The Temposcope: a Computer Instrument for the Idealist Timetabler

Meurig Beynon, Ashley Ward, Soha Maad,
Allan Wong, Suwanna Rasmequan, and Steve Russ

Department of Computer Science, University of Warwick, Coventry, UK CV4 7AL
{wmb, ashley, soha, allan, suwanna, sbr}@dcs.warwick.ac.uk
http://www.dcs.warwick.ac.uk/modelling/

Abstract. Integrating human and computer activity in semi-automated timetabling is a difficult problem. This paper attributes this difficulty to the choice of computational paradigm, and introduces an alternative framework for computer-based modelling in which to create an instrument for timetabling. The potential use of such an instrument (the 'Temposcope') is illustrated in connection with work-in-progress on a modest practical case study in timetabling (the scheduling of project oral presentations). The adoption of this framework is associated with a radical shift in perspective on the user-computer relationship, from realist to idealist users, and from the computer as an abstract machine to the computer as a physical device. This shift suggests new modes of human-computer co-operation with potentially broad implications for computer-supported timetabling.

1 Introduction

Timetabling has long been a target for automation. Finding an optimal solution to many variants of the timetabling problem has been shown to be an NP-hard problem [4]. This has stimulated much interest in heuristic methods aimed at sub-optimal solution. These include approaches based on graph colouring, simulated annealing, genetic algorithms and constraint satisfaction [8].

As technology has advanced, the role of the computer in recording, managing and presenting large and complex data sets has become as significant as its capacity for computation. This has motivated a more holistic view of how the computer can support timetabling activity. Timetabling can not only be regarded as an abstract algorithmic problem, but as integrating data capture, data modelling, data matching, report generation, and the storage of timetabling results (see e.g. Optime[1] and Schedule Expert[2]). This broader framework potentially offers more scope for the integration of manual and automatic timetabling activity. The major issue to be addressed is that the advantages of automation are typically gained at the expense of the qualities that human intelligence brings to manual timetabling and management of the timetabling process.

[1] http://www.asap.cs.nott.ac.uk/optime/
[2] http://schedulexpert.com

The objective of this paper is to outline and illustrate principles that can assist more intimate human-computer co-operation. In particular, these principles make it possible to create a single computer-based instrument that can simultaneously serve two complementary functions: supporting the cognitive model of the human timetabler working without computer assistance for matching, and providing state representations for automatic and semi-automatic timetabling activity. We review work in progress on constructing such an instrument (the 'Temposcope') in connection with an actual timetabling task, and indicate the potential implications for novel manual, semi-automatic and automatic approaches.

2 Perspectives on Computer-Aided Timetabling

2.1 Humans vs Computers

The contributions that humans and computers can make to the timetabling task are very different in character. In order to take account of both its "hard" formal and "soft" qualitative aspects, a co-operative approach seems appropriate. In this connection, one of the most challenging issues is to give computer support to the mental engagement that is involved in expert manual timetabling. This can be best appreciated by examining timetabling from a human-centred perspective.

In the manual timetabling process, human judgements are influenced by imprecisely specified factors that express themselves implicitly through exploratory activity. The manual timetabler explores different possible organisations of resources and makes qualitative judgements about them. This may involve an appreciation of issues of logistics, such as: the use and disposition of rooms; personal knowledge of people's scheduling preferences; aesthetic judgements about the quality of a solution. The judgements made stem from knowledge that eludes specification for various reasons: because it is tacit, too costly to articulate or because it emerges during or after the construction of the timetable.

The manual timetabler may also develop *ad hoc* representations of data that are situation-specific. In manual timetable construction, a mental grasp of the timetabling activity is gained through the construction of physical artefacts that serve a cognitive role: pieces of paper, a board representing the timetable, lists of events to be scheduled, etc. that are manipulated as the construction proceeds. The way in which the manual timetabler interacts with these artefacts is far less closely circumscribed than interaction with a conventional computer program. For instance, the timetabler may decide to staple pieces of paper together, to colour them or to organise them in piles in ways that are suggested by the situation rather than preconceived. Such opportunistic extension of functionality is obstructed by conventionally constructed computer software, especially where optimised algorithmic processes have been implemented.

The products of this manual activity resemble the results of scientific experiment prior to the identification of an appropriate theory. They are loosely connected, but typically elude comprehensive coherent organisation. Having access to automatic timetabling software, or to integrated suites for data input and report generation can to some degree complement open-ended exploration, but it is no substitute for it. Even an efficient and powerful timetabling algorithm cannot emulate the human awareness of a situation that experiment affords. The typical end result is that the human timetabler has to forego any possibility of close cognitive integration of manual and automated timetabling activity.

If it is to be possible to exploit the advantages of automation and yet retain the qualities that human interaction brings, it is essential to integrate the manual and the automatic steps in timetable construction. Whatever activity is delegated to the computer must also be comprehensible to the timetabler. The timetabler must be able to invoke, monitor, suspend and intervene in computational processes.

Current semi-automated systems rarely support this high degree of integration. This paper argues that the difficulty of integrating human and computer processing at a cognitive level stems from the traditional computational paradigms used in current practice. The significant difference between the cognitive models of a user and the builder of a program is well recognised [3]. It has motivated software development methodologies for user-interface design, for instance (cf [7]). It is also apparent from the difficulties encountered in integrating object-oriented analysis and design [6]. We attribute this problem to the fact that existing approaches to computer programming in all their varieties (declarative, procedural, object-oriented, genetic) rely essentially upon circumscription of the application prior to automation. This paper aims to show how to give automated support to applications with situated[3] character such as timetabling without circumscribing the domain. This is the aspiration of the 'Empirical Modelling' (EM) approach introduced in this paper — to allow a modeller to develop computer artefacts whose interpretation and evolution remains open-ended.

2.2 Realists vs Idealists

Some aspects of the semi-automatic timetabling activity can be described by an interactive algorithm in which the nature of the human input is predetermined. Others encompass activity where the interactions are informed by human intelligence about the world in ways that are not preconceived. The profound difference between these two varieties of semi-automated activity reflects two views of the human timetabler, as *realist* or as *idealist*.

The realist timetabler is a pragmatist. For the realist, the purpose of the computer is to save work. The characteristic sentiments of the realist are that time is precious, that this timetable is good enough, that people can adapt to the limitations of the automatically generated timetable. The computer's role in the realist's view is not to make the timetabling task more satisfying, but less frustrating and arduous.

The idealist timetabler is a perfectionist. For the idealist, there is no completely satisfactory timetable — it can always be improved. Even as the timetable

[3] The term 'situated' is used here to activities that are intimately bound up with their surrounding human environment, and can be directly affected by this environment in unpredictable ways.

is being constructed, the idealist sees ways in which it can be improved, and indeed sees beyond into ways in which the ways in which the timetable is being constructed can also be improved. These possible improvements cannot be preconceived because they emerge during the timetabling activity, and depend upon the special idiosyncrasies of the particular timetable being constructed. The computer's role in this context is less clear, but the idealist would like to dictate the terms, directing and understanding the computer activity and deriving satisfaction from constructing still better and more ambitious timetables.

The realist timetabler is content to rely on an algorithmic approach. This may be a batch algorithm in which knowledge of the constraints is pre-coded. Alternatively, it may be an interactive algorithm in which the observation of the external situation is to be interpreted in a preconceived manner and follows a specified pattern (cf. Fig. 1). The idealist rejects full automation and adopts a semi-automated approach. As in manual timetabling, the requirements for the timetable are subject to change even as the timetable is being constructed, and are affected by observations of the current state of the timetable and the external situation that involve no preconceptions about their nature or interpretation (cf. Fig. 1).

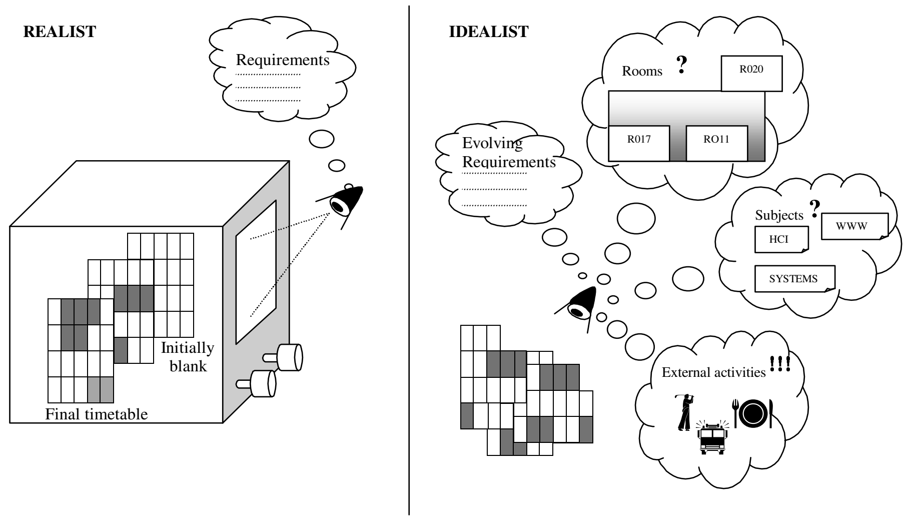


Fig. 1. Realist and idealist perspectives on timetabling

A useful analogy may be drawn between the perspectives of the realist and the idealist and two modes of exploration of a region that are aimed at finding the most congenial view. The realist, armed with helicopter and altimeter, is content to find the top of a high mountain, in the knowledge that this will generally provide a reasonable vantage point. The idealist recognises that the best view cannot necessarily be assessed by measurement alone, nor conveniently found by using powerful technology. Whatever means the idealist adopts in their search, the activity involved is open-ended and is at all times shaped and guided by interaction with the environment.

The research in this paper is in the first instance motivated by the outlook of the idealist human timetabler. Such a timetabler is at all times concerned with the relationship between the timetable and its intended role in the external context. Their evaluation of the quality of the timetable is more critical and potentially more subjective than can be represented by imposing a set of formal constraints or quality metrics upon an algorithmic solution. For this approach to be appropriate, the timetabling problem must be of an appropriate scale. Too small a timetabling problem makes it possible to separate the generation of timetables from their evaluation (e.g. to generate all plausible timetables satisfying hard and quantifiable soft constraints, and allow the timetabler to select the most suitable). Too large a timetabling problem limits the scope for situated evaluation during the construction, unless perhaps this is managed by calling upon feedback from all the potential users, as is typically done when constructing a university lecture timetable. In practice, there is no option but to be a realist if the scale of the timetabling problem is too great.

There are practical merits in the idealist perspective. Through a better understanding of the timetable generated, it may be possible to adapt the timetable more effectively, and such adaptation is generally necessary. In this way, some of the investment in analysis and evaluation can be repaid. The idealist approach may also suggest ways (possibly situation-specific) to improve on algorithmic methods. In general, the idealist is less concerned with improving efficiency than with delivering quality, and with related issues of evaluation, comprehension and maintenance. In supporting the idealist stance, our particular focus of interest in this paper is on achieving a more effective and seamless integration of human and automatic processing in timetabling.

2.3 Computer Support for the Idealist

The classical theory of computation promotes a narrow view of the role that a computer-based model can serve. Specifically, such a model is typically identified with a computer program having a preconceived function that is realised by a formally specified input-output behaviour. The automation of preconceived functions is precisely what the realist expects of the computer: the computer performs a labour-saving task to the user's specification, as far as possible reducing the need for effort and creative interpretation in use. Computer-based models that are interpreted solely in this closed and prefabricated fashion are peripheral to the idealist's agenda. The idealist is essentially concerned with observation and exploration that is either not amenable to such automation, or has yet to be automated in this fashion. In particular, should the idealist conceive an interest in some novel aspect of the timetabling task and then proceed to devise an automatic process to address this aspect, their focus of interest will shift to new concerns. In effect, the imagination of the idealist can inform the designs of the realist in specific aspects, but this will not exhaust the issues that can stimulate the idealist's imagination.

A useful analogy may be made with the way in which a design concept can evolve. A watch that was explicitly designed merely to show the time may have a button-operated built-in light. An opportunistic user might conceive the idea of operating the light button in order to attract a moth in a darkened room. A designer might subsequently devise a watch for entomologists that was designed both to tell the time and attract insects. Such a device could be conveniently used to perform more elaborate functions, such as monitoring the frequency of visits by a particular insect, estimating the time taken to locate the light source, or photographing insects. The idealist perspective is essentially concerned with those uses of a device that are not routine or automated; the realist perspective is concerned with the extent to which it is practical and appropriate to encapsulate such uses in a new device.

The conceptual framework of the idealist is distinguished from that of the realist by a focus on *situation* rather than *process*. A situation is particular to immediate current experience, whilst a process is concerned with sequences of situations that reliably follow a particular abstract pattern. In the design of the entomologist's watch, for instance, the idealist's perspective is associated with the opportunistic use of a standard watch in a particular *situation*, and the realist's perspective with identifying a *process* by which insects might be reliably attracted to an illuminated watch. Though both the idealist and realist perspectives are combined in effective design (for instance, in the systematic investigation of insect responses to a variety of situations that leads to the identification of a process for attracting insects), the situated activity of the idealist does not necessarily precede design.

The computer-based *interactive situation model* (ISM) to be described in this paper, unlike a computer program that is conceived with specific processes in mind, is intended to represent situations. A detailed discussion of the principles used in constructing an ISM is postponed until Sect. 4, but some appreciation of the distinctive nature of such a model is important both in interpreting the account of timetabling activities which follows, and in relating this to alternative approaches. Unlike a conventional program, an ISM is constructed and interpreted in a way that is directly shaped by particular situations rather than only being constrained by abstract input-output processes and context-independent computational semantics.

A precedent for computer-based models with such semantics is to be found in the spreadsheet. Plausible ways in which spreadsheets could be exploited in timetabling convey some of the essential principles discussed in this paper, albeit rather crudely. For instance, it is possible to use a spreadsheet to express how the violation of a constraint depends upon the configuration of a timetable, and to explore this dependency. Such a spreadsheet can be used in an interactive and opportunistic manner to investigate particular situations and constraints without conceiving a systematic process to address all constraints. The blend of human interaction and automated activity that is supported by the spreadsheet is distinctive. Possible activity includes open-ended experiment, both in the external world and within the spreadsheet, that can shape the timetabler's understanding of the timetabling problem and of the interpretation of the values in the spreadsheet itself. With appropriate dependencies in place, the spreadsheet can be used to assist timetable comprehension through experimental interaction whilst at one and the same time allowing processes and behaviours, such as are associated with timetabling algorithms, to be conveniently prototyped. Such use of the spreadsheet as a timetabling application is restricted by the computational overheads involved in realising algorithms through automating human interaction with the spreadsheet. In contrast, efficient algorithms and conventional programs to implement constraint satisfaction would require customised data representations and optimisations that relied upon restricting user interaction.

In broad terms, the EM approach to computer-based modelling can be viewed as generalising principles illustrated in the spreadsheet so as to create computer-based artefacts whose semantics resembles that of the physical models that an engineer or an experimental scientist constructs. The analysis that informs the creation of an ISM relies on identifying observables (cf. the cells in a spreadsheet), dependencies (cf. the definitions that relate spreadsheet cells) and agency (cf. the actions that are involved in introducing and redefining cells that are associated with spreadsheet development and use). In practical use of the many ISMs that we have developed at Warwick over several years, the readiness with which ISMs can be adapted to serve different roles is their most striking feature. Empirical evidence suggests that this adaptability is a fundamental rather than an accidental characteristic of our approach. It seems that analysing observation, agency and dependency has a central role in the way in which we construe situations, and accordingly in shaping our expectations of what sort of interactions and state-changes are plausible in any given situation. The aspiration of the EM approach, realised persuasively — if only partially — with our current modelling tools, is to create computer-based models that imitate observation, agency and dependency in their referents through integrating the construction of a model with the analysis of its referent.

The mode of construction of an ISM makes it radically different from a timetabling application that is implemented by a conventional computer program. In developing such a timetabling application, there should ideally be interaction between two people with quite distinct perspectives: one who understands the states and processes in the underlying computer program, and another who is familiar with the characteristics of the timetabling problem and the constraints and heuristics governing its solution. Since the environment for the timetabling task is subject to evolve due to changes in surrounding processes, resources, and adaptation to the introduction of automatic scheduling itself, it would be helpful if the expert timetabler could either make appropriate changes to the underlying program or at least understand changes made by the programmer. This is a difficult process, since it involves implicit interaction between the two very different mental models of the timetabler and the programmer. The virtue of the framework of observables, agency and dependency that is used to construct the ISM is that it offers explicit handles for comprehension by both participants. Firstly, the form of the code of the model itself directly embodies significant information about its semantics (cf. the way in which the definitions of spreadsheet cells discloses information about the external observables they represent). Secondly, the model is amenable to experiment in an open-ended fashion that allows hypotheses about relationships amongst observables to be framed and tested (cf. the insight into the significance of spreadsheet values that can be gained through experiment). On this account, the ISM may serve as a most appropriate vehicle through which the timetabler and the programmer can communicate.

3 Developing a Timetabling Instrument

3.1 The Timetabling Application

The potential for applying EM techniques and tools to timetabling will be briefly outlined with reference to work in progress on a real application. This application involves timetabling a week of oral presentations for final-year project students. Each presentation requires a timeslot of 30 minutes duration between 9 am and 5 pm from Monday to Friday. The presentation is attended by the project supervisor, the second assessor, and a moderator (cf. Fig. 2). 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 increasing year by year.

The administrative task of managing the presentation timetable is currently carried out with modest computer support, having several aspects: the collection of data on availability of staff and students, the assignment of second assessors and moderators to each project, and the preparation and publication of the timetable. Much of the data is communicated and recorded electronically, but, currently, the construction of the timetable is largely done manually. As an additional complication, there are often unexpected changes to the availability of staff and students, both throughout the period during which the timetable is under construction, and during the week of presentations itself. Exceptional projects that have to be scheduled outside the normal rooms also occur, for example, because of non-standard equipment needs.

The size and nature of the timetabling problem is such that automation is desirable. By the time all the relevant data has been collected, the time available for constructing the timetable is quite short. A simple conventional program to process the relevant data and construct a timetable is already available. This can generate solutions, but is quite unsuitable for use when any subsequent modifications to the timetable have to be made, since any change to the data results in a complete reconstruction of the timetable. Previous experience of

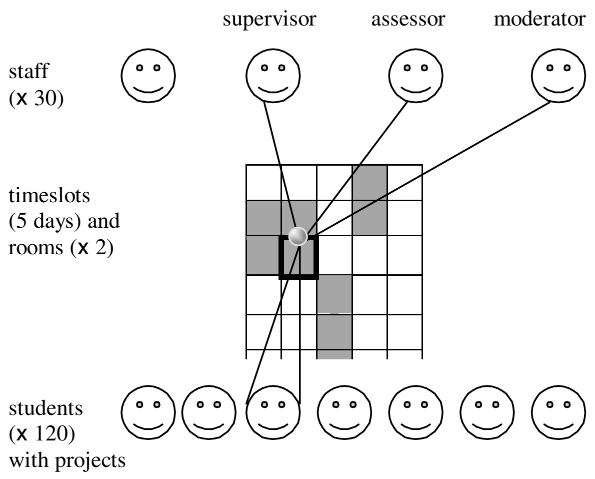


Fig. 2. The case-study context

manual construction of the timetable also reveals a number of qualitative and logistic issues, important in developing a satisfactory timetable, that are difficult to take into account when automating the task. For instance, it is useful to avoid fragmenting commitments of staff time, to eliminate delays introduced by staff having to switch between venues, and to organise presentations thematically. Many of these issues could be more easily considered in the past, when student numbers were smaller, and manual timetabling was more appropriate.

3.2 The Temposcope

The product of our current work in progress is a prototype "timetabling instrument", named the Temposcope[4], designed to support the preparation of the presentation schedule. The construction of the Temposcope has similarity with the development of a new scientific instrument, such as the microscope [5]. There are three principal roles in this construction: confirming that the instrument is operating as intended (the *designer* role), ensuring that the instrument is properly situated in an appropriate environment (the *technician* role), and becoming familiar with the most effective ways to use the instrument (the *user* role). During the earliest stages of the development, it can be difficult to distinguish the problems relating to these three roles. Once this distinction has become clear, there is scope for refining the instrument. This involves making it easier to set up, simplifying its internal mechanisms, and providing more effective modes of interaction.

This section informally describes the steps that have been taken so far towards building our Temposcope, and indicates the potential for future use. The

[4] an instrument for *T*imetabling with *E*mpirical *M*odelling for *P*roject *O*rals.

terms in *italics* have a special meaning to be elaborated in later sections of the paper.

3.3 Setting up the Temposcope

As the analogy with a scientific instrument suggests, the purpose of the Temposcope is to make it easier for the timetabler to access and manipulate *observables* of interest. For instance, in constructing the timetable, the timetabler will be concerned about such observables as 'whether a staff member is available in a certain timeslot', or 'whether a staff member has been double-booked'. The timetabler may decide that it is necessary to change the moderator for a session, and will be aware that this has potential implications for availability and workload, but will need to check the status of the appropriate observables. The Temposcope is intended to make it easier to identify the relevant observables and determine their current status. In fact, the state of the instrument itself should reflect the status of such observables accurately at all times, and it should be more convenient for the timetabler to consult the state of the instrument rather than the state of the real world.

Figure 3 outlines the structure of the Temposcope. The timetable data at the centre of the figure comprises information about the availability of staff and students, an assignment of examiners to each student (`SAM`), and an assignment of students to slots (`TT`). This data — which might be obsolete, inconsistent or partially complete — determines the state of the timetable with which the timetabler is currently concerned. Many significant observables associated with the timetable data can be derived using *functional dependencies*. These observables relate to issues such as the completeness and feasibility of the schedule for each member of staff and student, the suitability of the examiners, the constraints imposed by rooms, and the relative workloads of staff.

For the realist timetabler, a timetabling instrument is a tool that, once developed to an appropriate level of sophistication, can deal with all the observables of interest. As far as the realist is concerned, how the instrument is constructed is immaterial. For the idealist, in contrast, it is important that the timetabling instrument can be refined to take account of newly conceived observables, and that it is constructed so that such refinement is possible even during use. It would be desirable (but to our best present knowledge surprising!) if the idealist timetabler were able to carry out such a refinement unaided, but ensuring that correction and refinement do not interfere with its use is a more significant concern. In the Temposcope, this is achieved by using principles similar to those that enable the user of a spreadsheet to revise the tax rate without suspending a complicated financial analysis. Indeed, our timetabling instrument is almost entirely constructed by formulating functional dependencies.

Figure 4 is a screenshot that shows the Temposcope in operation. The instrument was assembled from several simple mechanisms, each of which relates to a different aspect of the timetabling task. One component, for instance, is concerned with calculating current availability from the declared availability and assignments for each staff member, and is declared by a functional dependency.

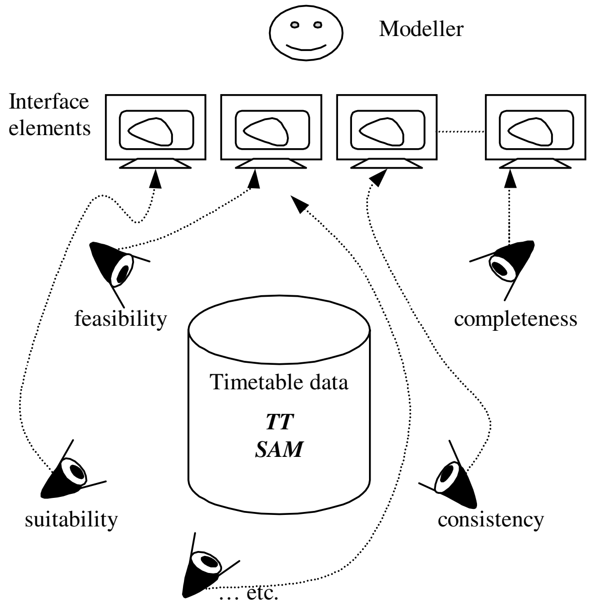
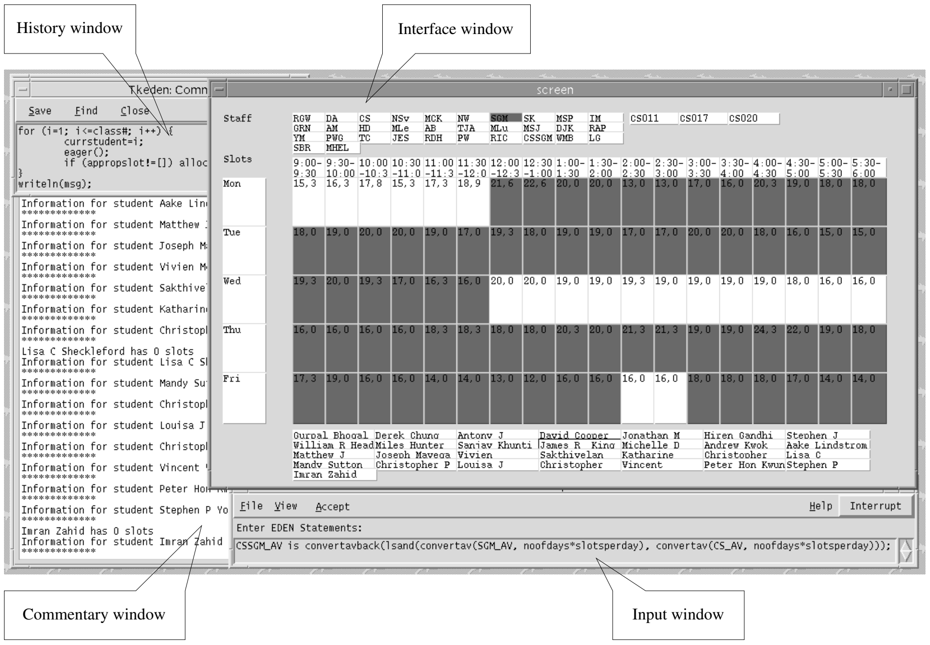


Fig. 3. Outline structure of the Temposcope

This mechanism was developed before the complete instrument was assembled, and tested with simple abstract data. In the Temposcope itself, the mechanism operates on data of a more subtle origin: the staff schedule is functionally determined by the matching of students to slots and the allocation of examiners, whilst the timeslots have to take account of room information. The first stage in assembling the complete timetabling instrument involved checking the dependencies and interactions between such sub-mechanisms.

For this purpose, our instrument was first applied to data extracted from an old timetable. This comprised a list of records detailing the names of 29 (a subset of the 120) students, together with their project titles, examiners, and scheduled slots. The declared availability of staff was recovered from old documentation. In bringing the timetabling instrument into an appropriate state, several semantic problems in the legacy data were identified. These highlighted invalid date information for three students whose projects had been rescheduled, and several discrepancies between the declared availability of staff and the actual schedule. Unlike a timetabling machine that might typically refuse to process such inconsistent input, our timetabling instrument is intended only to disclose the status of observables, whether or not they are consistent. The timetabler is alerted to such problematic issues through the commentary interface in Fig. 4, where some students have no allocated slot, and some staff are declared unavailable. A more interesting anomaly in the data was created by a student who was being jointly supervised. For this student, the pseudonym `CSSGM` was introduced



```
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 Sh
*************
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 Hon
*************
Information for student Stephen P Yo
*************
Imran Zahid has 0 slots
Information for student Imran Zahid
*************
```

Staff

RGW	DA	CS	NSv	MCK	NW	SGM	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	21,6	22,6	20,0	20,0	13,0	13,0	17,0	16,0	20,3	19,0	18,0	18,0
Tue	18,0	19,0	20,0	20,0	19,0	17,0	19,3	18,0	19,0	19,0	17,0	17,0	20,0	20,0	18,0	16,0	15,0	15,0
Wed	19,3	20,0	19,3	17,0	16,3	16,0	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	16,0	16,0	16,0	16,0	18,3	18,3	18,0	18,0	20,3	20,0	21,3	21,3	19,0	19,0	24,3	22,0	19,0	18,0
Fri	17,3	19,0	16,0	16,0	14,0	14,0	13,0	12,0	16,0	16,0	16,0	16,0	18,0	18,0	18,0	17,0	14,0	14,0

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(lsand(convertav(SGM_AV, noofdays*slotsperday), convertav(CS_AV, noofdays*slotsperday)));
```

Fig. 4. Screenshot of the Temposcope

for the joint supervisors CS and SGM. This had further implications, which are discussed below.

Because the data in the old timetable was well-understood and documented, it provided a good environment for testing and improving the timetabling instrument. This involved performing many experimental interactions, and seeing how far expectations were confirmed by the changes of state of the instrument. Example interactions used to check the consistency of the instrument included changing the availability of staff members and students; moving a student to another slot, and changing the examiners. Such experimentation revealed a significant omission in checking the consistency of the timetable data: there was no explicit indication that a member of staff had been double-booked. Another dependency was introduced to make this inconsistency accessible. The old timetable data also provided a means to classify projects and add information about the suitability of their examiners. This was done by dividing projects into two classes and setting up dependency relations to identify the supervisors or assessors of a project in each class as suitable for examining all the projects in that class. It is important to note that all the changes to the state of the instrument described here, whether they concern experimental interaction or correcting and improving the instrument, were achieved by attaching new values or dependencies to existing observables, or by formulating new dependencies.

The above discussion deals with the basic set-up and use of the timetabling instrument. It is a rational reconstruction that does not reflect the ambiguities that arose whilst it was unclear whether the source of an inconsistency was the data (technician's role), a fault in the instrument (designer's role), or a user error in observation or interaction (user's role). More significant use of the instrument could only begin after confidence in making such judgements had been gained.

3.4 Manual Timetabling with the Temposcope

The first exercise undertaken was manual re-timetabling based on the old timetable data. For this purpose, the obsolete timetable, currently defined by a list of (`student_name`, `slot`) pairs, was replaced by the empty list, and all other timetable data left unchanged. To find a suitable slot for a particular student involves finding their examiners and determining when they are all available, taking account of declared availability and any previous scheduling assignments. The dependencies introduced for this purpose are as follows:

```
curr_student = ...
Supervisor, Assessor, Moderator is[5] f_of(curr_student);
common_AVAIL_of_S+A+M is f_of(AVAILs_of(SAM), ASSGNs_of(SAM));
```

where `AVAIL` refers to slots where availability is declared, and `ASSGN` to slots to which staff are assigned.

Since the actual scheduling of examiners is dependent upon when the examinee is scheduled, there is a further dependency to be recorded, viz.

```
ASSGN_of_supervisor[curr_student] is f_of(curr_student, TT, SAM);
```

With these dependencies in place, a simple way to complete the table manually is to move the focus of the `curr_student` observable from student to student, making and revising assignments as appropriate. As an improvement on this blind search, a simple heuristic can be used. A single dependency introduces the number of possible slots currently available to each student as a new observable. The assignment of slots can then proceed automatically, for example, by following the normal practice of scheduling the most tightly constrained students first.

Whilst exploring activities of this kind, a hidden fault in the instrument emerged. The definition of `ASSGN_of_supervisor` above had been formulated so that it relied on consistent indexing between the two basic records in the timetable data — a consistency that was guaranteed when both records were extracted from legacy data, but which could no longer be assumed. The correction of this fault involved reformulating a dependency, and validating the result by revisiting the experimental interactions carried out at the previous stage of development. (Such re-validation is only in general required when revising rather than adding dependencies.)

Significant extensions of the original requirement can be addressed even after the instrument is well developed. In order to illustrate this, the issue of

[5] `is` denotes a functional dependency, `=` an assignment

room availability was deliberately neglected at earlier stages. The richness of the observations supported by the timetabling instrument allows several different approaches. It is possible to deal with room availability using a similar approach to that used for staff and students. However, if the restriction imposed by rooms relates purely to the number of parallel sessions, it is simpler to observe the number of sessions timetabled for each slot. If, in addition, particular projects have to be scheduled in particular rooms, the availability of the associated students can be expressed as dependent upon the availability of these rooms. By using strategies of this kind, and exploiting the combination of manual and automatic activity, the timetabler can deal with such situation-specific problems of scheduling for which it would be inappropriate or impossible to develop a generic solution.

3.5 Automated Timetabling with the Temposcope

In its present state, the timetabling instrument is ready for prototype use in connection with this year's timetabling task. The introduction of automated routines is simple. The manual process of switching the focus of the `curr_student` observable, for instance, can be performed by an iterative procedure:

```
for i = 1 .. class_size {
  curr_student = i;
  if (is_slot_available & none_allocated) update TT[curr_student];
}
```

A possible result of executing this iteration is depicted in the screen shot in Fig. 4. From this display, it is apparent that all but two students are scheduled. One of these students cannot be scheduled because of an inconsistency in the original legacy data. The other is unavailable because the spurious examiner `CSSGM` introduced earlier has been given no availability (amongst other things). This issue can be addressed by introducing another dependency:

```
avail_of_CSSGM is intersection_of(avail_of_CS, avail_of_SGM);
```

The timetabler can then allocate a slot for this student either manually, or by invoking the automatic iteration for a second time. The approach to handling an anomalous feature of the timetable data illustrated here is very different in character from redesigning a machine to accept a wider range of inputs. Both the timetabler and the instrument are involved in accommodating such a feature, and the timetabler has to exercise special discretion in interacting with the instrument and interpreting its response. For instance, it is not appropriate to modify the availability of `CSSGM` through the button interface — the availabilities of `CS` and `SGM` must be modified instead.

The timetable-related activities described above address only the first stage in the development of an effective timetabling instrument. They are nonetheless representative of activities that can continue in an open-ended fashion towards the creation of more powerful instruments. The future agenda for this project will feature studies in practical use of the instrument, with particular reference to the role and potential impact it can have in the administrative processes associated with timetabling. One key issue concerns identifying the most significant observables and displaying them effectively, with reference both to observables familiar to every timetable user and to those newly introduced by the idealist timetabler. Another, for which a framework has already been established by our existing software tools, involves distributing the timetabling instrument for co-operative use across a computer network.

4 Empirical Modelling: Principles, Techniques and Tools

The work in progress outlined in Sect. 3 uses a collection of principles, techniques and tools developed at the University of Warwick under the umbrella name Empirical Modelling (EM). The research has a human-centred focus that was motivated in Sect. 2.

The key concepts of EM[6] are the observable, dependency and agent [1]. These frame a form of analysis of applications that is radically different from a functional or object-oriented approach. Limited precedents for this kind of analysis are to be found elsewhere: in the analysis of functional dependencies in relational database design, and in the dependency analysis involved in creating a spreadsheet.

The central idea of EM is the construction of an Interactive Situation Model (ISM) [2], of which the Temposcope is an example. An ISM supplies the modeller with patterns of interaction that resemble those observed when experimenting with its referent. Like a spreadsheet, an ISM has an uncircumscribed set of useful interpretable states and interactions. How these states are manifest, and what interactions are invoked, is highly dependent on the situation and motivation of the modeller. The legitimacy of an interaction is determined in a situated rather than an abstract manner, by direct consideration of the external referent. This is similar to the way in which the definitions of spreadsheet cells are framed according to expectations about how the values of observables are interrelated. The precise usage and evolution of an ISM remains open-ended, and is typically shaped by value judgements and pragmatic issues. The role of the ISM is closely associated with modelling activities of an empirical nature that — in other approaches — precede formal specification.

An ISM is experienced as an open environment rather than a closed system. Like a spreadsheet, it is encountered in a particular state and invites exploratory interactions. The particular state of an ISM is determined by a family of definitions (a *definitive script*) that resembles the network of definitions of cells that lies behind the spreadsheet interface. As in defining the cells of a spreadsheet, each definition explicitly or implicitly specifies the value of an *observable* via a *functional dependency*. A typical implicit definition takes the form `x is f(a,b,c)`, where `x`, `a`, `b` and `c` are associated with observables in the referent, `a`, `b`, and `c` are defined elsewhere in the script, and `f` is an operator that

[6] see `http://www.dcs.warwick.ac.uk/modelling/`

reflects the way in which the value of `x` is perceived to depend upon the values of `a`, `b` and `c`. Semantically, the relationship `x is f(a,b,c)` is to be interpreted as referring to expectations about the consequences of changes to observables `a`, `b` and `c`, rather than as an equational constraint. It expresses the observation that if the value of `a`, `b` or `c` is changed, then the value of `x` changes according to the dependency `x is f(a,b,c)`, and that the change to the value of `x` conceptually belongs indivisibly to the change to `a`, `b` or `c`. In effect, neither the ISM nor its referent can be observed in a state in which `a`, `b` or `c` has changed but the concomitant change in `x` has yet to occur.

The script of an ISM serves as the basis for an exceedingly rich state-transition model. Transitions from state to state are associated with the introduction of a new definition or the modification of an existing definition. Only a small proportion of the possible transitions from a state are of interest — indeed many are not interpretable, but it is difficult to preconceive which, or to constrain their interpretation. In EM, transitions are frequently ascribed to agents. This gives support for semantic distinctions that are typically ignored in abstract computational accounts. For instance, in the matching process involved in constructing a timetable, there is a distinction between commitments volunteered by a member of staff and commitments subsequently introduced by the timetabler. In the ISM, transitions are associated with agent action, as mediated either through a single definition, or through several redefinitions to be executed in parallel. Automatic agents can be modelled in the ISM by introducing actions that are triggered according to the current values of script observables. All such activity involving definition in the ISM takes place at the discretion of the modeller, who can always intervene in any state in the role of a supreme agent, with power to introduce definitions in a potentially arbitrary and unrestricted manner. In this manner, human and automated activities in the ISM are closely integrated.

The construction of the ISM illustrates a number of significant features. It allows development to proceed interactively in an incremental fashion in such a way that there is continuity and a close correlation between the observables from one version of the ISM and another. The roles that the modeller and the computer play in maintaining the semantic integrity of the model are much more balanced than in a conventional program. Recall that the phantom combination examiner `CSSGM` was given availability in Sect. 3.5 by introducing a new dependency on the availabilities of `CS` and `SGM`, but that it was not appropriate to modify this availability directly. The interface does not enforce this constraint — it is up to the modeller to interact with the ISM in a way that respects this relationship. This illustrates a more general principle: the modeller in general wants the freedom to experiment, and to make small adjustments to the ISM that are particular to the situation rather than generic in character. It is more convenient and more appropriate to achieve these aims by using the ISM with discretion in a semi-automatic mode than to re-engineer the model to accommodate exceptions.

The software tool that we use to create computer-based artefacts such as the Temposcope is a special-purpose interpreter (the EDEN interpreter) that has been developed for constructing ISMs. It may seem that some aspects of the modelling task could be addressed more directly using conventional packages, but the use of EDEN involves subtle and significant differences. For instance, dependency relationships that are associated with views and reports are dynamically maintained, and are typically mediated to the user through visualisation that is dynamically refreshed. EDEN offers far greater scope for establishing dependencies amongst internal data values of all kinds, as well as dependencies between such values and the geometric, textual and iconic elements on the screen. These dependencies can also be reconfigured on the fly, and be distributed to users on a network of workstations.

The characteristics of the EDEN environment are illustrated in the screenshot of the Temposcope ISM in Fig. 4. There are three principal elements in the screen display. These are:

1. the EDEN *input* window, in which the values and definitions attached to variables in the current script can be interrogated, and through which new definitions, functions and actions can be entered into the script to change the current state of the ISM. It is the primary interface through which the ISM is constructed.
2. an *interface* window whose contents are functionally determined by the underlying data set (and the lists of staff and student names in particular) and the current script, that provides a simple interface for visual feedback and mouse interaction. The interface window can serve many different functions according to the configuration of the script. For instance, in Fig. 4, the grid of cells can be used to display the timetable, to display and modify data about the availability of staff, or to display quantitative information relating to the particular cells. In Fig. 4, the pair of numbers currently on display indicates the number of staff who have declared themselves available in that timeslot, together with the number of staff who have been scheduled in this slot. The values of these numbers are specified by definitions, and are updated automatically.
3. a *commentary* window in which information about the current state of the ISM is displayed through textual output as interactively requested by the modeller or output by EDEN actions. The commentary window is the most convenient medium through which to get informal feedback about the state of the model. For instance, it can be used for interactions that have a topical short-term interest in connection with singularities in the state resulting from invalid input or misconceptions about the external state.

5 The EM Approach — Ideal for the Idealist?

The immediate practical objective of our current work in progress is to see to what extent our computer instrument can be used to assist human-driven timetabling for project orals. This section discusses some of the issues that arise in using the Temposcope in an approach to timetabling that emphasises manual rather than automatic scheduling. It also considers the distinctive merits and limitations of the new computational principles that are being exploited. Finally, it speculates on the broader significance and potential implications of empowering the idealist timetabler.

5.1 Assisting the Manual Timetabler

The primary motivation for developing the Temposcope is best appreciated by considering the state of mind of the idealist timetabler engaged in manual timetabling. Such a timetabler will typically have created some portion of the final timetable. Their current understanding of the timetabling problem will encompass much more than is apparent from the partially completed grid, however. They may be on the point of identifying a new goal that could enhance the quality of the timetable, such as removing an inconvenient gap in a staff member's schedule. They will be aware of where conditions are close to critical in some respect: for instance, because there is limited scope about the scheduling of a particular combination of staff members, or limited discretion over who can otherwise serve as a second assessor. They may also be aware of particular reasons why a specific matching of project to timeslot has been made. Current knowledge of a situation of this nature is difficult to record formally, whether on paper or in a computer model. Exploratory interaction with the Temposcope provides the timetabler with a means to revisit the experience of the current situation that is informing their judgement. This is useful both in confirming intuitions about what is feasible, and in communicating the reasons for decisions to others.

The distinction between use of the Temposcope and a fully automated approach is highlighted by the support it gives for monitoring manual timetabling activity where there is temporarily inconsistency. Automatic support for timetabling typically imposes hard constraints upon all scheduling activity, whether explicitly performed by the user or carried out automatically. In contrast, the Temposcope is always open to interaction in which the modeller is free to redefine observables arbitrarily — in particular to venture scheduling assignments whether or not they are physically realisable. An experienced timetabler may well tolerate anomalies of this kind in a partially completed timetable, and pend certain problems for resolution at a later stage. This is feasible and appropriate only if the timetabler is fully engaged with the task, and has a sound awareness of the rationale for the decisions that have informed the construction to the present point. As the case study in Sect. 3 illustrates, flexible use of the Temposcope in this manner does not prevent the timetabler from exploiting a more orthodox strategy: an interface can be provided in which the modeller's input is restricted to scheduling actions that do not violate constraints, or to invoking scheduling agents that respect the constraints. It is on this basis that the instrument can support close integration of manual and automatic timetabling.

Many of the practical problems of constructing a timetable manually stem from the difficulty of recovering the mental contexts encountered in the construction process. Such contexts are associated with experiential aspects of the timetabling activity rather than more mundane and abstract knowledge about the current status of constraints. For instance, if the display representations of timetabled project orals are colour coded so as to reflect their subject matter, the appearance of the screen itself may be the trigger that prompts the timetabler to a particular train of thought. As the manual timetabler moves in and around the solution space, the Temposcope not only relieves the timetabler of error-prone checking of constraints, but enables states to be conveniently re-created in ways that restore the experiential cues.

5.2 Paradigms for Computer Support

An experiential emphasis in computer use is evident in current trends. Our ISM-based approach has more in common with modelling the cognitive artefacts used for timetabling than implementing efficient automatic timetabling routines. As in all applications where computerisation of manual processes is concerned, it is not necessarily appropriate to imitate existing processes and artefacts. In general, computer-based artefacts are more versatile than physical or mechanical artefacts, and admit more flexible transformations. The ever-increasing support for construction of artefacts that computer-based technology offers is reflected in developments in software that can be used in timetabling. For instance, relational databases, spreadsheets and visual programming techniques are now routinely used in combination to store the current data about availability of resources, and to create reports and views to represent this information in the many ways that are appropriate to the timetabling task and the needs of its developers and users.

The extent to which new technologies and practices can deliver a more satisfying integration of human and automatic activity depends crucially upon our conceptual perspective on computer use. In this respect, the computational framework in which the Temposcope operates is significant. The classical theory of computation does not directly address the role of the computer as a physical device. A traditional programming paradigm is not so well suited to the idealist stance. The programming process involves a number of separate phases: requirements, specification, design, implementation. It is hard to integrate these phases, whether or not they are executed to a rigid pattern. The idealist modifies the requirements for the timetable whilst it is being constructed. This can be achieved in the manual activity because the timetabler is always able to examine a physical representation of the partial timetable that can be interpreted in relation to its real-world significance.

Some key differences between a realist and idealist perspective on computer support, as they relate to timetabling, are summarised in Table 1. The sharp distinctions drawn between the realist and idealist perspectives in this table are intended to clarify two complementary views of the practical use of computers. It is impossible to conceive computer use that is so much oriented towards the idealist that it operates beyond the limits of any administrative environment, and involves no preconceived interfaces or user goals. Similarly, all computer use requires some element of informal engagement with the experience of the user, and invokes the computer both as an abstract computational machine and as a physical device and instrument. The effective integration of human and computer activities can be seen as giving conceptual integrity to these two perspectives within a single framework.

5.3 Empowering the Idealist

The concepts behind the Temposcope indicate the potential for a radically different relationship between human and computer activities. This is associated with a shift in perspective on the nature of the user — from realist to idealist, and in the role of the computer — from abstract machine to physical device. This can be illustrated by considering broader implications for the administrative processes that surround timetabling.

The traditional practice in automation is to separate concerns as far as possible — for instance, in the context of our case study, by presuming that comprehensive and reliable knowledge of staff availability is guaranteed, and that the assignments of second assessor and moderator are fixed prior to embarking on the timetabling exercise. The administrative activity is phased to conform to a pattern that guarantees that as far as possible the human and computer roles can be performed routinely and efficiently. A realist timetabler would prefer to work within a systematic framework where simplifying administrative constraints apply. The idealist timetabler, who is first and foremost committed to delivering a timetable of the highest quality, and derives satisfaction from the task alone, has a different — and truly unrealistic — stance towards peripheral matters of expediency. When seeking the timetable that best meets many different criteria, the freedom to revise decisions and to respond opportunistically to changing circumstances is appreciated rather than deplored. In effect, the change in user profile from realist to idealist is associated with a fundamental change in attitude to the unpredictability and unreliability of experience. The concept of an idealist timetabler does indeed demand idealised human qualities and characteristics: to be striving for excellent intelligence in relation to all the pertinent issues, to be meticulous and diligent, and to be indulgent where there is uncertainty about the reliability of information supplied.

The idealisation invoked here is unlike perfect knowledge in the logical sense. For example, the idealist timetabler is not assumed to know whether a given partially completed schedule can or cannot be consistently completed. On the contrary, the underlying assumption in EM is that all knowledge is incomplete and imperfect, and liable to be revised and improved in the future. The timetabling task is not viewed as having a precisely specified perfect solution, but many possible solutions, some more perfect than others. Rather than attempting to eliminate subtleties and obscurities by constraining the real-world problem, the modeller accepts the impossibility of finding a comprehensive solution and understanding of problems, and focuses on creating artefacts that assist in grappling

Table 1. Two perspectives on computer-assisted timetabling

The Realist (Conventional) Perspective	**The Idealist (ISM-Based) Perspective**
uses abstract state and data representation.	uses personal and situated data representations.
follows a hidden algorithm.	supports timetabling activity under the complete control (and ideally within the comprehension) of the timetabler, with as much automation as desired.
offers a system for a specified intended use.	offers an open-ended model, "environment" or "instrument" that the modeller can shape in ways that are not pre-determined, and can use for *ad hoc* experimentation.
is difficult to adapt to unforeseen uses.	is highly flexible (albeit only in the hands of an experienced modeller).
is efficient and optimised.	aims at quality rather than efficiency, although some optimisation is possible through run-time analysis of the dependency tree.
is scalable to large problems.	offers limited scope for large-scale situated evaluation, unless perhaps the evaluation is performed in parallel by many people using a distributed ISM.
uses batch-style interaction and is as fully automated as possible: set up an input, run the algorithm, examine the results.	uses spreadsheet-style interaction that gives rich support for modelling functional dependencies. The level of support for procedural activity ranges from purely manual to fully automatic at the timetabler's discretion.
is easy to use via an "accessible" user interface.	uses an interface that is (ideally) built and adapted interactively to the modeller's own personal and situated requirements. (An interface which admits any possible change to the model is difficult to conceive — by definition, interfaces restrict possibilities.)
accepts only perfect, clean (or very high quality) input.	accepts "arbitrary" input and assumes that all knowledge in the model is subject to future revision.
involves computer activity that is interpreted by the user according to preconceived conventions.	involves computer activity that is freely and dynamically interpreted by the modeller according to the current situation.
treats the computer as an abstract computational device with formally defined mathematical semantics.	treats the computer as an artefact, with a semantics that is mediated interactively and experientially.

with realistic complexity and confusion. The most significant aspects of an ISM are that it is open-ended, and, in so far as it represents experiences of the referent, these are consistent with experiment and observation. In this way, it provides the appropriate environment to complement the unattainable aspirations of the idealist.

6 Conclusions

This paper has outlined work in progress on timetable construction in a novel conceptual framework. Though the specific techniques considered are surely not without precedent in approaches to timetabling, there is here a more significant agenda attached to their exposition. The shift of perspective towards analysis based on observables, dependency and agency that is commended in EM cannot be viewed as merely proposing new features to be imported into existing computational paradigms. The full justification and evidence for this claim is beyond the scope of this paper (cf [1]). In practical terms, the full benefits of constructing an ISM can only be appreciated within a computational framework in which dependency maintenance and agent interaction can be used pervasively to address all associated matters of interface design, visualisation and data management.

Though the Temposcope is still unrealistically simple, it illustrates in essence several of the key features of the timetabling instrument and its potential applications. These include the potential for semi-automated activity, for exploring timetabling heuristics that could improve the efficiency or quality of the construction process, and for then implementing new automatic timetabling strategies. It can be the nucleus for a much more elaborate ISM that addresses matters of user-interface design for a semi-automatic timetabling application, for instance. It could also be readily extended by attaching new dependencies so as to provide reporting facilities giving information on marks awarded, and profiles of staff performance in supervisory and examining roles.

One of the most challenging issues for our future work is to create an acceptable interface to the timetabling instrument. To appreciate the scale of this problem, it is helpful to compare an ISM with an engineer's prototype. It is quite typical for such a prototype to be precariously constructed, in some respects incomplete or not fully engineered and possibly not even fully understood. Interaction with the prototype cannot be easily delegated — it often involves intimate knowledge of the significance and context for each interaction. It is hardly appropriate to instruct the engineer to put a proper interface on the prototype so that it is easy for just anyone to interact with it. The subtlety and sensitivity of an engineer's interaction with the prototype is to some extent a reflection of the degree of thought associated with its conception. Perhaps one of the most important characteristics of an ISM is that, like the prototype, it does not easily sustain mindless interaction.

References

1. Beynon, W.M., 1999, "Empirical Modelling and the Foundations of Artificial Intelligence", *Proceedings of CMAA'98*, Lecture Notes in AI 1562, Springer, pp322–364, 1999.
2. Beynon W.M., Cartwright R.I., Sun P.H., Ward A., "Interactive Situation Models for Information Systems Development", *Proceedings of SCI'99 and ISAS'99*, Volume 2, pp9–16, Orlando, USA, July 1999.
3. Cooper A., "The Inmates are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity", SAMS Publishing, 1999.
4. Garey, M.R., Johnson, D.S., "Computer and Interactability — A guide to NP-completeness", W.H. Freeman and Company, San Francisco, 1979.
5. Hacking I., "Representing and Intervening: Introductory Topics in the Philosophy of Natural Science", Cambridge University Press, 1983.
6. Kaindl H., "Difficulties in the transition from OO analysis to design", IEEE Software, September / October 1999, pp94–102.
7. Roberts D., Berry D., Isensee S., Mullaly J., "Designing for the User with OVID: Bridging User Interface Design and Software Engineering", Software Engineering Series, IBM, Macmillan Technical Publishing USA, 1998.
8. Schaerf A., "A Survey of Automated Timetabling", Report CS-R9567 1995, Computer Science/Department of Software Technology, Centrum voor Wiskunde en Informatica, The Netherlands.