator design problem has been precisely identified, the relevant observables that determine whether a solution is of acceptable quality have been established through interview and experiment, and the dependencies that relate these observables in the script of the ISM are known to be faithful to the observation of actual elevator systems. What is more, heuristics that can in general be followed to resolve problems with any design have been discovered, and their effects can accordingly be examined through interaction with the ISM.

The crucial import of the special relation between the elevator design ISM and the timetable ISM is that the elevator design ISM can be represented in the logicist idiom. As far as the particular goal of elevator design is concerned, there is no further need to consult real-world elevators or expert elevator designers - there is an artefact in which the observables are numerically and logically specified with which our interactions and observations mirror their real-world counterparts reliably and precisely. This establishes the separation of 'form' and 'content', fundamental to logic, to which Cantwell-Smith alludes in [17]. In contrast, the measures of quality that the expert timetabler applies, and the observables that are invoked in assessing this quality, are neither circumscribed nor explicitly representable by abstract data values. Only by confining these measures to a narrow set of criteria, and estimating them quantitatively, can we expect to develop a classical expert system from the timetable ISM (cf [23]).

5.0 Human-Computer Integration using an ISM

Classical expert systems were primarily directed at displacing human decision making [15]. In keeping with current aspirations for decision support systems, the use of ISMs potentially offers broader scope for decision support that allows flexible human-computer engagement. Some of the possibilities can be illustrated by the timetable ISM. Human and computer activities can be combined in the environment supplied by the timetable ISM in several ways, each exploiting a slightly different configuration of dependencies between key observables, or reflecting a different mode of interaction. These include:

Timetable comprehension and maintenance: It is typically difficult for a human timetabler to evaluate an automatically generated timetable. The timetable ISM can be regarded as essentially composed of a collection of observables relating to the available resources and their present assignment, together with a family of agents that monitor the state of the timetable and can be authorised to change the state. The source of the resource assignment data is immaterial - it is possible to extract data from any given partial or complete timetable and insert it in the ISM. In this way, it becomes possible for the human timetabler to explore the neighbourhood of given solutions to the timetabling problem in much the same way as if they had constructed it themselves.

The implementation of procedures or algorithms for timetabling: It is not usually possible to develop or to study an algorithm by interleaving automatic computation with manual step-by-step execution and revision. This can be done using an ISM. The most direct approach to automating manual interactions and heuristics for timetabling has been outlined above. As illustrated in [4], such interaction can be the basis for algorithm development, and may in due course lead to a formal specification. For an algorithmic problem as complex as timetabling, optimised execution of algorithms is essential for all but the simplest processing tasks; it is too inefficient simply to carry out the steps of manual execution with all their attendant visualisation and dependency maintenance. The potential for translation techniques to achieve such optimisation has been demonstrated in our previous work [1]. Though it is not possible for the human timetabler to intervene in the execution of such an optimised algorithm, it is possible for the results of the execution to be integrated into the manual timetabling activity. For instance, the product of such an algorithm might be a suitable observable (such as whether the completion of a partial timetable is provably impossible) or a timetable that can become a new subject for comprehension and exploration.

Opportunistic extension of the ISM to encompass a richer family of observables: One of the weaknesses of automatic activity is that it is typically specified in such a way that it is hard to adapt to a new requirement, and that it is typically executed in a manner that is oblivious to its environment. Providing support for timetabling is a task that raises many issues concerning unexpected interactions between human and automatic activities. The timescale for completion of the manual task is such that the even the hard requirements typically change during the very process of construction. In these circumstances, the role of a decision support system may be to prevent the timetabler from being thrown into such complete confusion that much of their previous work must be discarded. Though the speed at which solutions can be generated automatically may mean that little time is lost in discarding one solution and creating another, the human context for timetabling is such that at some point serious commitments, directly affecting the lives many people, need to be made. In any event, the likelihood is that some revisions will have to be made, and that typically some of these arise during the period when the timetable is in operation. As frequently illustrated in the above discussion, one of the most interesting aspects of a timetabling task that is sufficiently modest in scale for human involvement, is that - in seeking a high quality solution - the timetabler will always be liable to elaborate upon the requirement. What is more, this elaboration does not typically take the form of adding features that are generically useful in timetabling, but has a specific and topical character. For instance, it may be that - in the interests of fairness - it is important to monitor how the schedules for students in different streams are distributed over the week.

Each of these scenarios demands a form of close integration of human and automatic activity. In principle, the EM paradigm is well-suited to the task, but the technical challenges are difficult to meet with our current tools. Introducing new scripts is an exceptionally versatile technique for the dynamic extension and revision of the internal computer model. The development of interfaces to reflect such an open functionality is a more perplexing issue, since these cannot be preconceived, and are subject to dynamic extension and reconfiguration.

Integration of the timetable with other applications: More often than not, automated procedures are conceived with specific functionality and objectives in mind, so that it can be hard to integrate one procedure with another. Current developments in relational database and spreadsheet technology have demonstrated how principles related to those proposed in this paper can help to address this problem. For instance, in an integrated business package, it is a routine matter to generate many different kinds of report from a single data source. This illustrates that establishing dependencies between one data representation and another is a powerful mechanism for integrating different software packages, but its full exploitation is inhibited by the conceptual emphases of the dominant programming and communication paradigms. For instance, object-oriented methods focus on local integrity of state and behaviour, whilst propagation of state is passive, and relies upon an explicit request from a user or program. Such problems do not arise in a well-conceived observation-oriented environment. For instance, in such an environment the timetable ISM can be readily integrated with a simulation of the operating timetable that is programmed to synchronise with the enactment of the actual timetable, maintains dependencies to record which marksheets are pending and which have been returned, and sends automatic reminders to absent-minded staff. Moreover, this effect can be achieved without having to pre-specify how any particular aspect of this functionality is to be developed.

6.0 Conclusion

This paper has described how the introduction of a new paradigm for computer-based modelling can lead to a closer integration of manual and automated activities. Though the illustrations that have been discussed are simple in nature, the potential implications for decision support are broad and wide-ranging. More evidence for this claim can be found elsewhere, notably in distributed ISMs that have been constructed to simulate a historic railway accident [2,19], to implement a virtual electrical laboratory and to provide support for restaurant management [13].

The emphasis in this paper has been on relating the development of ISMs to those decision support frameworks in which the "expert system" is the archetype. This gives the greatest prominence to the distinction between our use of Empirical Modelling and logicist approaches to knowledge representation and problem solving [2].

Though it is beyond the scope of this paper to develop the theme, it is also quite natural to view the construction of ISMs as generalising another more familiar paradigm for decision support that lies outside the logicist tradition, viz. the use of spreadsheets. A companion paper to the present one on this topic is in preparation. This generalisation involves dependency maintenance that embraces much wider and more subtle aspects of state, encompassing issues such as visualisation, geometry, screen layout and interface design. It also gives much richer expression to agency, concurrent interaction and distributed viewpoints. The implications of this generalisation include:

* promoting the use of the computer as an instrument and physical device;
* enhancing the quality of the interaction and cognitive support afforded;
* establishing a new framework for collaboration in relation to tasks such as administration, concurrent engineering and system development.

Our current research is directed at exploiting these qualities in the challenging areas of requirements engineering [19] and strategic decision support [13]. A major factor in this research is to develop improved tool support that can make the use of Empirical Modelling principles more accessible to non-technical decision makers.

Acknowledgements

We are indebted to Paul Ness, Mike Joy and Simon Yung for developing the elevator design ISM, and to other members of the Empirical Modelling group for useful discussions on themes relating to this paper.

7.0 References

[1] J. Allderidge, W.M. Beynon, R.I. Cartwright and Y.P.Yung, Enabling Technologies for Empirical Modelling in Graphics, Proc. Eurographics UK, 16th Annual Conference, pp. 199-213, 1998

[2] W.M. Beynon, Empirical Modelling and the Foundation of Artificial Intelligence, Proceedings of CMAA'98, Springer Lecture Notes in AI, 1562, Springer-Verlag, pp. 322-364, 1999

[3] W.M. Beynon, R.I. Cartwright, P.H. Sun, A. Ward, Interactive Situation Models for Information Systems Development, Proceedings of SCI'99 and ISAS'99, Orlando, USA, 1999

[4] W.M. Beynon, J. Rungrattanaubol, J. Sinclair., Formal Specification from an Observation-oriented Perspective, (submitted) 1999

[5] W.M. Beynon and A. Ward, S. Maad, A. Wong, S. Rasmequan, S. Russ, The Temposcope: a Computer Instrument for the Idealist Timetabler, Research Report 372, Department of Computer Science, University of Warwick, UK, 2000.

[6] P. Checkland, Soft System Methodology, IOS Human Systems Management Vol.8 (1989) pp. 273 -289.

[7] W. James, Essays in Radical Empiricism, Longmans, Green & Co, 1912

[8] M.R. Klein, L.B. Methlie, Knowledge-based Decision Support Systems, John Wiley & Sons, 1995

[9] R. Kowalski, Logic for Problem Solving, North Holland, 1979

[10] S. Marcus. J. Stout, J. McDermott, VT: An expert elevator designer that uses knowledge-directed backtracking, AI Magazine, 9, pp.95-112.

[11] H. Mintzberg, D. Raisinghani, A. Theoret The Structure of "Unstructured" Decision Processes, Administrative Science Quarterly Vol. 21 (June 1976) pp. 246-275

[12] M. Pidd, Tools for Thinking, John Wiley & Sons, 1996

[13] S. Rasmequan, C. Roe, S. Russ, Strategic Decision Support Systems: An Experience-based Approach, Proceedings of the 18th AISTED Conference on Applied Informatics, Innsbruck, 2000

[14]T.E. Rothenfluh, J.H. Genarri, H. Eriksson, A.R. Puerta, S.W. Tu, M.A. Musen

Reusable Ontologies, Knowledge-Acquisition Tools, and Performance Systems:

PROTEGE-II Solutions to Sisyphus-2, International Journal of Man-Machine

Studies, 44, pp.303-332, 1996

[15] V.L. Sauter, Decision Support Systems: An Applied Managerial Approach, John Wiley& Sons, 1997

[16] H.A. Simon, The New Science of Management Decision, Harper & Row, 1960

[17] B.C. Smith, Two Lessons of Logic, Computational Intelligence (3) (1987) pp. 214-218

[18] L. A. Suchman, Plans and Situated Actions, Cambridge University Press, 1987

[19] P-H. Sun and W.M.Beynon, Empirical Modelling: a New Approach to Understanding Requirements, Proc. 11th International Conference on Software Engineering and its Applications, Vol.3, Paris, December 1998

[20] E. Turban, J. Aronson, Decision Support Systems and Intelligent Systems, Prentice Hall, 1998

[21] G. R. Yost, Configuring Elevator Systems (Technical Report) Marlboro, MA: Digital Equipment Corporation, 1992

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

[23] http://schedulexpert.com