Cadence and the Empirical Modelling conceptual framework: a new perspective on modelling state-as-experienced

Meurig Beynon and Nicolas Pope
Computer Science, University of Warwick, CV4 7AL, UK

The aspiration in Empirical Modelling (EM) is to build artefacts for sense-making ("construals") that exhibit interactive characteristics similar to those observed in the situation to which they refer. The relation between an artefact and its referent is established through constructing a close correspondence between dependencies, observables and instances of agent action. Specifically, different kinds of agent interaction with the referent have counterparts in the model that are recognisably congruent in that they disclose similar dependencies between observables. The full elaboration of this notion lies beyond the scope of this paper - it has been a central theme of the EM project as documented at [27]. A crucial aspect of the approach is the emphasis that is placed upon the experiential nature of the correspondence between an artefact and its referent. This represents a radical departure from the conventional functional and operational manner in which a computer program is interpreted. It means that the interpretation of an artefact is open and fluid. For instance, it is subject to evolve over time (e.g. "facility in recognising dependencies can be learned"), can be dependent on the observer (e.g. "relationships can only be discerned if the observer isn't colour-blind"), and on the specific situation within which interaction and observation is being conducted (e.g. "whether changes to observables can be identified may depend on the level of lighting").

To date, the typical approach to EM has been to exploit modelling with definitive scripts (MWDS) as supported by the EDEN interpreter. This approach has limitations with practical and conceptual consequences. The Cadence environment, as developed by Nick Pope, the second author, offers an alternative framework - as yet less thoroughly explored - for supporting EM. This report - which has been largely written by the first author but draws in essential ways on Pope's ideas - is concerned with the impact that Cadence has had in exposing problematic ways in which MWDS has biased the conception and practice of EM, and the extent to which developing Cadence can help to redress this bias. It has three principal sections. The first (Section 2) reviews the perspective on modelling state-as-experienced afforded by MWDS, drawing on previous publications [2,3,7,8,12,13,24] but also paying closer attention to specific ways in which the conception and practice of EM can be linked to William James's treatment of 'terms in experience' in his essays on radical empiricism [18]. The second (Section 3) briefly introduces the Cadence environment and discusses its impact on the conceptualisation of the computer as an instrument, and on the perspective of the modeller. The third (Section 4) focuses on the new characteristics it brings to modelling state-as-experienced, and the ways in which it can address the limitations of MWDS. Future prospects for exploiting Cadence are also discussed. A brief introductory section is included by way of orientation on the 'current' and 'new' perspectives on modelling state-as-experienced alluded to in the title of this report.

1. Modelling state-as-experienced

The conceptual framework within which EM is conducted is typically described with reference to Figure 1. The modeller builds the construal on a computer with reference to an external domain. The modeller attends to two aspects of their experience: the current state of the construal with which they interact on the computer and the current state of its intended

referent in the external domain. In keeping with the fundamental tenet of William James's radical empiricism - that conjunctive relationships between two aspects of experience can themselves be "given in experience" [18] - the modeller aspires to build the construal and engineer its referent in such a way that the association between them is itself directly experienced. As will be further discussed in later sections of this report, EM contrives this experiential connection between a construal and a referent by establishing a close correspondence between observables in the construal and the referent and between the patterns of dependency and agency to which the construal and the referent can be subjected through interaction with both. The full significance of Figure 1 can only be appreciated by recognising the fluid nature of the *Construal*, the *Referent*, the *Context* and the *Understanding*: each of these elements of the modelling activity is open-ended and interactively shaped, and is realised afresh each time the construal is further exercised and explored.

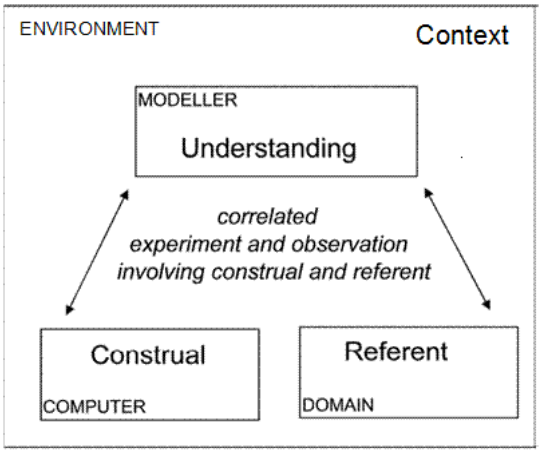

Figure 1: The semantic framework for Empirical Modelling

MWDS based on the EDEN interpreter has so far been the dominant practice in EM (cf. project archive [28]). An in-depth discussion of the semantic framework for EM, as depicted in Figure 1, can be found in Rungrattanaubol's doctoral thesis *A Treatise on Modelling with Definitive Scripts* [24]. Rungrattanaubol's account highlights two complementary aspects of building the construal. On the one hand, the modeller has to develop an interactive environment on the computer that affords experience that can be reliably revisited. On the other hand, they must also match this experience to the experience of the putative referent. Both of these activities contribute to the semantics of the definitive script that represents the current state of the construal. The former activity resembles the engineering of a virtual machine - it shapes the way in which the state that the computer presents to the modeller is made accessible and intelligible so that it can be placed under the reliable control of the modeller. The second activity shapes how the script acquires a richer meaning than 'being (e.g.) a particular configuration of visual elements on the computer screen' and is experienced as representing the current state of an external object of interest, such as the floorplan of a room, a scene from a computer game, or a partially completed Sudoku puzzle.

In the terminology of [24], these two ways of interpreting the script associated with the construal respectively relate to the 'internal' and 'external' semantic relations. As explained in [24], EM activity necessarily addresses both kinds of semantic relation: for instance, in one context, the modeller is concerned with technical issues to do with the quality of the visual display, but in another, they are concerned with tailoring the visual display to its intended

"real" interpretation. In the former context, the modeller's focus in the construal is on the script itself and how this is associated with the display as its referent; in the latter, on the display attached to the construal and how this is associated with an external meaning.

Though the modelling concept depicted in Figure 1 was first conceived with MWDS in mind, it can be invoked more generally (subject where appropriate to placing other technologies that can be a reliable platform for interactive experience in the role of the COMPUTER in Figure 1). The term 'construal' was introduced by David Gooding to characterise the kinds of artefact that Faraday developed in his experimental work, and this has been the archetype for "an EM construal" (see [3] section 9 for further discussion of this claim). Spreadsheet-based modelling has many characteristics in common with EM [3]. Other writings on EM (see e.g. [4]) draw a parallel between the relationship between the construal and the referent in Figure 1 and that between a musical composition and its content (as defined by the "extra-musical" experience of a composer and performer who engages with it). Approaches to software development that are based on prototyping activity aspire to exploit model-building that is similar in spirit (e.g. [15]), especially where the contribution made by intuition is taken properly into account [9].

In understanding the EM conceptual framework more fully, it is essential to consider activities that are more general than MWDS, yet at the same time to be able to discriminate between EM and other approaches. The aim of this report is to explore how the development of Cadence as an alternative tool for supporting EM sheds new light on the characteristic notion of EM: that of modelling state-as-experienced with reference to observables, dependencies and agency. To conclude this preamble, key characteristics of Cadence will be informally introduced in connection with some of the main issues they highlight where EM is concerned. These issues will be further explored in the rest of the report. Where appropriate, further details of Cadence will be given in part 2 of the report.

Enhancing computer support for construal: Faraday's exploitation of construals demonstrates that a construal need not be based on computer technology. As Gooding explains in [13,14], Faraday's construals are physical artefacts that may have some mechanical aspect that can be animated but whose sense-making role can only be understood in general by *imagining* the consequences of interaction. The advent of the computer has made it possible to develop far more sophisticated, integrated and automated interactive artefacts for sense-making and this has been one of the principal motivations for developing EM. The level of computer support for construal is still limited however. With reference to Figure 1, the full significance of a construal based on MWDS cannot be appreciated with reference to what is explicitly captured in the computer - for instance, the modeller still has to keep information about the context in mind, and to rely on being able to recall the names of observables and ways in which these are organised within files. Cadence is much better oriented towards using the computer to record essential information about context and structure.

Observables and dependencies: The way in which observables and dependencies are represented in typical definitive notations belies the generality of these notions as framed in EM. An observable is characterised in general in EM as "having an identity together with a current value or status". The interpretation that MWDS typically imposes constrains values to belong to an underlying algebra. This means that only very exceptionally and to a limited extent (as for instance in the ARCA notation [7,8]) can values designate references to observables. Such a narrow notion of 'a value' precludes values that capture the current

context for the modelling (cf. Figure 1) and cannot do full justice to *dynamical* observables (cf. [22]) that are process-like in nature and are experienced "in the stream-of-consciousness". MWDS also lacks the expressive power to accommodate relations between observables in their full generality. To express the *relationships* that connect observables that are so fundamental in the Jamesian account of state-as-experienced involves adapting the limited collection of built-in relations that underlying algebras and their associated canonical visualisations afford. For instance, observables may be framed within a list structure, or displayed in such a way that two observable stand in the same relationship that is experienced when we observe a geometric point together with its x-coordinate. Representing relationships that correspond to "observing an event" are also problematic where MWDS in concerned - there is no way to express the observation that ("x was previously 0 and is now 1"). Cadence has its roots in prototype-based object-oriented modelling, where structures are moulded to suit their semantic roles rather than being necessarily based on standard classes. It also incorporates means to manipulate 'stream-of-consciousness" observables and refer to their previous and next values.

The computer as instrument: In MWDS, the internal semantic relation is shaped by the standard visualisations that are afforded in the implementation of definitive notations. The control that the user exercises over this relation is likewise mediated by the syntax of these notations, and (as currently implemented in the EDEN interpreter) relies upon knowledge of how these are represented in the core 'lowest-level' Eden notation. In contrast to "an evaluator for definitive notations" such as EDEN, Cadence has been conceived with "modelling the computer as an instrument" in mind. At the core of Cadence is a reconceptualisation of the computational processing performed by the computer ("computing as navigation") that makes it possible to give a more holistic account of the computer as a physical artefact engineered from peripherals that are computer controlled. (It is in this respect that the Cadence environment resembles "an operating system", though in fact its appreciation demands a radically different conceptual framework from a conventional OS.) The significance of Cadence for EM is that it brings the experiential and instrumental perspectives to bear on computing technology, enabling the modeller to exercise much greater control over engineering the internal semantic relation.

Each of the three above issues concerns a transition from human cognitive and manipulative activities to computer-based activities that can be progressively more comprehensively automated. This transition is characteristic of EM in general, as is illustrated explicitly in EM modelling studies on themes such as heapsort [12,29], Sudoku solving [32], and ant navigation [20]. It broadly reflects the way in which the construction of knowledge is conceived within a Jamesian framework as originating with conjunctions in personal experience and potentially attaining assured objective status through association with shared interactions and objects. There is no suggestion that this process of construction, being essentially human-centred, can be fully automated, but Cadence highlights ways in which automation is introduced into the process to expedite understanding can be much more nuanced than MWDS permits. The merits of Cadence over MWDS become apparent in this procress as observables associated with structures and processes emerge in conjunction with contexts and modes of observation and interaction.

Cadence is particularly relevant to applications of EM that relate to software development [23]. In [22], Pope sets out his vision for plastic applications developed through using Cadence to support the migration from construal to program. To date, one of the most effective applications of Cadence has been in games design (cf. Stargate [22] Chap. 5), where

developing software through prototyping has an essential role in shaping the player experience. For such applications, sense-making activity making use of computer technology that informs the design of a piece of software should ideally be closely integrated with the computer implementation of this software. Whilst MWDS offers computer support for the sense-making activities of different human agents involved in development, and for modelling the interfaces to automated agents, it is necessary for the modeller to address the integration of these different agent perspectives. Cadence, in contrast, provides computer support for describing structure and process that - where this is appropriate - serve to model agency in context on the computer itself.

The ways in which human and automated agency can come together in EM activity are extremely rich and subtle. Within one application of EM, the balance between human and machine perspectives not only reflects the nature of the application, but can also shift as the modelling proceeds. A helpful analogy can be drawn between tools for EM and instruments for music-making. At one extreme, music can be made by the human voice without technological assistance, at the other, music can be synthesised electronically. The role of technology in generating music is apparent in each particular variety of instrument. The integration of different instruments can be effected entirely through human agency (as in a string quartet or via the conductor of an orchestra) or automatically (e.g. through mechanical, electromagnetic or electronic means as in different varieties of organ). In MWDS, individual definitive notations can be seen as the counterparts of musical instruments, and the role played by the modeller in integrating these when using the EDEN interpreter as analogous to that of the orchestral conductor. This analogy helps to clarify the role for the computer in supporting MWDS: advances in computer-related technology are significant even in relation to the development of traditional musical instruments (e.g. the electronic guitar). The much more radical shift in perspective on EM that Cadence promotes may be likened to the shift in perspective promoted by the advent of purely electronic music, where the construction and orchestration of instrumental sounds became merely *one* of the strategies that could be exploited in crafting musical sound.

The remaining sections of the report elaborate on the ideas summarised in this section. The main novelty in the report is in the discussion of Cadence in Section 3, which should be read in conjunction with Pope's account of Cadence in [22]. A reader who is familiar with EM principles may omit Section 2, though it does contain some new thinking regarding MWDS.

2. The MWDS perspective on modelling state-as-experienced

Much has been written about the connection between Empirical Modelling and William James's radical empiricism (cf. [4,5]). To understand the impact of Cadence as new tool for supporting EM, it is necessary to revisit the relationship between EM and radical empiricism with closer attention to how observables, dependencies and agents relate to James's account of 'pure experience' [18].

The richness and diversity of the experience and agency associated with an EM construal reflects the characteristics of "The World of Pure Experience" as characterised by James in his essay of that title [18:46-7]:

> Taken as it does appear, our universe is to a large extent chaotic. No one single type of connection runs through all the experiences that compose it. .... space-relations fail to connect minds ... Causes and purposes obtain only

> among special series of facts. The self-relation seems extremely limited and does not link two different selves together. On the face of it, if you should liken the universe of absolute idealism to an aquarium, a crystal globe in which goldfish are swimming, you would have to compare the empiricist universe to something more like one of those dried human heads with which the Dyaks of Borneo deck their lodges. The skull forms a solid nucleus; but innumerable feathers, leaves, strings, beads, and loose appendages of every description float and dangle from it, and, save that they terminate in it, seem to have nothing to do with one other. Even so my experiences and yours float and dangle, terminating, it is true, in a nucleus of common perception, but for the most part out of sight and irrelevant and unimaginable to one another.

When practising EM using definitive scripts (MWDS), the messiness of the experiential environment depicted in the above quotation is to a certain degree faithfully reflected in the loose agglomeration of observables, the plethora of metaphors being used simultaneously, the potential for open anarchic interaction that subverts interpretation etc. But the reticular qualities of experience to which James refers can only be fully appreciated when our attempts as rationalising them are also taken into account. As James observes in [18:132-3]: Experiences come on an enormous scale, and if we take them all together, they come in a chaos of incommensurable relations that we can not straighten out. We have to abstract different groups of them, and handle these separately if we are to talk of them at all. Whilst MWDS may be good at conveying "the chaos of incommensurable relations", it does not address what James identifies as the need to "abstract different groups of them, and handle these separately". And though being able to capture conjunctions is highly significant, it is also essential to be able to refer to the components of conjunctions.

Empirical Modelling based on MWDS can lead towards coherent structures. A construal that is initially conceived as a personal sense-making artefact can evolve towards what can be interpreted from an objective perspective as a program. As ilustrated in [12], for instance, EM can serve to trace a path from the informal preliminary conception of an algorithm to a formal specification. Supporting this evolution means developing computer representations and execution mechanisms that are efficient and program-like in character. The EDEN interpreter is not a tool designed for this purpose, and can be frustrating to the programmer on this account (to such an extent that gifted programmers lose patience with it and deprecate its use). A closer examination of how MWDS relates to James's account of 'pure experience' highlights some of the conceptual and technical issues to be addressed in developing better tools for EM.

First and foremost, an effective tool for making EM construals should support making connections that can be experienced between the construal and its referent. Such connections are instances of what James calls *conjunctive relations* that connect "terms" in experience. The notion of a 'term' is broad, and it is of the essence that a relation may itself be a term. Though James does not define "a term in experience" explicitly, it is implicitly defined through its use in [18] as referring to "an element that is directly experienced" and that is being viewed as atomic rather than relational in character (cf. [18:42]). James [18:44-5] classifies conjunctive relations according to their "different degrees of intimacy". The "most external relation" is that of being common to a universe of discourse. Next come the relations associated with simultaneity and time-interval, then those associated with space-adjacency and distance. Next are the "relations of activity, tying terms into series involving change, tendency, resistance, and the causal order generally". The most intimate relation is that

"experienced between terms that form states of mind, and are immediately conscious of continuing each other".

It is instructive to examine the way in which a definitive script (such as is used to represent the state of a construal in EDEN) links to this Jamesian account of conjunctive relations. James identifies two ways in which we make sense of our immediate experience: through discriminating between different elements ("terms") - apprehending "disjunctions", and through registering connections between different terms - apprehending "conjunctions". Traditional empiricist outlooks recognise disjunctions as given-in-experience; the characteristic feature of *radical empiricism* is that it also recognises conjunctions as potentially given-in-experience. Whereas traditional empiricism seeks to account for experience with reference to primitive terms such as are presumed given in the atomic sensory ingredients of immediate experience, radical empiricism contends that experience is not reducible in this way. To do justice to experience requires that we acknowledge that *relations* are also directly apprehended.

The primary concepts in EM are *observables*, *dependencies* and *agents*. In brief: an observable is an element in experience to which an identity together with a current status or value can be attributed; a dependency is a connection between a change to one observable and that to another that is given-in-experience; an agent is a component of experience which is deemed to be responsible for changing the state of certain observables. It is evident from practical EM models that an "observable" in a definitive script is associated with what James calls a "term in experience" of a particular kind. For instance, in the Sudoku model, the observables that are explicitly represented by identifiers in the script include: the entire Sudoku grid, the points that locate the individual cells, the digits that appear in the cells, the background colour of a cell, the list of digits that can plausibly be entered into a cell taking account of entries in the same row, column or region etc. Saying that an observable has "an identity and current status or value" associates an observable with what James identifies as "the most intimate relation" - the current status or value of an observable being linked to its 'next' status or value in such a way that (to paraphrase James) we are immediately conscious of the one continuing the other. In referring to the 'next' status of value of an observable, there is not necessarily any assumption that observation of time is being made - the continuity to which James alludes is associated with states of mind (cf. the way in which we speak of "pursuing a train of thought", and can resume work on a task after interruption "as if" in the same state of mind). James's notion of 'term' as an element of experience is more general than 'observable' in that it embraces effects so exceptional, transient or elusive (as for instance we may experience in brief exposure to a wholly unfamiliar context, in traumatic situations where "things happen so fast", or in dreams) that they are not amenable to being revisited in sense-making activity.

A dependency corresponds closely to what James identifies as a 'relation of activity' - it can be seen as expressing the idea that one change "causes" another (cf. how entering a digit into the Sudoku grid affects the plausible digits for other cells). An agent is something to which state-change is attributed. Both dependency and agency are in the first instance appreciated from the viewpoint of the modeller. Dependencies help to shape the modeller's sense of what change is under their control ("their agency"): in addition to those things that the modeller can directly change, there are the contingent changes that these entail. The changes that the modeller initiates are not necessarily or typically the only changes that can be observed. Agency may be invoked to account for the changes that are not initiated by the modeller. One aspect of the sense-making activity in EM is the identification of other agents and the

attribution of state-change to them. An important part of the process of exploiting sense-making in developing software using EM principles involves projecting this approach to understanding and shaping the observation and agency of the modeller to the other agents. It is then natural to interpret dependency and agency as relative to a particular agent. A key aspect of dependency and agency is that they are identified through open-ended autonomous action and are always provisional in character, being in principle subject to experimental refutation in some future enactment.

In the light of the above discussion, with reference to the "degrees of intimacy" of conjunctive relations outlined by James, it is apparent that observables and dependencies as perceived by the modeller are associated with the most intimate kinds of conjunction. Experience of EM has illustrated how model-building can begin with the development of personal construals and how this can in principle subsequently lead to the construction of objective system-like behaviours. Certain practical problems have been encountered in supporting the move from construal to program, however. To some extent, these problems can be attributed to the fact that modelling with definitive scripts based on the EDEN interpreter has been the main vehicle for practising EM. And, as will be now be explored, modelling with definitive scripts does not give good support for modelling relations that are "more external" in James's sense.

2.1 Definitive scripts and their limitations

A definitive script of the kind that specifies an EDEN construal is developed by introducing "metaphorical" counterparts for 'all' the observables of interest in a particular context. The metaphors that EDEN routinely makes available are line-drawing features (shapes, lines, points), display elements, such as panels with geometric and textual content, together with tables, lists, strings and scalars. Instances of these various ingredients serve as *metaphors* on the basis that their current value or status is presented to the modeller and the interpretations that they acquire is primarily based on the experience that this offers to the modeller. Specifically, the current state of script as a whole has some visual counterpart that forms one component of a conjunctive relation given in the experience of the modeller. (The visual counterpart may be a rich display with geometric and graphical content, but might also simply be a suitably annotated presentation of meaningful numbers - as in a spreadsheet, or a cricket scoreboard.) What is significant about the script is that the observables within it all have counterparts in the referent (cf. Figure 1), and that their current values are either evident in the visualisation, or are held in mind in the current context and can be readily made visible (cf. the way in which awareness of one's bank balance is implicit in a shopping context).

A characteristic feature of EM is that the observables and dependencies are fluid. As the modelling proceeds, different kinds of agency and patterns of interaction are identified. With reference to Figure 1, the construal, referent, understanding and context are all being concurrently shaped by the modeller's interaction and interpretation. New kinds of agent perspectives may become topical as understanding deepens, the referent or context is refined, and elements of automation are introduced. From a Jamesian perspective, this modelling activity can be viewed as exploring the terms and relations that are - or will come to be - given-in-experience in a context that is itself typically being negotiated.

The way in which a definitive script serves to support such modelling activity is well-illustrated by the Sudoku-solving construal studied in [32]. In looking at a Sudoku grid, many different observables are relevant. The primary emphasis in traditional modelling is on

abstracting those observables that are pertinent to a specific task in hand, so that the model is conceived as a simplification. In EM, in contrast, all manner of observables are potentially of interest throughout, and different observables are topical according to which patterns of agency, interaction and interpretation with the construal are currently the focus of attention. As a simple example, in developing the Sudoku construal the background colour of a Sudoku cell is normally a standard shade chosen to contrast with the digits in the puzzle. The choice of colour is then of practical and perhaps aesthetic concern - *can you see the digits clearly? is the grid pleasing on the eye?* After making an initial standard choice of background colour, the modeller might later revisit this choice, if for instance the way in which the grid was being lit became a matter of concern. In a variant of Sudoku that emerged from experimentation with the construal [31], the background colour of a cell was used as a way of giving the solver a hint as to the plausible digits for the cell. Specifically, distinct dark colours were assigned to the digits 1 to 9, and the background colour assigned to a cell was determined by dependency as a blend of the colours of the digits that are plausible for that cell. In this way, observables that had been hitherto of peripheral interest in the Sudoku solving modelling exercise - indeed the kind of observables that a traditional model to support Sudoku solving would fix at the outset and discount - played a most prominent role in a novel variant of the solving activity.

The qualities of the definitive script stem from the way in which it can capture different types of agency and the observables that are associated with these. For the human solver of the colour variant of Sudoku, the background colour of a cell is an observable that is registered in quite a different way from the incidental colours that might be used e.g. in the border that surrounds the grid. The experienced solver recognises that dark colours signify cells with few plausible digits, will recognise the strategic importance of observing when inserting a digit changes the background colour of a cell, and may be able to identify a unique value for a particular cell in a given row by observing that it has a distinctive characteristic relative to other cells in the same row. The choice of colours assigned to the digits 1 to 9 is an important factor in determining the level of support the construal affords for exploring this kind of observation. By giving the solver agency to modify this choice of colours, the qualities of the 'colour' metaphor for "set of plausible digits" can be explored.

When modelling with definitive scripts, the cumulative effect of introducing additional observables to the construal, considering new modes of agency and contexts for interaction and interpretation is to make high demands on the modeller's cognitive processes, powers of concentration and memory. It is hard to keep track of the names of observables in the script and the relations to which they are subject. These issues are particularly relevant if and when development reaches a stage where there is no longer the same need to explore and experiment, and parts of the model can be regarded as engineered to suit a well-established context. Subject to such engineering of context, the Sudoku-solving construal can be used as a basis for semi-automated solution, for instance.

In this context, other kinds of conjunctive relation become significant. To retain a conceptual grasp over the construal, the modeller needs to be able to organise observables so as to conveniently identify structural relations. They also need to be able to adapt and reason about the automated behaviours that are incorporated into the model. For this purpose, better methods of referencing and organising observables than EDEN scripts afford are required. To date, EM construals based on EDEN in effect attempt to deal with structural and dynamical relations by linking them to the perception of dependencies. For instance, an EDEN list with the definition

```
%eden
l is [a,b,c];
```

establishes a dependency such that l depends on its components a, b and c. This captures one aspect of a structural relation, in that the value of l is registered as changed whenever c is changed. This kind of dependency mechanism has been used to give limited integrity to geometric objects (e.g. through setting up dependencies between the corners of a square table so that all four corners move appropriately when one particular corer is moved etc). A similarly weak form of dynamical relation can be specified in EDEN through introducing an 'edenclock' observable to control a associated time parameter. These techniques do not do justice to the integrity of structural and dynamical relations however. In the case of structural relations there is no support in the EDEN symboltable for the idea that a, b and c are components of the object l. In the case of a dynamical relation, there is no satisfactory way in which to reference the 'previous' and 'next' values in the time sequence. These problems of reference are also in evidence when trying to specify dependencies between the components of an object or the values of a dynamical observables at different points on time. (These have motivated the idea of associating a 'mode of definition' with complex observables, but its implementation within the EDEN framework has proved to be too complicated.)

The limitations of definitive scripts as a way of supporting human sense-making and understanding stem in part from the way in which the state of the construal is being conceptualised. Definitive scripts are based on ideas that are rooted in the classical computing framework. A script is framed with reference to an underlying algebra of data types and operators that are a powerful means to express functional relations (as in approaches to software development based on "algebraic specification" [26]). No matter how sophisticated the underlying algebras involved, the primary abstraction associated with an observable is as a placeholder for a *value*. As the history of computing has shown, trying to find a coherent theoretical framework within which to handle values and references (such as are implicit in data structures and in objects in the OO sense) is problematic. The problem is topical in practical computing in relation to two particular challenges: in specifying the architecture and operating system for a machine, and in formalising software development activities.

To overcome the limitations of definitive scripts where expressive power and intelligibility are concerned, a better conceptual framework is needed. As has been discussed in previous EM papers when considering the potential for implementing definitive principles at lower levels of abstraction, the idea of an observable as designating a *value* is too limited. What is required is something at least as expressive as the kind of state representation that features in machine code, where a word can not only designate a value, but also an address, or an encoded action (cf. section 4 in [2]). Another source of inspiration for more flexible representations of state are object-oriented practices - particularly those based on prototyping of objects, which have practical promise, but are even more difficult to formalise productively than those that predominate in the class-based OO culture.

3. Cadence as an approach to modelling state-as-experienced

The Cadence environment [22] has been conceived and designed by Nick Pope. Its conception reflects the influence of both machine architecture and prototype-based OO perspectives. Cadence supports modelling and programming with several features in common with Empirical Modelling, but many significant advantages, especially in relation to the management of observables and dependencies and to topical issues of performance and

integration with conventional programming environments. Specifically, Cadence is intended to serve *both* as a modelling environment that emulates and in key respects surpasses environments for EM as currently conceived, *and* as a programming environment that exploits a paradigm somewhat related to dataflow that affords a much higher degree of integration between modelling and programming than any existing proposal for EM environments offers. With reference to the semantic framework for EM depicted in Figure 1, and in comparison with existing EM tools such as the EDEN interpreter, Cadence deals much more explicitly and effectively with specifying and manipulating the "Context" and with relating the state of the "Construal" to that of the "COMPUTER".

One of the most significant distinctions between Cadence and EDEN is that it allows the clustering of observables into objects, and offers much greater expressive power where managing and referencing groups of observables are concerned. Whereas EDEN is ill-suited to dealing with large numbers of observables, and relies heavily on syntactic features such as definitive scripts, Cadence is intended to cope with orders of magnitude more observables (such as might be required in modelling multi-agent simulations in a virtual environment). When appropriately implemented, Cadence will address the problems that arise in managing such observables at the user-interface by introducing a form of graphical interface (thereby avoiding the need for the modeller to explicitly devise and record observable names), and address the complexity of updating and processing by exploiting concurrent implementation on multi-processor architectures. An EDEN-based implementation cannot possibly deliver to such a specification.

Cadence highlights many of the significant limitations of MWDS, in the process challenging previous claims about the qualities of definitive scripts. In the first instance, Cadence makes it possible to conveniently specify and manipulate object-like *structural relations* [22] in ways that pose comprehension and implementation problems when ventured in definitive scripts. This gives enhanced expressive power where representing kinds of dependencies that are commonplace in experience is concerned. It is also apparent that - contrary to the critique of object-orientation that has often been made under the auspices of the EM project [14], to the effect that identifying objects goes beyond the primitive notion of observable that is empirically given in experience - Cadence is in some key respects *more* primitive than a definitive notation, in that it does not presume the notions of "type" and "operation" that are needed to identify an underlying algebra. This puts a new perspective on issues that have long been topical for EM regarding the status of functions in an environment for modelling with definitive scripts. In particular, it highlights the distinction between "functions as vehicles for expressing dependencies that are not amenable to observation in the same sense as observables on the left-hand side of definitions", and "functions as themselves expressible using definitive scripts, and thereby subject to some form of description in observational terms".

Another significant respect in which Cadence is more expressive than EDEN is in its treatment of observables that are associated with what Pope calls "dynamical relations" [22]. Such observables have an analogue time-dependent nature, and are typically subject to the kind of definitions that feature in a dynamical system. In EDEN, attempts to model such observables have typically used the 'edenclocks' construct, but this is unsatisfactory in key respects. The qualities of a dynamical observable cannot be captured simply by making its value dependent upon real-time - its character more closely resembles the solution of a differential equation that is to be computed with an accuracy contingent on context using the most appropriate and feasible step-size. Within Cadence, in the context of such approximate

computation, it is possible to refer to the 'previous' and 'next' values of dynamical observables. In this way, process-like behaviours to which these observables may be subject can be specified by using dependency relations to relate future values to current and previous values. It is also possible to refer to and define the instantaneous value of such an observable without interfering with the definition of its time-dependent behaviour. A further advantage of Cadence is that it is possible to specify events with reference to the current and previous values of dynamical observables.

3.1 Cadence orientation on state-as-experienced

Construals within the Cadence environment are fashioned from tokens that are counterparts of terms in experience in the referent. In keeping with a Jamesian outlook, these terms may be relational in nature.

The possible terms of experience that might conceivably arise in making a construal are so numerous, diverse and contextually determined that it is inappropriate to imagine *a set* of tokens to represent them all. In preference to this, in the spirit of object-oriented development, the modeller is able to invoke 'new' tokens (cf. OIDs) to represent newly topical "terms in experience" as they are encountered. Following [22], the set of tokens associated with the current moment in the EM activity will be denoted by R.

In order for R to serve its role in modelling state-as-experienced, there must be counterparts for terms in experience that are relational in character. In order to handle such relations given in experience, it must be possible to refer to connections between elements in R. For this purpose, the set of currently topical tokens is complemented by a binary mapping $\varphi : R \times R \rightarrow R$. The mapping φ is a well-defined function: to ensure this, R includes a "NULL" token which stands for a term in experience that signifies 'empty' or 'nothing', and $\varphi(x,y)$ has the default value NULL where this is deemed appropriate.

The relational structure established by (R, φ) can be represented as an edge-labelled graph Γ which has tokens in R as its nodes, and such that there is an edge labelled y from node x to node z if $\varphi(x,y)=z$. Conceptually, the explicit relational structure expressed in Γ is the counterpart of the set of current observables in a definitive script together with *their current values*. From an EM perspective, the explicitly specified structure of Γ, like the current values in a definitive script, relates to state-as-experienced in the same way that a snapshot relates to live experience. The snapshot captures what can be observed in the current moment by inspection, but does not take acount of what can only be discerned through intervention (cf. how a photocopy of a spreadsheet dispenses with the definitions behind the cells).

As visual art shows, the contemplation of a snapshot can be itself a rich experience. The picture that is being viewed is not construed as itself changing, but the way in which the human viewer regards it- e.g. through paying attention to different features and the relationships between them - effects changes in the state-as-experienced. Experience of this nature that is associated with implicit state-change is of course represented in EM models developed through MWDS (as e.g. when we give focus to a particular square in the Sudoku grid by redefining the observable '`currsquare`'), but is typically only supported clumsily and informally (modes of inspection being one of the aspects of interaction with the construal that the modeller has to document or commit to memory). By comparison, the relational structure Γ gives direct and natural support for implicit state-changing activities - consider for instance

how well-adapted Γ is for representing an experiential object like a Cayley diagram [7,8] or a look-up table that invites interpretation through inspection.

The rich snapshot of state-as-experienced that Cadence affords is complemented by two ways of expressing expectations about changes of state. To this end, the structure of the graph Γ, as determined by the mapping φ, can be specified implicitly by introducing formulae to define the nodes φ(x,y) for any x and y in Γ. Two modes of definition can be used - "is" and "willbe" (denoted by ":=").

An 'is' definition establishes a dependency relation similar to that which features in a definitive script. In experiential terms, it expresses an agent's expectations about the consequences of action ("if I make this change, these attendant changes will result as if indivisibly" / "if this change occurs as a result of action on the part of another agent, these attendant changes will result *as if* indivisibly"). Dependency relations of this nature have played a central role in the account of EM to date. In a definitive script, the value of an observable is expressed in terms of the values of other observables using a formula based on operators drawn from underlying algebras. In Cadence, the role of the 'is' definition is to construct *structural* dependencies within Γ. This makes it possible to specify that changing one incidence relation, as in redefining φ(r,s) (so that the edge labelled s emanating from the node r is now linked to a different node in Γ), will simultaneously change other incidence relations (viz. those that are specified by means of an 'is' definition in which the expression φ(r,s) features on the RHS).

The 'willbe' definition in Cadence has no precedent in MWDS. Its introduction reflects a different stance towards state-as-experienced from that underlying MWDS. Specifically, Cadence acknowledges a complementary - or perhaps alternative - perspective on experience of change that is in keeping with the "stream of consciousness" metaphor with which William James's name is often linked. Such a perspective can be invoked whenever it is appropriate to think of the modelling context in Figure 1 as affording stream-like observation of a linear sequential nature. The terms of experience to which "willbe" definitions are attached in Cadence are then *dynamical* in character (cf. Pope [22]) - they resemble observables that are analogue and process-like and are subject to continual change. Modelling dynamical observables necessarily entails linking discrete observations made in sequence in such a way that the time interval between observations is 'as small as possible' and in aspiration 'arbitrarily small'. The form of a 'willbe' definition then resembles that of an 'is' definition but is to be interpreted as asserting that the current value of φ(r,s) is the previous value of its defining formula. A 'willbe' definition makes it possible to define the current value of φ(r,s) in terms of its previous values without circularity.

In practising EM in the Cadence environment, the definitions that currently determine the mapping φ also determine the current state of the construal in Figure 1. The modelling activity proceeds through manipulating these definitions; as a result, φ is modified, in some cases through the identification of new terms in experience and the introduction of new tokens from R to correspond to them. As in the development of a definitive script, each redefinition of the graph Γ can signify a *refinement* of the modeller's construal of state-as-experienced or a *change* in the state-as-experienced.

The graph Γ is one of several ways of representing the binary function φ so that it can be experienced by the modeller. The special-purpose DASM notation [22] can also be used to specify the current state of the mapping φ. In the current implementation of Cadence, both

types of realisation feature in the interface. In essence, the graph Γ is displayed as an inventory of nodes and their definitions organised in such a way that nodes may be anonymous and there is a link labelled y from node x to node z whenever φ(x,y)=z. The interpreter also has a text input window into which DASM definitions can be entered.

In EM, the character of the relationship between the construal and its referent typically changes so that the context affords progressively more scope for identifying structural relations and process-like observables. With respective reference to the notions of 'external' and the 'internal' semantic relations introduced in section 1, EM can be conceptualised as exploratory human-centred activity that proceeds in parallel with the development of a machine-like environment that is well-adapted to supporting this exploration. In MWDS using EDEN, the "machine-like environment" is typically at a quite different level of abstraction and sophistication from the actual computer hardware on which it executes (cf. Figure 1). Such an environment might take the form (for instance) of a line-drawing script that incorporates the kind of observables and dependencies that are well-oriented for shapes that resemble drawing room layouts. In that setting, the way in which the definitive script is visualised is almost always given by default on the basis that the observables in the script are of a standard type (e.g. a geometric line) with a standard realisation (e.g. the corresponding pixellated line on the display). Cadence gives the modeller to scope to craft different kinds of agents that can 'observe' suitably demarcated nodes or fragments of Γ and attach a perceptualisation to them in a manner that is suited to the characteristics of the underlying machine. The Warwick Games Design library [33] illustrates how this technique can blend customisation to the actual machine with greater power to express high-level abstractions of a machine-like character such as are associated with structures and processes.

This sketchy overview of how Cadence can support modelling state-as-experienced indicates its potential in two complementary respects: as a way of establishing machine-like environments, and as a way of enriching the experience of the modeller. Each of these themes will be developed in the following two subsections, with further reference to how the use of Cadence compares with MWDS using EDEN.

3.2 Conceptualising the computer as a modelling instrument

Unlike MWDS in EM, the development of Cadence has been centrally concerned with how traditional computers actually operate, and how this can best be exploited in flexible modelling. A natural point of departure is considering how Cadence can provide a computational model to support an unconventional programming paradigm. As will emerge, the true goal in Cadence - as in EM - is a broader framework for thinking about the nature of the computer, and the role of the computer within machine-like architectures that are radically different from the classical computer.

The computational paradigm which Cadence natively affords can be described as "computation as navigation (C-a-N)". C-a-N relies entirely on interpreting structural relationships within the graph Γ. The principles of C-a-N will be sketched here with reference to how arbitrary computable functions involving booleans and integers can be evaluated (for more details, see the account in [22]).

It is initially helpful to consider boolean functions alone. In keeping with the *machine-like* interpretation of Γ, it is necessary to assume that certain nodes of Γ can be given a standard interpretation as nodes that correspond to the boolean values T and F, and these nodes are

thereafter to be understood as 'representing' these values. In effect, 'true' and 'false' are here being invoked as familiar 'terms in experience'. By introducing additional nodes associated with φ (true, AND), φ (false, AND), φ (true, OR) and φ (false, OR) the boolean operators can then be modelled within the graph Γ in such a way that "computing an AND or OR" is reduced to a matter of elementary navigation. Thus, φ ((true, AND), false) is the node 'false' etc.

The principle of choosing particular nodes in the graph Γ to designate values and operators on these values, and supplying a suitable incidence structure to reduce operator evaluation to navigation of the graph can be generalised. It is first necessary to create a virtual extension of the the graph Γ in which there is a node to correspond to each integer value as a term in experience. (There is here an assumption that each integer value is a legitimate "term in experience", though of course the degree of experiential familiarity we have with different integer values varies enormously.) As discussed in Pope [22], the incidences that are required to perform 'arbitrary length' integer arithmetic with standard operators (traced out by edges associated with the nodes φ (1,+), φ (2,+), etc) can be described within this virtual extension of the graph Γ - this is the approach that has been adopted in the current Cadence implementation. The essential idea behind this technique for extending the range of values and operators available to the modeller in Cadence is that of emulating a look-up table for operators, so that evaluation of an arithmetic expression can be effected simply through navigation.

It is instructive to consider the evaluation of an arithmetic expression within the "virtual extension of Γ" in experiential terms. Whilst it my be deemed possible for the human imagination to conceive the infinite incidence structure associated with integer addition, this infinite structure must be treated differently from an experiential perspective - in much the same way that a Turing Machine tape is to be regarded as *unbounded* rather than infinite. Specifically, given an arithmetic expression, there is a finite portion of Γ that has to be traced in its evaluation. On this basis, evaluation of an expression can be regarded as navigation that progressively discloses a finite portion of the infinite graph.

As discussed in [22], it is possible to frame the evaluation of more general expressions that involve conditionals (see [22], p94) and recursive definition (see [22], p150) within a similar conceptual framework. Such evaluations require dynamic virtual extension of Γ that is specific to the expression being evaluated in ways that are much more difficult to conceive as exploration of an extant infinite structure. It is also challenging to find ways to formulate such expressions, as the discussion of the sugared if-construct and of meta-relations in [22] illustrates. This aspect of Cadence has yet to be fully addressed in its design and implementation. In due course, it may be relevant to issues such as the transformation of construals to programs that is required in some applications EM [23]. In the context of this report, where the emphasis is on an experiential account of Cadence, it has only peripheral interest.

In its full generality, the C-a-N treatment of functions is in the tradition of the classical universal Turing machine - it aims to show how in principle any computable function can be abstractly computed in the Cadence environment. From an EM perspective, there is no propect of framing a generic way of modelling state-as-experienced that is based on such an abstract computational foundation. The experimental introduction of computable functions as putative 'observables' was first explored in MWDS, when "a definitive notation for functions" was developed (cf. the admira prototype for specifying Miranda functions in a definitive

script [27]). Though this at first appeared to be a powerful step towards enhancing the power of EM so that it could conveniently be used to construct any computable function, the correspondence in experience that is characteristic of EM (cf. Figure 1) is not respected by extending MWDS in this way. Showing that any abstract computation can in principle be carried out within the Cadence environment is equally irrelevant to the principal goals of EM. What calls these constructions into question as vehicles for EM is the difficulty of connecting such a construction to the modeller's current experience as envisaged in Figure 1. In effect, the difficulty of admitting either treatment of computable functions into EM is related to the fundamental implausibility of "formalising experience", whether this is ventured through a precise account of functionality as implemented on the computer or of functionality as conceived by the model-builder.

In the classical approach to computation, there is a strong separation between the abstract computational semantics and the concrete entities within the application domain to which this computation ultimately relates. In building construals in EM, the semantic relation between the construal and the referent is shaped with direct reference to experience at all times, as depicted in Figure 1. A powerful feature of Cadence is that provides a richer metaphor for state that a definitive script in which both structural and dynamical features can play a more immediate role. The issue of how Cadence gives support for grounding of semantics is further discussed in section 3.4.

As far as EM is concerned, the most important role for the elegant conceptualisation of expression evaluation in Cadence discussed above is in relation to creating a model of a computer-based system that is directed at capturing state-as-experienced. The core notation underlying Cadence ("DOSTE") was originally conceived as the basis for an alternative operating system. Cadence observables are accordingly well-suited to describing the current state of the machine and peripherals. For instance, there are dynamical observables in Cadence that correspond to the current status of the mouse buttons, and these can be used to express the notion of a mouse-click as an event:

```
@mouse = (@root input mouse buttons);
.mousedown := {@mouse left};
.mouseclick is {.mousedown and (@mouse left not)};
```

The interpretative behaviour of the machine can be controlled through explicit observables, as can the interface to the modeller. In principle, the file system itself can also be represented within Cadence.

When used in conjunction with models of devices, arithmetic expressions framed in Cadence provide a powerful way to model different kinds of machine architecture. The intimate connection that can be made between computation by navigation and computer hardware is illustrated by the way in which 4- and 8-bit integer adders can be emulated in Cadence (cf. the calculator in Cadence study developed by David Evans for WEB-EM-7 in 2010-11). The integration of boolean and integer expressions in this exercise highlights the coherence of the internal representations that can be achieved within Cadence (to be contrasted with the complexity of the dependency structures needed to communicate between different definitive notations in EDEN). It also has clear potential for exploitation on concurrent architectures.

A prominent theme in EM is that the crafting of a "virtual machine" can be regarded as a crucial aspect of complex computing systems development [10]. EM activities that are at first

exploratory and experimental lead to the identification of ways in which processors and devices can be configured so that they conform to predictable stimulus-response patterns - in this way, they lay the platform for further experiment and exploration that can address the high-level modelling objectives more clearly and directly. A feature of EM is that the 'virtual machine' that is developed in this way evolves as the modelling activity proceeds and must as far as possible always be open to revision and extension. By its very nature, Cadence is exceptionally well suited to this engineering task.

In EM, two quite different kinds of observable are represented within this development of this 'virtual machine'. There are those observables that reflect built-in characteristics of the underlying hardware devices that cannot be negotiated. There are others that are manufactured by the modeller through configuration of these devices and the management of the interactive context surrounding them.

In MWDS with EDEN, the operators supplied by standard definitive notations can be seen as built-in to the modelling instrument, and to be viewed as oracles by the modeller - that is to say, without reference to how they may actually be implemented. (Consider the way in which, in some contexts for EM - such as pertain when the construal is made without using the computer, the functions are not computed but are implicit in the modelling medium.) When additional functions are introduced by writing procedural code in EDEN (that is to say, by invoking orthodox programming techniques not within the scope of EM activity), this is then best conceptualised as adding to the repertoire of oracles that can be deployed. An advantage of modelling with reference to these standard operators as a base for the virtual machine is that they have an objective status for the participants in the modelling activity and promote effective communication between them. Even prior to the modelling activity, the interpretation and visualisation of standard algebraic data types is already an established part of the 'shared understanding'.

The drawback of basing the modelling exclusively on such standard types and operators is that this constrains the modeller, who may not be able to develop the structures, processes and metaphors that are best suited to the application and matched to the physical characteristics of the computing technology. Whilst it has been possible (at least at the level of proof of concept) to use MWDS in EDEN to devise operations in exceedingly subtle ways (consider e.g. the way in which heapsorting can be traced to its primitive observational roots through MWDS [12]) and to link EDEN models to hardware devices such as keyboards and wiimotes, potential extensions to the virtual machine of this nature come at a heavy cognitive and computational cost.

In Cadence, the base for the virtual machine is much better oriented to the modeller's cognitive requirements and to the nature of the diverse underlying architectures as these may be encountered in an EM application. A key concept is that the agencies that operate in an arbitrary computer-supported environment can be modelled in Cadence with reference to state-as-experienced, and that these models can be "glued together" in creating the virtual machine. As has been illustrated (e.g. in using Cadence in the design of computer games featuring sophisticated graphics and exploiting wiimotes, and by prototype models for a video wall and sound generation (cf. [22] Chapter 5) being able to develop structural and dynamical observables that are well-matched to the characteristics of these agencies is crucial. This approach promises to support EM by delivering greater intelligibility, flexibility and efficiency. The aspiration is to exploit Cadence in integrating a wide range of devices and software utilities, enabling communication and conceptualisation between components that

exploit different computational paradigms, and providing diverse and pluggable interfaces for the modeller.

3.3 A helpful analogy

The kind of support that the Cadence environment affords for EM is radically different from that given by MWDS. It nature is also (by comparison with MWDS) as yet little explored in practice. This makes it difficult to develop a clear understanding of the relationship between Cadence and MWDS. This section aims to address this difficulty by introducing an analogy between Cadence and EDEN as instruments for EM and instruments that can be used for making music.

A natural parallel may be drawn between the computing environment supplied by EDEN or Cadence and an instrument. One aspect of an instrument is concerned with giving the instrumentalist control over the effects that are generated. This involves a particular kind of interaction with the instrument (cf. "tuning the piano / setting up the computer display") that is different in nature - and complementary - to characteristic use of the instrument (cf. "playing the piano / using the computer"). An important consideration in playing an instrument is the extent to which the instrumentalist is able to engage with the characteristics of the instrument itself (e.g. the pianist is not expected to - and cannot whilst playing, or perhaps even in principle - retune the piano). Cadence is intended to provide an environment that has the qualities of a 'plastic' instrument, where the instrumentalist can both play and adapt the instrument - possibly even in the process of playing. By comparison, a definitive notation within the EDEN environment resembles an instrument that lacks plasticity, so that in MWDS the flexibility and expressivity to which EM aspires stems from the way in which their use is coordinated by the modeller.

Some of the ways in which Cadence might assist the modeller achieve these goals are illustrated by considering the challenges to comprehension posed by the EDEN-based Sudoku construal at [32]. In interacting with the construal, the modeller is obliged to manipulate a space comprising some five thousand explicit observables. These observables are only loosely structured through the use of lists; in addition to this, the modeller must hold in mind many auxiliary features of the state that are not so explicitly represented in the script. Whilst the explicit observables have traditional values, it is clear that in some contexts, the modeller conceives them as having other significance - they may be attributes of objects (e.g. the observables associated with a cell in the grid), or in effect references to objects (e.g. the middle region of the grid). Over and above this, there are meta-properties of the interaction with the construal that are associated with different contexts for observation, modes of agency, and foci of attention. Being able to employ Cadence to introduce object-like structures can reduce the cognitive load on the modeller, and potentially enable the model-building to address more complex issues that are beyond the scope of what EDEN can support. For instance, it would be valuable to be able to record, reference and recall partial solutions in a convenient manner.

The fact that Cadence has the qualities of an operating system and incorporates such a comprehensive model of the computer state is most helpful in this management role. For instance, a common device in managing EDEN construals is use of the file system as an auxiliary resource for representing and manipulating state - setting up a model typically involves the systematic introduction of a family of files in a prescribed order that is recorded in a run file. Changes in context, focus of attention, and agent perspective are then effected

by introducing new files of definitions. Being able to integrate these files into the core modelling environment, as envisaged in Cadence, eliminates potentially overwhelming problems of establishing, maintaining and recalling a coherent file organisation for a construal. A complementary issue is maintaining the computer environment itself so that it serves the appropriate instrumental role for the human agents who interact with it. Certain construals have to be "tuned" for instance, in order to change state at an appropriate pace (a feature that is much in evidence in particular when running an older model on new hardware): having standard representations for these parameters within the modelling environment, as in Cadence, gives human interpreters direct control over them.

In their role as modelling instruments, both EDEN and Cadence present an interface through which the modeller can conveniently manipulate and view the state of the construal. That is to say, in addition to experiencing the construal as (e.g.) a screen display the modeller must in some way simultaneously experience the current state of the definitive script or the binary function φ that describes this perceptualisation of the construal. With reference to a Jamesian account, the modeller will be aware of terms in experience that do not belong to the external semantic relation but have only internal significance for moulding the construal.

In Cadence, the nodes of Γ typically correspond to terms in experience with external significance. The edges of Γ, in contrast, in general do not. By way of clarification, in state-as-experienced, terms are given in association with myriad potential relations that may or may not be registered as significant as-of-now (consider for example the pairs of objects in your current field of vision, and distances between them). The role of the function φ thus directly relates to the computer *as an instrument* - enabling the modeller to refer to the implicit conjunctions between terms-of-experience that may acquire importance in the current context (consider how the relation between two objects, hitherto unremarked, becomes topical when one is about to collide with another). An interesting parallel may be drawn with the way in which the generators are chosen in specifying a Cayley diagram for a group, and the way in which this choice can illuminate the group structure and make it possible to reference group elements conveniently.

To appreciate the full subtlety of the experiential considerations involved in giving tool support to EM, it is helpful to make an analogy between EM and music making. In this analogy, a definitive notation may be seen as the counterpart of a particular species of orchestral instrument, and the values within its underlying algebra as musical sounds that can be generated by playing this kind of instrument. An observable associated with the definitive notation is then the counterpart of a specific instrument within the appropriate family. With this interpretation, MWDS is the counterpart of composing and performing orchestral nusic, with the modeller cast in the roles of composer and conductor.

Invoking this analogy, Cadence resembles a synthesiser, whereas EDEN has more of the characteristics of an orchestra. The synthesiser can serve directly both to generate new sounds and to imitate the sound of existing instruments; with investment in interfaces etc it can then readily supply new instruments and emulate traditional ones. The orchestra, by comparison, is based around well-established instruments whose individual characteristics are not negotiable in the same way, and the generation of sound from the orchestra necessarily requires great human investment of effort on the part of skilled instrumentalists aimed at the overall coordination of potentially ill-matched effects.

As a modelling environment for developing EM construals, Cadence has a versatility comparable with that of the synthesiser as a means of generating musical sound. A few illustrative examples hint at the rich interpretative possibilities, but a full exploration of how this analogy can assist the conceptualisation of EM activities is left to the reader's imagination. Modelling with the Abstract Definitive Machine (ADM) [11] may be viewed as resembling orchestral rehearsal. Cadence affords connections between what might be conceived as independent instruments that may be likened to the use of a sound mixer for an orchestra in the recording studio , in contrast to EDEN which resembles a live orchestra that offers no alternative to human mediation in connecting the sounds of different instruments. Cadence accordingly gives the modeller more control over the modelling experience, promising more consistent results in exercising a construal, whereas MWDS with EDEN emphasises that interaction with a construal has the qualities and hazards of a 'live' improvisation or performance. But the comparison between Cadence and MWDS as a vehicle for EM must go beyond the conceptual frame that is familiar from MWDS.

The full force of the analogy between the capabilities of Cadence and a synthesiser is only appreciated by recognising that there is no obligation in synthesising music to conceive sounds as generated by a body of instruments in the conventional sense (compare a synthesiser with a traditional pipe organ, which also provides a means to coordinate a collection of instruments whose characteristic sound cannot be altered). From the perspective of the composer of electronic music, whose primary interest is in the effects that can be created using the synthesiser, there is something artificial about conceiving a composition on the basis of devising and coordinating the interaction of conventional instruments. Taking this approach does not do justice to the fundamental conception of the synthesiser, which arguably merits an alternative style of composition that is oriented to its particular exceedingly general and flexible mode of representing musical sounds. The synthesiser makes it possible to specify musical sounds directly with reference to their physical description as wave forms that vary over time. The composer is then able to take control of the way in which these wave forms are combined and shape their physical and temporal characteristics. In this same spirit, Cadence affords richer means than MWDS both to contrive and to connect the counterparts of terms in experience.

4 Comparing and contrasting Cadence and MWDS in EM

This section concludes the discussion of Cadence as a vehicle for modelling state-as-experienced by summarising and elaborating some of the key ideas introduced in previous sections, and re-examining them from the perspective of a modeller familiar with practising EM using MWDS.

The synthesiser-orchestra analogy is helpful in getting a conceptual grasp on the relationship between modelling state-as-experienced using Cadence and using MWDS. The thrust of modelling with definitive scripts is to create instruments ("definitive notations") and coordinate their use in a distinctly messy human fashion. In this approach, the values of the observables (preconceived in association with the underlying algebras) are already engineered and familiar to the modeller. The Cadence approach to modelling state-as-experienced is radically different from modelling with definitive scripts in that it is not primarily rooted in establishing relations between terms in experience and *'values'* (though relations of this nature can be invoked). The intention in Cadence is after all to give computer support to the much richer range of relations between terms in experience that have to be handled informally by the modeller in MWDS (consider for instance: the relations that are

implicit in the syntactic forms of observable identifiers - both user-contrived and automatically generated in a definitive script; the coincidental relationships that are apparent in visualisation - as in the visual proximity and configuration of points and lines that make up a Donald object such as a digit; the relations that - to a limited extent - are expressed in higher level notations such as Donald and Scout but disguised or discarded in the translation to Eden). In the same way that electronic music synthesis was introduced as a new paradigm for 'making music' that can dispense with the traditional conception of 'established instruments' - and indeed the whole conception of 'instrument', so Cadence subverts the idea of 'a standard value' - and indeed the very idea of 'a value' itself. In effect, Cadence favours a conceptualisation in which all terms in experience are understood with reference to relations that need not necessarily be of the same nature as observable-value relations.

This reorientation towards relations in Cadence is evident in the way that the graph Γ can itself (e.g. as in the graphical user interface in the current implementation) serve as a metaphor for state-as-experienced. In the first instance, the graph Γ already reflects a different stance on the idea of an *observable* as this is represented in MWDS. As discussed in section 3.1, each node of Γ corresponds to a term in experience, but not all nodes may be conveniently interpreted as observables with a conventional value or status of the kind that is expected in MWDS. With reference to the mapping $\varphi : R \times R \rightarrow R$, there are three ways in which a specific node α features in φ - in relationships of the form $\varphi(\alpha,*) = *$, of the form $\varphi(*, \alpha) = *$, and of the form $\varphi(*,*) = \alpha$. The nature of the semantic relations, both internal and external (cf. section 1), in which the token r can participate depends on how it features in all these three templates for relations. By way of a simple illustration, with reference to section 3.2, the node 'true' that is given the standard interpretation as the boolean value T and the node φ(true, OR) will feature in characteristic ways in relations of the above templates in a Cadence model. There will be a directed edge labelled 'true' in Γ that connects the node φ(true,OR) to the node 'true' for instance. There will be expressions matching the template $\varphi(*,\text{true}) = *$ in all conditional expressions. Nodes such as φ(true,OR) and $\varphi(1,+)$ do not have a value or status in the traditional sense - this is reflected in the way in which they feature most prominently in relations of the form $\varphi(\alpha,*) = *$ amongst the three relational templates above. A different kind of observable that is likewise uncharacteristic of MWDS might be a node 'name' that was introduced to the Cadence environment to attach a textual identifier to each node, which would then feature as a generic attribute of a node in relations of the form $\varphi(*,\alpha) = *$.

In [22], Pope uses 'observable' to refer specifically to the occurrences of r that match the template $\varphi(*,\alpha) = *$, or equivalently correspond to an edge in Γ that is labelled by the node α. The rationale for this is that an expression of the form $\varphi(*,\alpha) = *$, such as $\varphi(c,r) = v$ can be naturally interpreted as 'in the context c, the observable r has the value v'. The association of v with a *value* in connection with a relation on the template $\varphi(*,*) = v$ reflects the idea that $\varphi(c,r)$ is being assigned a meaningful term in experience - something that is familiar and generic (such as 'true' or '2') which has an interpretation independent of this specific setting for its use. In MWDS, this notion of 'terms in experience that are familiar and generic' is elaborated by invoking underlying algebras with richer values that admit standard interpretations; it has also been developed in such a way that such terms in experience can be dynamically crafted in the context of the modelling activity. In this report, it will be convenient to acknowledge that all terms in experience that have counterparts in Γ can be deemed to be 'observables' in a generalised sense (cf. the discussion in Section 2), but that - unlike observables in MWDS - their 'status' will be determined by the relations matching *any* of the three templates in which they occur. (This alternative choice of terminology does not

have serious semantic implications, since knowing the occurrences of α that match the template φ(*,α) = * for all nodes explicitly determines the edges of Γ and thereby implicitly determines its nodes.)

Generalising the notion of observable enables the modeller to express dependencies that are difficult to capture in MWDS with EDEN. For instance, it is possible to make a good construal of a notion like "the next customer", for which the characterising observables are of a different nature from the personal attributes of the customer themselves, and where it is appropriate to introduce an observable that has the status of a selector of a "person id" rather than a comprehension of personal characteristics. It is also possible to model the way in which (e.g.) switching the specification of a gender observable from "male" to "female" changes the content of a selection of people from "brothers" to "sisters". Modelling such relations in MWDS with EDEN is by comparison both inconvenient and inelegant: it involves introducing operations on lists that are described in procedural terms. It also introduces technical and conceptual problems where referencing and redefining the components of functions that return lists is concerned.

A fundamental issue in EM is the manner in which the conjunctive relations that link a construal to its referent are grounded. In MWDS, the key element that makes it possible to infer the correspondence between observables in the construal and observables in the referent (and indeed to experience the conjunction between corresponding observables, in much the same way that we connect the cells in a column in a spreadsheet with the marks of individual students on a particular module) is the dependency between observables set up within the script. It is on this basis that the modeller is able to regard an observable with a familiar and generic value - e.g. a point or line - as the counterpart of an observable in the referent, such as a corner or a wall of a room.

In generalising this principle to construals within the Cadence framework, it is helpful to revisit the discussion in Section 3.2. As explained there, the graph Γ differs from a definitive script in that, when viewed in snapshot, it admits richer forms of interpretation by inspection. (Compare the way in which a graph of the dependencies amongst the observables that feature in a definitive script - such as might be generated using the Dependency Modelling Tool [30] - gives insight into the interpretations of the observables.) Whilst the observables in a definitive script have a standard visualisation that can be routinely generated and automatically maintained by built-in agents, the precise nature of the perceptualisation of observables in Γ is intentionally much more a matter for the modeller. The modeller is responsible for crafting the internal semantic relation: for instance, laying out the nodes of Γ so as to assist comprehension by inspection, and confirming that changes to the incidences in Γ affect its perceptualisation in a predictable way (cf. the way in which data is organised in the rows and columns of a spreadsheet).

Similar principles apply to the moulding of the external semantic relation that conjoins the construal to its referent in the modeller's experience. Subject to incorporating standard observables (such as booleans, integers, operators and observables that reflect the state of the mouse or the GUI etc) that relate to the underlying machine, the way in which the structure of Γ is interpreted is also for the modeller to determine - in some instances, a node α in a relation of the form φ(α,*) = * might be interpreted as referring to a *context*, in another to an *object*, and in a third to a *function* for example. In engineering the conjunction between experience of the construal and experience of the referent, the key principle used in MWDS still applies: the modeller ensures that the dependencies that are disclosed in the construal

when some element of the structure of Γ is redefined are matched to the dependencies disclosed in the referent. But over and above the spreadsheet-like correspondence between values of observables that are linked by dependencies, the modeller is also able to observe whether expectations about relationships that can be read from a snapshot of state using "look-up" are confirmed, and to observe whether there are counterparts in the construal of structural and dynamical relations in the referent.

Where modelling state-as-experienced is concerned, a remarkable quality of the graph representation afforded by Γ is that it does not attach a specific absolute characterisation to an observable. That is to say, the observable β is being viewed as 'resembling a place' when we consider an edge emanating from it in Γ, as 'resembling an discriminator' when we consider it as the label of an edge; as 'resembling a value' when we consider it as the endpoint of a directed edge. This addresses a fundamental problem that besets the use of definitive scripts viz. that an *identifier* on the left-hand side of a definition is categorically quite distinct in character from a *value* on the right-hand side of a definition. It also deals with a critical problem that arises in making a satisfactory metaphorical representation of what we experience: the fact that one-and-the-same-thing can be viewed 'simultaneously' in a multitude of different ways. For instance, at this moment I can conceive the "number 12 bus" as an abstract service independent of any specific instance, as a means to travel from my home to the university, or as a physical entity that has just now stopped outside my window. To appreciate the full subtlety of the semantics that contextualised interpretation of the graph affords, it is helpful to consider yet other modes of observation, such as 'all the edges in the graph that are labelled by β', 'all the labels of edges emanating from α' and 'all the endpoints of edges emanating from α' etc. Viewing the graph Γ from many perspectives in this fashion makes it possible to lay the foundation for expressing richer types of conjunctive relation than definitive scripts alone can support. This is in keeping with the plurality of possible interpretations that co-exist in immediate experience, as vividly conveyed by James's 'dried human heads with which the Dyaks of Borneo deck their lodges' metaphor cited in Section 2.

Through modelling state-as-experienced with reference to the *structure* of the graph Γ, Cadence promotes a holistic perspective on experience. This is best appreciated by looking more critically at the way in which - and the extent to which - state-as-experienced is expressed in definitive scripts. Science and conventional programming encourage us to think of state in predominantly quantitative terms. Though we may experience the amount of water in a beaker or the distance between two objects as a relation between terms, a characteristic move in capturing this experience is to attach a numerical value to terms ("the volume of water" / "the length of the line between the two nearest points on the objects"). The use of definitive scripts, based as it is on the idea of underlying algebras of values, is strongly influenced by this approach to representing state. Pope's key insight (which has guided the development of Cadence) is that modelling with definitive scripts relies upon principles that in some respects represent a shift away from the 'pure' experiential perspective that Figure 1 suggests.

To elaborate, as James's account of "the world of pure experience" makes clear, concepts such as "integers" and "arithmetic operations" must be construed as themselves sophisticated products of experience. When we see the water in a beaker, we do not directly experience the 0.56 litres that we might attach to its volume in a definitive script. The very process of 'setting up the observational framework' within which we can determine what a "litre" is and how a quantity of liquid can be measured is itself contingent on a modelling activity that can be conceived within the framework of Figure 1, but is conceptually prior to the modelling of

this water in *this beaker* using a definitive script *as-of-now*. Of course, the fact that quantification of a liquid is observationally and experientially extremely subtle (consider, for instance, how difficult it may be to measure the volume of a liquid that is in motion - like the water in the hull of a leaking boat - or subject to evaporation - like liquid nitrogen) does not mean that we cannot become adept at estimating it instantly (as in "how much milk is left in this 1 litre bottle?") or else able to treat it as a pragmatically useful "term in experience" despite being unable to specify its value precisely as-of-now (as in "there's enough milk for a bowl of cereal"). But beyond doubt, the use of definitive scripts appeals to an implicit process of identification and calibration that assumes familiarity with standard construals that are associated with a variety of underlying algebras. In this connection, construal of standard algebraic concepts has two aspects: establishing a repertoire of experiences that both inform the notion of an abstract value (such as "the number three") and creating quasi-canonical concrete embodiments of this value (e.g. a set of three objects, a triangular configuration, a point designating 3 on the real-line, a 3-dimensional space etc). In EM based on "modelling with definitive scripts", the choice of embodiment is guided by the kind of conjunction that is appropriate from an experiential perspective.

From a Jamesian semantic perspective, the introduction of observables with canonical values in making the EM construal is incidental to the goal of promoting conjunctions in the modeller's experience. This emphasis is quite often echoed in practical EM applications, where the modeller's real interest is not in the precise values (as in "this student had an overall average mark of 69.14583..%") but in the relations associated with these values (as in "this student very nearly achieved a first-class overall average"). On this account, putting the emphasis in EM on associating identifiers with values can be limiting, and potentially obstructive. In the Cadence environment, the structure of the graph Γ is a metaphor for the relations between terms in experience - such relations are not being mediated via concepts and constructions based on preconceived underlying algebras. This reflects the influence of modelling practices conceived in the same spirit as EM within the object-oriented tradition, especially those that favour a prototype rather than a class-based approach to object-orientation. In these approaches, the aspiration is to devise representations that are engineered for the specific application in the light of the modeller's emerging experience of the domain - these representations are then not first and foremost suggested by generic abstract structures like quasi-algebraic data types but are directly derived from observation of the domain. This potentially has the effect of opening up applications to modellers who lack the level of mathematical or computing expertise needed to manipulate algebraic representations etc, and gives further motivation for promoting modelling based on concrete engagement with the domain. It is also relevant to giving an account of how an artist or musician might create a meaningful artefact, and to the goal of making modelling accessible to less technically accomplished users of computing technology.

In relating the alternative metaphors for state-as-experienced afforded by a definitive script and by the graph Γ, it is helpful to reflect more deeply on the ways in which experience can be viewed. As discussed in section 3.1, Cadence accommodates a dynamic perspective on experience in which terms change as in the modeller's stream of consciousness. Dynamical observables in Cadence can be conceived as processes ("the next status of α is related in the following way to the previous status of α") and as events ("condition A pertained in the previous moment, and condition B now pertains"). They also allow causal chains of stimulus-response activity to be framed ("when α changes now, β will change in the following way in the next moment"). A significant feature of the "willbe" definition of a dynamical observable α is that it is to be used in conjunction with assignment of a current value to α. In effect, a

'willbe' definition of an observable expresses the "vector of change" to which the observable is currently subject (cf. the velocity of a particle, the derivate of a function at a point), whereas the "equals" definition assigns a current value (cf. the position of a particle, the value of a function at a point). The assignment of a current value is independent of the vector of change.

When EM is used as a means of developing a system, there is a natural progression towards a context for interaction and interpretation in which human and computer agency are being conceived as akin to processes and programs. Migrating to this process-oriented perspective is an essential element in the automation of human activities that has been a characteristic theme throughout the history of computing. As the discussion of how EM can be used for animation via LSD and the ADM shows [11], there is a key role for analysing and implementing agent behaviours on the basis of stimulus-response reactions (cf. also the LSD-engine [1] which seeks to implement systems on the basis of an LSD account by framing stimulus-response mechanisms subject to regimes for scheduling and prioritising, and the animation system based on similar principles that has been developed by Denis Kravtsov in [21]). Dynamical relations in Cadence are an alternative means to attack a similar objective. Indeed, earlier forms of Cadence (in which "=" and ":=" were the only forms of definition) could be regarded as requiring all system behaviours to be reduced to stimulus-response reactions in this way (cf. the reference above to causal chains of the form "when α changes now, β will change in the following way in the next moment").

A core theme of EM has been that - no matter what underlying representation for programming the computer is adopted - such engineering of system-like behaviours has to be approached from an experiential perspective in general. Specifically, it is desirable - and essential unless the exercise is simply a routine design - to model the perceptions (the projected "experience") of all the agents associated with the system. This involves adopting an alternative perspective in which processes acting over time are treated as constructions. It is primarily modelling of this nature that has been effected by means of definitive scripts.

To clarify the issues involved in adopting this perspective, it is helpful to revisit the Jamesian concern for 'what is given in experience'. As I look out of my study window as-of-now I see a swift following a trajectory through the sky. Beyond question, the swift is in motion, and moves in a way that is characteristic of this particular bird. Though I cannot predict its movements precisely nor identify it as a specific swift, I have enough past experience of observing swifts to be able to recognise its flight as characterising it. In making a construal of my present experience, it makes good sense to venture to describe the movement of a swift via dynamical relations that reflect its position, velocity and the changing configuration of its wings etc.

There is a significant distinction between terms and relations like primitive observables, structural and dynamical relations that are constructed from past experience but can be apprehended as-of-now and terms and relations that involve some observation of change of a kind that is unforeseen, exploratory or experimental in character. The paragraph I am currently writing is different in nature from the flight of the swift. It is being written incrementally through crafting sequences of words in a way that is to some degree blind and responding to the conjunctions that they create. I am motivated by the idea that I shall write what I have never written before. The paragraph has no autonomous capacity to change in a predictable way - it must be constructed through my initiating change.

Writing a paragraph is just one of many activities that entails turning attention in our experience towards the contemplation and exploration of state. Supporting such activities means engineering a context and contriving a mode of interaction that enables us - for the present moment - to discount the passing of time and the inexorable attendant change. Other examples include performing scientific experiments within an environment that is sufficiently stable to enable the impact of the experimenter's interventions to be registered and interpreted, and the traditional use of a spreadsheet to study the current state of affairs in a business or education setting. In such activities, the focus shifts from the dynamic mechanisms that establish and maintain relationships to the ways in which particular agents perceive relations. Progress in writing a paragraph is measured not with reference to time elapsed or characters typed, but to thoughts to some degree having been articulated. The scientist attributes a reaction to a specific intervention even though this may not occur instantaneously. The spreadsheet modeller does not register a state in which one cell has been updated and a dependent cell has yet to be updated. In effect, each agent has its own perception of how changes synchronise and which are indivisibly connected.

In EM, support for creative activities in which the aspiration is to establish conjunctive relations through exploratory interaction is enabled by focusing on dependency relations. Unlike dynamical and structural relations that can be experienced as-of-now, a dependency relation is a latent relation that is registered only as a result of performing an action. Dependency relations have a highly significant role in the apprehension of all kinds of relations encountered in experience. For instance, changing the value of a component of a structure indivisibly affects the structure itself. Formulating a processual description of a dynamical observable entails identifying how its current value depends on its previous values. When attempting to capture structural and dynamical relations using definitive scripts, the underlying idea has been to exploit the constructive nature of these relations, and to trace them to their roots in agent experience. This has contributed to the formidable difficulties of maintaining conceptual control over EM construals. Though plurality can to some degree be achieved through superimposing many different agent perspectives this typically leads to the introduction of a whole cluster of observables associated with a term in experience. In scripts there is no satisfactory way in which to express the fact that the same observable may be viewed in many ways: cf. a snapshot of α in motion, an animated view of α in motion, α with its rich internal structure, α in close proximity with β etc. Building the construal thus entails a plethora of agents, and has insufficient support from the underlying representation.

Where MWDS in EDEN exposes the way in the semantic relation is shaped by external agency, Cadence highlights the role of agency that is closer to the machine. By obliging the modeller to base construals on established underlying algebras and their standard experiential counterparts, MWDS hides agency. In contrast, Cadence incorporates mechanisms that can be used to specify richer custom-built data types and to establish more flexible structural relations, and to enable different modes of implementation (e.g. exploiting special features of the architecture, such as concurrency, or linking to external applications) that deliver better performance and control and support richer forms of visualisation. A key issue in Cadence is deciding how to realise the internal semantic relation by means of which the graph Γ acts as a metaphor for state-as-experienced. For instance, as discussed in section 3.2, the illusion that all the integers are realised within Γ can be sustained by a process of virtualisation. It is possible to imagine that all strings could be treated similarly, though this is not the case in any current implementation. In implementing Cadence, there is a tension between conceiving Γ as a 'pure' structure in which all the content lies in the incidences between nodes, and

matching this to the physical characteristics of the computing environment and the disposition of the modeller. One important motivation for foregrounding mathematical concepts that afford means to reference and manipulate structure over strings is the desire to steer clear of the pitfalls of linguistic representations (cf. the rogues' gallery of syntactic devices that EDEN exploits in order to do overcome the expressive limitations of plain definitions - execute(), backticks, virtual agents etc). This is consistent with the aspiration to develop an environment for Cadence that is essentially based on direct manipulation, and in line with a Jamesian outlook on the relation between experience and language: "The truth is that neither elements of fact nor meanings of words are separable as our words are" [19].

In making sense of such use of Cadence in capturing state-as-experienced, it is once again helpful to consider the parallel between Cadence and the synthesiser. The sound that is generated by a synthesiser is of its essence a temporal phenomenon - it involves an oscillation that is registered by the listener in a very different way from a musical note. Whereas traditional instrumental music is typically conceptualised in terms of notes sounding at specific standard frequencies disposed in some rhythmic pattern comprising discrete beats, the musical effects that can be achieved using a synthesiser are not most appropriately conceptualised in this way. Like the synthesiser, Cadence engages with the construction of dynamical effects at such a primitive level that the distinction between state-changing activities that will not be registered state-by-state (cf. "the pressure variations that generate a sound at a particular pitch" / "the intermediate states in the Cadence model of the 8-bit adder circuit") and the state-changing activities that are registered moment-by-moment ("a discrete movement from one note to the next" / "the consequences of changing a boolean observable from **true** to **false**") becomes blurred.

Though it may seem that environments that give access to a more primitive model of state necessarily afford greater expressivity, this idea needs qualification. From the perspective of the performer, a synthesiser is in no way more satisfying than a traditional instrument without the kind of analogue interface that can mediate state-changes appropriately. What 'appropriately' signifies here has to do with individual physical characteristics and skills but is also deeply culturally embedded (cf. the pleasure of clashing a pair of cymbals together, and of seeing the cymbals clashed together, and contrast listening to the sound of a flute on a pipe organ with listening and watching the movements of a flautist). "Manipulating pure sound flexibly" differs from "composing music" in somewhat the same way that "having perfect pitch" differs from "appreciating music". In experiential terms, what matters is what conjunctions can be conjured, and in this respect the powerful expressive qualities of culturally developed and mediated sounds, such as words in a familiar language, are self-evident. On that basis, the qualities of Cadence as a medium for EM need to be assessed with reference to the ways in which it can be integrated into a richer semantic context than a "pure Cadence" environment affords.

An interesting aspect of the above discussion is the role of elements of experience that we deem "concrete" and "abstract". In modelling with definitive scripts, observable names and the types and operations associated with underlying algebras are in some sense appropriately viewed as "abstractions" that impede intelligibility. Certainly, both managing observable names and establishing links between the diverse representations of values in different underlying algebras are a major source of complication in EDEN. Cadence seeks to ameliorate these problems by foregrounding the explicit graph Γ and giving the modeller discretion over the extent to which they attach names to nodes and introduce classes in preference to prototypes. In these respects, the aspiration is towards eliminating observable

names and scripts of definitions in favour of direct inspection and manipulation of Γ via a GUI. From an experiential perspective, it is not clear that these steps towards perceived 'concretisation' are well-conceived. In James's account of experience, the primary focus is not upon the traditional distinction between concrete and abstract, but between what is and is not experienced. For instance, James asserts: "Everything real must be experienceable somewhere, and every kind of thing experienced must somewhere be real" [18:159-60] and contends that what is traditionally perceived as abstracted from concrete experience can itself participate in conjunctions within that experience: "Returning into the stream of sensible presentation, nouns and adjectives, and *thats* and abstract *whats*, grow confluent again, and the word 'is' names all these experiences of conjunction".

Concluding remarks

Cadence gives new insight into design questions concerning the form that tools for EM may take in the future.

The expressive limitations of definitive scripts and of the EDEN interpreter are grounds for adopting Cadence as an alternative and superior platform for modelling and programming, but whether it would be a good idea to dispense entirely with the explicit use of definitive notations and scripts is open to question. Definitive scripts are an interface with significant qualities where making interaction meaningful is concerned - of course limited by their symbolic nature when it comes to matters of comprehension and management on a large scale, but well-suited to what humans can take in - or at least grapple with - in one conceptual chunk of experience. It is possible to imagine that - even in the large scale symbolically intractable models that may be constructed using Cadence - definitive scripts could play a role in giving access to the conceptually manageable portions of the entire state. For instance, in large models constructed by cloning from prototypical objects, a definitive script might be especially useful in specifying the objects to be cloned. And even though, in this scenario, it makes most sense to imagine that the underlying implementation framework is Cadence-based, there would then be some need for blending definitive scripts ("for supporting human interaction and interpretation in construal") with object-based structures ("for enabling effective large-scale implementation") and dynamical observables ("for modelling an underlying machine architecture and other user-conceived processes").

The ideas in this report have been informed by a relatively small number of practical studies in modelling with Cadence. Several different experimental prototypes that integrate Cadence with MWDS in EDEN have been constructed. The first of these ("Cadence-in-Eden") introduced a new definitive notation ("%dasm") to allow Cadence-style structural definitions to be framed (cf. [22], p.169), thus giving partial support for imposing structure on EDEN observables. The most recent hybrid tool treats EDEN as a module of Cadence (cf. [22], p.184), and enables specified Cadence observables that can be deemed to have traditional scalar values either to be observed or to be redefined by EDEN. Though no hybrid tool that has been developed has as much integrity as Cadence and EDEN, experiments of this nature highlight some ways in which MWDS as so far practised in EM is unsatisfactory:

- MWDS exploits underlying algebras that admit 'objective' interpretations and perceptualisations and this is not well-oriented towards adopting a personal subjective stance in the preliminary stages of making construals;
- the flat data structures in EDEN mean that MWDS with EDEN needlessly obstructs communication between different definitive notations;

- in the absence of dynamical observables, MWDS is not well-suited to dealing simply, directly and efficiently with interfaces to physical devices.

Exploiting the incidence graph Γ rather than a definitive script as the primary metaphor for state-as-experienced as in Cadence promises to address all these issues in principle. But whereas MWDS have a precdent in spreadsheets, there is perhaps no such obvious widely adopted precedent for the grounded semantic use of incidence structures in the style of Cadence.

In considering how Cadence should best be developed, it may be helpful to reflect on the many ways in which a synthesiser might be used as the basis for musical composition. This question is after all central to the issue of making musical artefacts that are experienced as meaningful. If musical culture is an appropriate guide, then it is not be expected that 'raw' use of Cadence will readily yield semantically rich associations. We should rather expect that conjunctive relationships in experience can be conjured in many diferent genres, but that each will have some culturally-mediated normative practices (cf. the use of counterpoint, motif's, tonality etc). Where modelling is concerned, the closest analogues to different musical styles of composition may be found in the different modes of representation that are emerging in the object-oriented practices, though these seem as yet immature in relation to the focus on state-as-experienced that is characteristic of EM. By drawing on the object-oriented tradition, Cadence promises to do more justice than MWDS to the extraordinary semantic potential for 'conjunctive relations' that is self-evident in our wider experience.

References

1. Adzhiev, V., Beynon, M. and Rykhlinski, A.. (1999). Empirical modelling of multi-agent systems with inherent concurrency. Technical Report. Japan: University of Aizu.
2. Allderidge, J.A., Beynon, W.M., Cartwright, R.I. and Yung, Y.P. (1998). Enabling Technologies for Empirical Modelling in Graphics. Proc. Eurographics UK, 16th Annual Conference, 199-213.
3. Beynon, M.(2011). Modelling with experience: construal and construction for software. Chapter 4 in *Ways of Thinking, Ways of Seeing* (ed. Chris Bissell and Chris Dillon), Springer-Verlag, 2011, to appear
4. Beynon, M. (2011). From formalism to experience: a Jamesian perspective on music, computing and consciousness. Chapter 9 in Music and Consciousness: Philosophical, Psychological, and Cultural Perspectives (ed. David and Eric Clarke), OUP, July 28th 2011, ISBN: 9780199553792, 157-178.
5. Beynon, W.M. (2005). Radical Empiricism, Empirical Modelling and the nature of knowing. In (ed. Itiel E Dror) *Cognitive Technologies and the Pragmatics of Cognition: Special Issue of Pragmatics and Cognition*. 13:3, December 2005, 615-646.
6. Beynon, W.M.(1999). Empirical Modelling and the Foundations of Artificial Intelligence. *Computation for Metaphors, Analogy and Agents*. Lecture Notes in Artificial Intelligence 1562, Springer, 322-364
7. Beynon, W.M. (1986). ARCA - A Notation for Displaying and Manipulating COmbinatorial Diagrams. CS-RR-078, Computer Science, Warwick University.
8. Beynon, W.M. (1983). A Definition of the ARCA Notation. CS-RR-054, Computer Science, Warwick University.

9. Beynon, M., Boyatt, R., Chan, Z.E. (2008). Intuition in Software Development Revisited. In *Proceedings of 20th Annual Psychology of Programming Interest Group Conference*, Lancaster University, UK, September 2008
10. Beynon, W.M., Boyatt, R.C., Russ, S.B. (2006) Rethinking Programming. In *Proceedings IEEE Third International Conference on Information Technology: New Generations* (ITNG 2006), April 10-12, 2006, Las Vegas, Nevada, USA 2006, 149-154.
11. Beynon, W.M., Norris, M.T., Orr. R.A. and Slade, M.A. (1990). Definitive specification of concurrent systems. Proc UKIT'90, IEE Conference Publications 316, 52-57.
12. Beynon, W.M., Rungrattanaubol, J., Sinclair, J. (2000). Formal Specification from an Observation-Oriented Perspective. *Journal of Universal Computer Science*. Vol. 6 (4), 407-421.
13. Beynon, W.M., Ward, A., Maad, S., Wong, A., Rasmequan, S. and Russ, S. (2000). The Temposcope: a Computer Instrument for the Idealist Timetabler. In *Proc. of the 3rd international conference on the practice and Theory of Automated Timetabling*, Konstanz, Germany, August 16-18. 153-175
14. Fernandes, K., Raja, V., Keast, J., Beynon, W.M., Chan, P.S. and Joy, M. (2002). Business and IT Perspectives on AMORE: a Methodology for Object-Orientation in Re-engineering Enterprises. In Systems Engineering for Business Process Change: New Directions, (ed P. Henderson) Springer-Verlag 2002, ISBN 1-85233-399-5, 274-297.
15. Gooding, D. (1990). *Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment*. Dordrecht: Kluwer.
16. Gooding, D. (2007). Some Historical Encouragement for TTC: Alchemy, the Calculus and Electromagnetism. At url: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/thinkcomp07/gooding2.pdf (Accessed 22/9/2011)
17. Harel, D. and Marelly, T. (2003). *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag.
18. James, W. (1912/1996). *Essays in Radical Empiricism*. (Reprinted from the original 1912 edition by Longmans, Green and Co., New York) London: Bison Books.
19. James, W. (1911/1996). *Some Problems of Philosophy: A Beginning of an Introduction to Philosophy*. University of Nebraska Press (April 1, 1996)
20. Keer, D., Russ, S., Beynon, M. (2010). Computing for construal: an exploratory study of desert ant navigation. In *Procedia Computer Science*, Volume 1, Issue 1, May 2010, 2207-2216
21. Kravtsov, D. (2011). *Hybrid Modelling of Time-Variant Heterogeneous Objects*. PhD Thesis. NCCA, School of Media Studies. Bournemouth University.
22. Pope, N. (forthcoming 2011). Supporting the Migration from Construal to Program: Rethinking Software Development. Draft PhD Thesis (submitted), Computer Science, University of Warwick, Coventry, UK.
23. Pope, N. and Beynon, M. (2010). Empirical Modelling as an unconventional approach to software development. In *Proc. SPLASH 2010 Workshop on Flexible Modeling Tools*, Reno/Tahoe Nevada, USA, October 2010
24. Rungrattanaubol, J. (2002). A treatise on modelling with definitive scripts. PhD thesis, Computer Science, University of Warwick.
25. Smith, B.C. (1996). *The Origin of Objects.* The MIT Press.
26. Wirsing, M. (1990): Algebraic Specification. In J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publisherrs.

27. The Empirical Modelling website. Online at http://www.dcs.warwick.ac.uk/modelling (Accessed 22/9/2011)
28. The Empirical Modelling archive. Online at http://empublic.dcs.warwick.ac.uk/projects (Accessed 22/9/2011)
29. Rungrattanaubol, J. (2001). An EM model of heapsorting. heapsortextendRunbol2001 in the EM archive. (Accessed 22/9/2011)
30. Wong, K T A. (2003). The Dependency Modelling Tool. dmtWong2003 in the EM archive. (Accessed 22/9/2011)
31. Harfield, A.J. (2007). Colour Sudoku. Online at url: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/sudoku/ (Accessed 22/9/2011)
32. The Sudoku Experience. Online at url: http://www.dcs.warwick.ac.uk/~wmb/sudokuExperience/workshops/ (Accessed 22/9/2011)
33. The Warwick Games Design library. Online at http://www.warwickgamedesign.co.uk/projects/wgdlib (Accessed 22/9/2011)