Liberating the Computer Arts

Meurig Beynon

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

Abstract

Computer-based technology has had a major influence over business, politics and education worldwide. The practical consequences of ubiquitous computing are easy to see, but there has also been a more subtle impact of the Information Age upon the way we view the world. The power of the computer to transform our interactions with our environment and each other through the digitisation and symbolic representation of observables is patent. These developments have enhanced the intellectual influence of a theoretical framework endorsed by classical computer science, yet – at the same time – they disguise from the user and expose to the designer the limitations of that very framework itself. In the process, received computer science and its associated technologies have helped to legitimise and promote an incomplete view of science, and detracted from the real and potential role of the arts and humanities in shaping our lives.

This paper examines these issues with reference to the search for an alternative software culture that can better serve the agenda of the computer arts. It attributes the difficulties in establishing such a culture not merely to commercial and political vested interests, but to the problem of integrating typical computer use with human sense-making activities rooted in engagement with the world, the acquisition of experience and reflection upon that experience. Addressing this problem involves a reappraisal of the philosophical roots of classical computer science that is motivated in this context by contrasting the paradigms for the representation of experience that might be seen as distinguishing the sciences from the arts. The paper concludes with an introduction to a philosophical stance and practical approach to computing, originating from research in the Empirical Modelling research group at the University of Warwick [EM01], that is aimed at liberating the computer arts.

1. Introduction

The enormous and accelerating advances in computer technology over the last few decades have had a major practical and intellectual impact upon our world. The role of classical computer science in this is paradoxical. Within the sphere of computing it seems that practice is outrunning theory. Where once Codd's mathematical relational model of data accounted for the major part of data processing applications, for instance, there is now a vast body of practical activity concerned with expert systems, interfaces, visualisation, object-oriented modelling, new media and architectures for which the mathematical foundations are far less assured. An emergent trend amidst the

technological developments is the aspiration to use the computer as an artefact rather than a numeric and symbolic processor, and to explore the associated activities that might more aptly be described as computer *arts* rather than computer *science*. Though nothing can detract from the profound insights that lie behind the classical theory of computation, in logic and the theory of algorithms, their relevance to computing practice is far less conspicuous now than it was twenty years ago. There may even be a danger that in the future computer science will suffer the same fate as economics, where mathematical models have become divorced from practice and so narrowly applicable as to be perceived as "the theory of nothing".

Intellectually, the picture is very different. Pervasive digitisation is apparently reducing everything to a symbolic mathematical foundation. Computation, information and rules are the dominant metaphors we use in thinking about the brain, business practice and education, and the human genome. As the discipline that interfaces abstract prescriptions with real engineering and human activities most prominently and routinely, computer science contributes to the perception of the universe as fundamentally rational and explicable, if we but knew the program. The possibility that in due course we ourselves will be recognised as complex machines assures computer science of a central place in "the theory of everything".

The future of software will be a major factor in determining the influence of computer science in the 21st century. The evolution of the software culture will in turn depend upon how far it adopts the ethos of computer science or that of the computer arts. This paper explores this issue by appraising the impact of computer science and technology on how we view those aspects of experience that are most characteristic of the arts. It argues for an alternative philosophical stance on the representation of experience via computer-based technology, and proposes complementary practical principles for computing.

2. Issues for the software culture

2.1. The pop-software culture

The culture that surrounds the software industry has a critical influence over the ways in which computers are used, and over all the human activities associated with their use. This influence is reflected in several contrasting emphases that can be placed on the development of software: on software as product vs. software as resource, on software as object vs. software as place, on software as polished vs. software as unfinished.

- software as product vs. software as resource

Whether software is conceived as freeware or to be sold for profit, it is typically developed in a manner that reflects a commercial outlook. The software is seen as a product to fulfil specific functions, and perhaps (as in a database, information resource or expert system) to incorporate knowledge. The recommended ways to develop software reflect this view of software: the primary emphasis of such a development is upon

identifying users and user needs. The presumption behind this approach is that the software developer is like a traditional manufacturer: they supply a product to be used in a specific fashion by someone who precisely knows and understands its intended use. In this context, the relationship between the supplier and the user is strongly hierarchical – the power of the user is limited to possibly making a choice of product, having the right to register a complaint, and being able to request additional guidance or (over a longer term, and where appropriate) additional functionality. This hierarchical relation is indeed the main rationale for a commercial stance: the supplier protects their expertise, and their information from open access, and discloses it for profit.

The concept of software-as-product is often at odds with its status as a resource for the intellect. For instance, the precise function that an operating system serves for its users is difficult to circumscribe having regard to all the software the user might wish to run or develop. A useful analogy can be made with education, where the limitations of treating subjects as products to be consumed or applied according to preconceived rules are well-recognised. The aspiration in software, as in education, is for a relationship that is less hierarchical in character, where there is room for negotiation and personal expression on the part of the recipient. In this view, software is – as is effective education – a resource for personal exploration, thought and social development.

- software as object vs. software as place

The software-as-object perspective of the consumer complements the software-as-product perspective of the producer. An object is something that we typically own rather than share, use in certain ways, and value for specific features. The production of software objects that fulfil specific roles effectively is the primary function of the software industry. The way in which software products are developed and marketed sometimes reflects the consumer's interest in objects as status symbols, to be revered for their size or complexity.

In contrast, a new piece of software can resemble a new place. The metaphors of acquisition, ownership and use are not so appropriate to the software-as-place perspective. A place is somewhere to which we come, that we can work in, where we can meet and share with others, can make our own without needing to exclude others. Unlike an object, a well-designed place is intended to accommodate people, to give them scope to express themselves, and to be creative. The creation of a place is better aligned to the aim of an artist than the creation of an object.

- software as polished vs software as unfinished

Notwithstanding the imperfections of many present software products, they are often guided by an aspiration to perfect software and make it complete. In academic courses in software development, great emphasis is typically placed upon getting the initial specification right and delivering error-free code to this specification. The effect of polishing software can be to distance the user. Whether or not it is the intention of the software developer, the addition of new features, the introduction of optimizations, and

the simplification of interfaces make it more difficult for the user to understand the conceptions and mechanisms that lie behind the software as experienced through use and accessed in source code. Even where there is good will and no commercial obstacle to cooperation between the developer and the user of a piece of software, it is typically difficult to create a polished piece of software without operating to the same hierarchical producer-consumer pattern: "let me know what you think is wrong, and if I agree with you I will fix it, and return the enhanced version to you".

The culture of DIY software, where trade is in unfinished software that is to be developed by the recipient or user is not widely represented commercially, but is practiced within corporate software development environments and in the open source community. The great merit of software-as-unfinished is that admits a level of engagement that is missing from sophisticated polished software products. Beyond a certain point, as the latest versions of Microsoft Word illustrate, the attempt to supply sophisticated functionality and agency to software in line with the anticipated needs of the user comes into conflict with their desire to make independent judgements in ways that reflect the current context and cannot be second-guessed by a rule-based software agent.

2.2. Implications of the software culture

The view we take of software has significant implications for its impact upon our lives. In conceiving software as polished-object-product to be sold for profit (or "pop-software"), a producer is indirectly shaping the work practices and environment of its users. By its nature, such a product is best suited for use in conjunction with other products specifically co-manufactured to fulfil this function. There are serious obstacles, both technical and commercial, that obstruct the integration of products from different producers. This promotes a typical pattern of usage that is exceptional amongst manufactured products: to make effective use of a software product requires an entire software package. Such is the resulting difficulty of establishing widespread acceptance of alternative products that diversity is hard to sustain.

In this context, the hierarchical relationship between producer and user of the software-as-polished-object-product has its most potent influence. The intellectual power in this relationship is with the producer, who dictates key characteristics of the pattern of use to the customer. There are enormous fortunes to be made by successful producers, and major pressures upon all to subscribe to their successful model of use. The social and cultural implications of dominant software packages are even more significant. Pervasive technology shapes human and social practice, promoting a culture in which conformity to standards that can marginalize the individual – and to rule-based norms imposed by managers rather than practitioners – are viewed with approval.

The danger of exaggerating these concerns, and of making simplistic assumptions about the social and commercial mechanisms that are involved, is acknowledged. In relation to other technologies, the computer is nevertheless exceptional in the degree to which it impacts upon our mental life, and upon activities that involve creativity, self-expression

and communication. The application of computer technologies influences patterns of social interaction, shapes the dynamics of international business and finance, and has a major impact on the way in which ideas are developed, presented and distributed. This gives software-as-product producers unprecedented power to affect society through its interface with the mind. This amplifies Wiener's concern about the influence of mass production [Wie52]: that products would be created "not so that they will give great satisfaction to some who have an understanding, but that they will offend as few as possible". Computer-based technologies are already demonstrating some of their potential to impose tyrannical workloads upon those in professions in which thought and social interaction are at a premium. Addressing these issues will require more than an emphasis on ease-of-use, and something complementary to Nick Donofrio's call for computer-based technologies for the mass delivery of "what we want, where we want, when we want it" [Don01].

The above discussion is not intended to deny the pop-software culture its central place. There is a real and legitimate need for polished objects as software products, and they will inevitably be of greater interest and direct benefit to the majority than unfinished alternative software. That such products should dominate the market is one thing, but that their ethos should dominate our thinking is quite another. From the perspective of this paper, the rationale for an alternative software culture is much more than finding a more congenial way to deliver the same kind of software products. In championing the concept of software-as-resource, software-as-place and software-as-unfinished, the objective is not only to influence the way that software is created, but also to use computer technology to enfranchise the individual in ways that are distinctive of human creativity in all spheres, and of arts and the humanities in particular.

3. Foundations for the alternative software culture

The development of unfinished software as a place of resource raises many issues. It might be argued that the principal features of current software technology have been shaped primarily by commercial agendas, and the self-interest of multinational software producers in particular. Though these factors have surely played their part, the thesis of this paper is that the essential problems of the software culture go much deeper, and are concerned with challenges at the heart of computer science such as Brooks explores at length in [Bro87]. It is not at all apparent for instance that orthodox software development is well suited for corporate work, that there is any alternative to imposed standards as a basis for cooperation, or that the user can be effectively engaged in customising or extending software. In addressing these concerns, it proves necessary to take a radical position regarding fundamental preconceptions of computer science, such as the ways in which it is appropriate the use of the computer as a representational medium. To frame these issues, it is helpful to compare the two contrasting ways in which representation is traditionally approached in the sciences and in the arts.

3.1. Representation in the sciences and the arts

For the scientist, the most appropriate representations are general and the abstract. Science is centrally preoccupied with rationalising occurrences and classifying instances of phenomena. The identification of a phenomenon and the development of its representation are closely linked in as much as what is explicitly observed is precisely what is to be recorded in the representation. Other observation is extraneous and irrelevant to the particular abstraction of current interest. Science deals with observations that can be objectively repeated to a high degree of precision. Its focus is upon representing abstractly what accords with authenticated experience. In summary, science practises what might be conveniently termed "The 4R's principle": placing the **r**ational in the **r**ight **r**elation to the **r**eal.

For the artist, the most appropriate representations are particular and concrete. The significance of the representation is not to be sought in a predetermined decomposition into parts and convention for interpretation such as underlies a logical specification. The work of art is a holistic creation, whose expressive power may derive from an allusion to a specific situation or other contextual association. Ambiguity, subjectivity and unpredictability are of the essence, and the work succeeds through the engagement of the observer and the degree to which the imagination of the observer is stimulated through its contemplation. Where science may be seen as focusing on establishing 'the right relation' between the rational and the real, art incites us to fantasy, to the irrational, and to confront conflict and uncertainty. Where Computer Science and Computer Arts are concerned, the distinction between these two viewpoints is most clearly seen with respect to the representation of three aspects of everyday experience: **s**ituation, **i**gnorance and **n**onsense. On this basis, representation in art may be deemed to favour "The SIN principle" (cf. Poem 1).

To elaborate upon this theme, it is useful to contrast the perspective that Computer Science and Computer Arts adopt on representing situation, ignorance and nonsense.

- The representation of Situation from a Computer Science perspective

From the Computer Science perspective, the representation of a real-world situation is invariably understood with reference to some specific objective for modelling. Though the process of constructing a representation is recognised to be difficult, the principal ingredients are uncontroversial. The modeller analyses the environment to identify the observables that are sufficient to the purpose in hand, makes a mathematical representation of the relevant observables, generally presuming them to be objectively characterised, then constructs firmware to monitor, manipulate and record these observables and their mathematical representations.

In conventional software, the emphasis in dealing with situation is to generate a pattern of user-computer behaviour that synchronises the relevant observables of the situation with the states of the human users and of the computer. This is done in such a way that all

relevant observables can be interpreted according to a pre-determined scheme. Objectivity of the observables involved is essential where they serve to mediate between human agents in the environment of use. The subject matter of interest in any situation almost invariably features in the state of the environment rather than that of the human participants.

- The representation of Situation from a Computer Arts perspective

Situation is the primary focus of concern for the artist. Though many situations may be represented in a work of art, as in a drama or a piece of music, the artist is essentially concerned with what can be apprehended about state in each moment. The state of the observer is an essential part of the situation: enriching the state of the observer is of itself the objective. The situation is represented by an artefact in a manner that may involve some conscious exercise of principles or craft, but relies primarily upon the immersion of the artist in the experience. Where Computer Arts is concerned, the emphasis is upon those aspects of computer-based technology that are experiential in nature.

- The representation of Ignorance from a Computer Science perspective

To the computer scientist, knowledge and representation are intimately associated, if not inseparable. Knowledge is perceived as prerequisite to representation – we can only represent what we know. Moreover, as the negation-as-failure principle illustrates, what we can represent is also the best indicator we have of what we know. Knowledge of how to achieve goals is similarly crucial to the representation of action. There is no program without an algorithm – anything we *can* do has to be something we know how to do. Two reflections help to disclose the subtlety of this viewpoint. We all do things that we don't know how to do all the time: *knowing how to do* is a way of referring to experiences that we can reliably revisit, which is a small part of our total experience. In this connection, Computer Science also has an issue to address in relation to systematic methodologies for software development: they are in danger of systematically delivering solutions to insoluble problems, and are also in danger of endorsing only those solutions that stem from systematic problem solution.

- The representation of Ignorance from a Computer Arts perspective

The artist acknowledges that not all knowledge can be articulated, and that every situation is essentially more than can be represented. In viewing a work of art, the observer is faced with a situation that serves no specific purpose and invites entirely uncircumscribed observation. The mature artist is aware that one experience can represent another in ways that we can reliably experience, but cannot explain. The acknowledgement of ignorance is an essential element of art: the work of art is intended to stimulate the imagination in ways that cannot be prescribed, and to evoke wonder at in what ways, how and why our imagination responds.

- The representation of Nonsense from a Computer Science perspective

In Computer Science, representations are perceived as valid and useful, or invalid and useless. In a program, representation is directed at a normal execution, and events outside this norm are singular and beyond this representation. If a failure in the synchronisation of situation with user or computer state occurs, or there is an unforeseen need to consult the environment – perhaps even to take account of observables outside the scope of normal execution – the interpretation of the entire representation is subverted. In a program there is no provision for an event not to occur according to plan (except in so far as "an event not occurring to plan" can be appropriately interpreted as "an event occurring to plan"). Fuzzy logic, or the consideration of probabilities of events can widen the compass of normal execution, but does not affect the way in which the semantic relation between a program and its environment breaks down at its boundary.

The techniques for knowledge representation accredited by Computer Science are also neutral with respect to the quality of the experience that informs them. In practice, knowledge is differentiated in our experience according to its source and degree of authenticity. The manner in which this knowledge is represented in computer programs gives us no means to take account of this.

- The representation of Nonsense from a Computer Arts perspective

Parody, absurdity, exaggeration and fantasy are central to art. The representation of nonsense relies on the subtlety of the mechanisms that create associations between one experience and another. These exploit what Turner [Tur96] characterises as 'blending' of experiences that can result in incongruous or funny associations.

Representing something 'as it is' – as in so-called 'representational art' – is one aspect of what an artist does in order to invoke a blending of explicit experiences, and nonsense derives from using this device to bring together familiar experiences in unfamiliar ways. Unlike the scientist, the artist differentiates nonsense from 'having no meaning'. All works of art potentially have meaning in as much as they are subject to blend with other experiences in mind of the observer, but the degree to which a particular blending is part of the artist's explicit intention is arbitrary. Meaning can be dependent on context, one work of art or aspect of a work of art being meaningful in relation to another as in a variation on a musical theme, a picture in a series or a cartoon in a political situation.

The appreciation of art relies upon the characteristics that experience acquires through ritual. We come to understand a work of art by familiarity through our interaction with it. We can also come to understand it through acquiring extraneous experience. There is a parallel here with the way in which we associate measures of sense and plausibility to reported experience on the basis of our own prior experience.

4. Empirical Modelling

Empirical Modelling (EM) is a new approach to modelling, and in particular *computer-based* modelling, that has been developed by the author and his collaborators over the last 15 years or so [EM01]. This section provides some orientation on the relationship between EM and the computer arts; describes and illustrates the principles of EM with a simple example; then discusses the ways in which experience gained from project work by undergraduate and graduate students at the University of Warwick relates to the agenda for an alternative software culture. The focus for this discussion will be on the extent to which EM potentially supports the kinds of representation that are favoured by computer arts.

4.1. Background and Orientation

EM provides a general framework for a wide range of activities that involve the construction of artefacts to represent situations or phenomena. These activities are encountered in a wide range of disciplines, and are not confined to a particular modelling medium. Examples of potential applications include: prototypes in engineering, construals in experimental science [Goo90], computer simulations, spreadsheets for financial modelling and works of art. In some respects, computer-based modelling is best suited to situations in which there is a quantitative mathematical basis for observation, but the aspirations and orientation of EM are for the most part better aligned to situations where there is no clear understanding or theory to guide – or prejudice – the modeller's observation of the domain. This is true, for instance, in the early stages of the identification of a scientific theory [Goo90], in complex issues of decision-support in business and finance, and in the creation of a work of art.

In elaborating the principles of EM, bearing in mind the limitations of the scientific stance on representation discussed in the previous section, there is a narrow path to tread between two uncomfortable alternatives. On the one hand, it would be absurd to propose principles that could account for the creation of works of art, and so represent "a methodology for Mozart". On the other – acknowledging that the mechanism by which one experience represents another is mysterious, but that one experience *can* represent another – it is essential to take a more constructive view than that the construction of artefacts to represent an experience defies all explanation or introspection. In the former case, our purported principles would be so strong as to preclude the computer arts perspective on experience, and in the latter, too ineffectual to offer any practical endorsement.

The example of the so-called – if inappropriately called – "scientific method" is a useful point of reference in this dilemma. Few would doubt that the activities that lead to the identification of a new scientific phenomenon are beyond abstract formal prescription. On the other hand, only certain activities are accredited as recognisable applications of the scientific method. New scientific theory has to be rooted in experimental practice that follows well-established conventions. Where artefacts or instruments are developed to

mediate a phenomenon to the observer, the authenticity of the experimenter's claims about the correspondence between observation of these artefacts and observation of the phenomenon has to be established, and verified by others.

The authenticity of the artist's experience of their work of art does not demand the universal confirmation expected of a scientific discovery, but we at least expect the artist to be one for whom their own work has some significance, whether or not we can confirm this to be so. In this respect anything is potentially a work of art, but only some of these will speak to us. A great artist no doubt has a powerful personal response to their own work, and for them – whether or not this is representational in character (that is, it refers to a specific experience) – it is evocative of rich complementary experience.

Though the observation-dependency-agent framework of EM affords no recipe for the construction of art, it is intimately connected with activities such as blending whose significance spans science and art, and may be significant in the unconscious, unarticulated activity that the artist explores in creative work. Practice in EM is similar to the way in which both the artist and the experimental scientist working in an unfamiliar domain approach the task of creating artefacts: in particular, the manner in which their grasp upon the artefact is shaped by experience, and by provisional commitments to reliable understanding and acceptance.

EM is perhaps more explicitly relevant to those aspects of art that are consciously introduced in creating or analysing a work of art. For instance, in the work of a major artist there are stylistic elements and devices that are reworked and redeployed, and in so far as these things are associated with instruments and crafts, they can be taught and objectively evaluated. Within a genre, the analysis of a work of art is then typically cast in terms of specific kinds of observables, dependencies and agency, as in the harmonic patterns, instrumentation, pitch, tempo, timbre, rhythmic patterns etc associated with a piece of music.

To sum up, EM is not a recipe for creativity, but provides a setting in which to "account for" its products after the fact. EM principles are relevant to enabling the computer to be used in ways that would meet the needs of the artist. As will be apparent, the practical application of EM thus far is at an early stage of development in relation to the computer arts agenda. Nonetheless, we shall show that in EM – in the same way that, in the process of creation, the artist may make small changes that carry deep meaning – simple interaction with the computer can correspond to rich thought processes and changes of perspective on the part of the modeller. Equally importantly – despite the evident limitations of our current tools for EM – there are some respects in which associating EM with modelling for the arts can help in clarifying the distinction between EM and computer use as traditionally conceived in computer science.

4.2. Principles and Tools of EM

The principles and tools of EM will be briefly described and illustrated with reference to a simple but extended example. This focuses on model-building activity surrounding an imaginary analogue clock.

The primary focus for representation in EM, as in art – and in contrast to computer programs that are targeted at behaviour, is on situation. In the initial situation in which we observe the modelling activity, the model consists of the outline of the clock. This is defined by a family of definitions of variables (a definitive script) in a special-purpose notation (a definitive notation) for line-drawing – donald.

```
%donald
viewport CLOCK

openshape clock

within clock {
    real sixthpi
    line eleven, ten, nine, eight, seven, six, five, …, one
    line noon
    point centre
    real radius
    circle edge

    sixthpi = 0.523599
    radius = 150.0
    eleven = rot(noon, centre, -11 * sixthpi)
    ...
}
```

The variables in this script represent observables in the clock: the rim of the face, represented by the circle **clock/edge**, its centre **clock/centre** and the divisions **eleven, ten, nine,** ... etc that indicate the hours.

The artefact that is defined in this fashion is depicted on a screen, as in Figure 1.1:

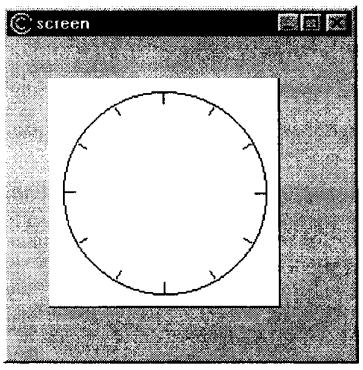

Figure 1.1: The clock face

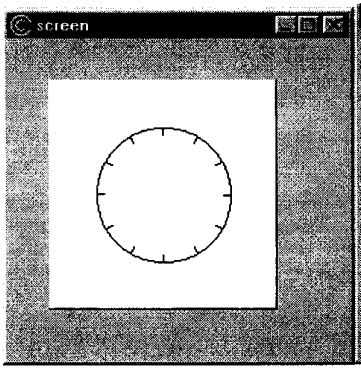

Figure 1.2: The scaled face

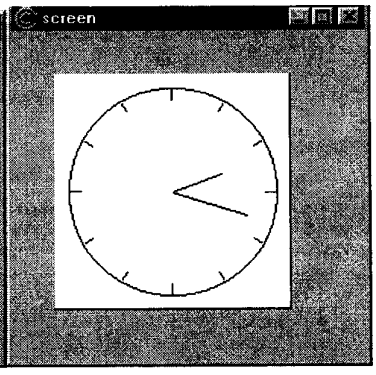

Figure 1.3: With time set

The definitions in the script establish dependencies between the variables similar to those in a spreadsheet. Interaction with the script takes place in an environment in which the values of variables are always open to redefinition. For instance, the redefinition

```
clock/radius = 100.0
```

has the effect of making the clock smaller, simultaneously changing all the positions of the divisions and the rim of the clock. The display also depends directly upon the values of variables in the script, and is simultaneously updated (see Figure 1.2).

The hands of the clock are not yet displayed, but there are already variables and definitions in the script that refer to them:

```
within clock {
  line minHand, hourHand
  real minAngle, hourAngle
  real size_minHand, size_hourHand
  int t
  size_minHand, size_hourHand = 0.75, 0.5
  minAngle = (pi div 2.0) - float (t mod 60) * (pi div 30.0)
  hourAngle = (pi div 2.0) - float (t mod 720) * (pi div 360.0)
  minHand = [centre + {size_minHand*radius @ minAngle}, centre]
  hourHand = [centre + {size_hourHand*radius @ hourAngle}, centre]
  centre = {200, 200}
  ....
}
```

They are not currently displayed because the value of the integer `t`, which represents the current time, has yet to be determined. Assigning different values to `t` readily establishes that `t` can be interpreted as the time elapsed in minutes from midnight, so that (for instance) assigning `t` to 138 sets the clock to time 2.18 (see Figure 1.3.).

The modeller's interaction with the clock script is not directed towards any particular goal or constrained by a preconceived interpretation. The modeller views interaction with the artefact from the perspectives of many different human agents, shifting perspective arbitrarily, much in the way that an artist saturates their imagination through attention to the emerging work of art. For instance, in developing a play a playwright might reflect upon a particular situation from the viewpoint of the fictitious characters in the play, the actors, the audience or the producer. The simultaneous consideration of all these viewpoints is not associated with a separation of concerns, but with a dwelling in the situation so as to draw out all its possibilities and enrich the experience of the author in the writing and the audience in the appreciation of the work. This holistic approach is vital to the activity, and is complemented by an openness and responsiveness to what is encountered that characterises creative thought.

The act of setting the time on the clock supplies a modest illustration. It might be that the clock is to be sold, and the hands placed in the most aesthetically pleasing configuration. It might be that the user is setting the clock to the current time. The time on the clock potentially represents an aspect of the clock that is beyond user control. The script is

intended to reflect all these possible interpretations, and support them in so far as they can co-exist in the modeller's imagination. The novelty in this approach lies in the ontology of the model: there is no prescribed interpretation, only certain interpretations that may acquire particular significance and permanence as the modeller's imagination shapes the model and the referent.

The description of EM activity is framed in terms of 'how the modeller construes the situation'. Observables, agency and dependency are the key concepts used to express construals. Where there is a fixed external physical referent, construal is concerned with how the observations of the referent can be explained with reference to agency, dependency and observables. It is also appropriate to regard the relationship in which the modeller chooses to stand to the referent as a form of construal, where the emphasis is placed upon the agency of the observer rather than agency that operates within the referent. Interaction with the script can be associated with elaborating a construal of either kind. The parametrisation of the clock face in terms of **noon** and **radius** is concerned with construal from the modeller's perspective. As a simple illustration of how construal applies to the clock mechanism itself, the following revised definition for **hourAngle** establishes the dependency between minute and hour hand that is typically present in a mechanical clock:

```
within clock {
      minAngle = (pi div 2.0) - float (t mod 720) * (pi div 30.0)
      hourAngle = (pi div 2.0)  - ((pi div 2.0) - minAngle) div 12.0
      ...
}
```

Through this redefinition, **hourAngle** depends upon **minAngle**, and is no longer directly defined in terms of the time t. Note also that the definition of **minAngle** has also been modified (so that it records the reflects the number of minutes elapsed over a 12 hour period rather than a single hour) so that it delivers the correct result for **hourAngle**. This can be interpreted as reflecting whether or not we are taking the state of the internal mechanism of the clock into account in observing the position of the minute hand. The relationship between these two scripts illustrates how the removal of dependency can be associated with optimisation and information loss.

The practical techniques that the modeller can use to support construal include the use of definitions to express dependency and the introduction of triggered procedures to represent agent actions. The underlying framework is supplied by the EDEN interpreter, which serves both as an "evaluator of definitive notations" (in particular for DONALD) and as a hybrid definitive/procedural environment that interfaces with the user and the computer. Through EDEN it is possible to increment the variable **clock/t** repeatedly for instance, so as to simulate the clock in operation. The speed at which it is appropriate to carry out this update is at the discretion of the modeller: it may be useful to observe the hands in slow motion, as in the construction of the rotating clock described below, to run it as fast as possible to obtain an overall impression of the clock behaviour, or to synchronise the update of variable **clock/t** with the system clock.

The above discussion illustrates many of the distinctive features of EM in a simple setting. The most significant issue is that the context is such that the term modelling is not entirely apposite (so obscure is the referent and the goal of the modeller): there are many roles for agency, the conventions for observation and interpretation are fluid, and modelling activity is as much concerned with exploring what the referent is as with representing it.

4.3. The SIN principle in EM

The messiness of our real engagement with the world is at odds with the systematic models of behaviour to which a science aspires. In building artefacts that can support this engagement, the idea of making things easy for the user is suspect. It is appropriate to eliminate unnecessary frustration, but not to suppose that all frustration can be eliminated (cf. Donofrio's "what we want, when we want it, where we want it"), or even that that would be a desirable goal. Art works often both with and against the tools and the medium, and this is not something that can be designed away. One aspect of the artist's skill is to overcome the limitations of the instrument: "a bad workman blames his tools".

The evidence is that in certain respects the tools of EM are not easy to use. Even after making allowance for some obvious flaws in the tools – and acknowledging that a bad designer blames the workman – there is an essential reason for difficulty in use. It is quite usual in conventional programming to achieve an end without explicitly considering what assumptions have been made in order to achieve it. In EM, there is no choice but to engage with the experience that should inform our constructions. This activity is expensive in terms of human time and effort – but there is no substitute for it. The potential advantage of the EM approach is that, when we subsequently identify problems, the model itself can help us to access the knowledge that informed our original solution, and that is required to improve this solution.

An actual illustration drawn from developing a rotating clock model is useful at this point. I wish to highlight the noon position on the clock by marking it with a longer line segment. I choose to do this because of the limitations of the tools: the attributes of a line drawing are not preserved if I create a new image of it by rotation. I first think of refining the line **noon**, but realise that this will affect all the other divisions by dependency. This means that I shall introduce a new line **noon2** to mark the noon division. My idea is to derive **noon2** by elongating the line **noon.** This I can do by addressing its endpoints **noon.1** and **noon.2**. My first attempt is:

```
within clock {
      line noon2
      noon2 = [noon.1, noon2*2]
      ...
}
```

This is identified as a cyclic definition, since **noon2** appears on the right hand side of the definition of **noon2** – a mistake precipitated by my own choice of notation. I correct the redefinition of **noon2** thus:

```
noon2 = [noon.1, noon.2*2]
```

This is a conceptual mistake – it results in a spike at **noon**, not a longer line – I am forgetting that the elongation has to be along the direction of the line **noon**. I next try:

```
noon2 = [noon.2 + (noon.1 - noon.2)*2];
```

This a type checking mistake – I am getting confused about what sub-expressions are lines and what are points. To correct this, I enter:

```
noon2 = [noon.2 + (noon.1 - noon.2)*2, noon.2]
```

This is a mistake because the endpoints **noon.1** and **noon.1** are not the way round that I thought they were. Only then do I get what I wanted:

```
noon2 = [noon.1, noon.1+ (noon.2 - noon.1)*2]
```

The interpreter itself poses its own syntactic challenges to correct input, and the steps detailed above were further complicated by such vagaries. Pencil and paper also played a role in supporting his interacton. On several counts, EM offers poor quality end-user interaction, but it is unusual in that it supports a degree of engagement even in error and misconception.

The experimental form of my interaction points to a significant danger: that EM encourages sloppiness in thought and practice. That said, I know that the kind of activity exposed above is quite characteristic of my inner thought processes, and (modulo the vagaries of the tools) that it is harder to trace this on paper or in my head. The end result is also much more satisfactory – not only do I derive the correct answer, but I construct an environment in which my mistakes and misconceptions have been captured and recorded to an extent that is otherwise problematic.

As highlighted in this example, 'making mistakes' is an essential part of employing EM. For the traditional programmer, this is a difficult concept: the most significant mistakes that the skilled programmer makes are in the early stages of design, and hopefully never reach the implementation. Though it can be frustrating and embarrassing to follow through the experimental phases of design in EM interactively – and whilst it is tempting to focus on flaws in the notations, the interface or the interpreter – there is some virtue in exposing our imperfect thought processes.

4.4. Illustrating the SIN principles of representation in EM

This section develops the theme of the simple clock model to illustrate how representation operates in EM. The significance of using EM in representation is best understood by considering the continuity in the cognitive activity that accompanies the modelling. When a conventional program is executing, there are at least two aspects of its state that are relevant. There is the computational state, with which the user may or may not be interacting, which – in so far as it is intended to be interpreted by the user – is meaningful

in terms of the application of the program. There is also – at a meta-level – the state of the program code itself, which is not typically known to the user.

When making a redefinition in a script, the modeller can be changing the associated state in what conventionally would be regarded as affecting both of these aspects. On the one hand, the state that is visible to the user may be changed. On the other, the underlying pattern of dependency maintenance ("part of the program code") may also be changed. The aspiration for EM, to some considerable degree supported even by our current tools, is for it to be possible to change the program code without disrupting that part of the state with which the user is engaging. An analogy can be made between a conventional programming paradigm and the way in which a traveller might use a vehicle built by a mechanic at a workshop, travel about in it to discover its limitations, then return to the workshop so that it can be modified to overcome these. The aspiration for EM is that the mechanic and his workshop can journey with the traveller, to effect modifications in the context where they are needed, potentially with more first-hand appreciation of the requirement.

A few examples will serve to illustrate how EM helps to address issues concerned with situation, ignorance and nonsense.

- Representing situation

The representation of situation in EM can be illustrated in many ways. For instance, to set the clock to Japanese time:

```
%eden
uk_time is tnsecs / 60;
/* tnsecs = the number of seconds that have elapsed since a fix date */
japan_time is uk_time + 480;
_clock_t is japan_time;
```

To represent a broken clock, in which the minute hand hangs loose:

```
%donald
clock/minAngle = - pi div 2
```

It is also possible to take account of observables present in the situation, but previously unrecorded. For example, to add a secondhand that is coloured red (see Figure 3):

```
%donald
within clock {
      line secHand
      real secAngle, size_secHand
      secHand = [centre + {size_secHand*radius @ secAngle}, centre]
      size_secHand = 0.8
}

%eden
sec_mod_60 is tnsecs % 60;
A_clock_secHand = "color=red";
```

- Representing ignorance

The primary respect in which EM deals with missing knowledge is through supporting variables whose value is as yet undefined. The graceful handling of the unspecified time on the clock discussed above is a simple illustration.

Another kind of ignorance is that associated with exploratory design, where something is known only after it is constructed and recognised in interaction. As a simple illustration, the modeller can act in the role of a clock designer via the redefinition:

```
within clock {
      circle inner_edge
      real width_edge
      inner_edge = circle(centre, radius - width_edge)
      width_edge = 20.0
      ...
}
```

Such a redefinition is here to be construed as changing the clock itself.

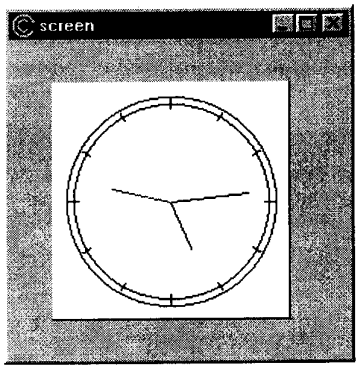

Figure 2: Hand and Rim added

Further experimentation with values of **`width_edge`** elicits tacit knowledge about what is an acceptable design through interaction.

Conventional programming is knowledge-driven, in that a program is designed with a specific functionality in mind. Being able to make use of a model even though we are ignorant about how we might wish to use it is possible to the extent that we have an effective construal. The rotating clock, in which the clock pivots freely about its centre according to the moments associated with its hands, was conceived as an opportunistic extension of the clock model. This construction illustrates the way in which automatic agency, such as here represents the clock rotating into equilibrium as time passes, is developed from – or as if from – a pattern of interaction by the modeller. For instance, the modeller might compute the moments of the clock with minute hand and hour hand pointing vertically downwards, then when the rotation of the clock lies halfway between

these positions, and proceed by binary search to locate the rotation that makes the moment of the clock zero. This can be represented by using the pattern of observation:

```
%eden
hA = _hrAngle;
mA = _mnAngle;
momentH is moment(_hrAngle, _mnAngle, hA);
momentHM is moment(_hrAngle, _mnAngle, (hA+mA)/2.0);
momentM is moment(_hrAngle, _mnAngle, mA);
```

– where **_hrAngle** and **_mnAngle** refer to the current positions of the hour and minute hands in the normal sense, and the function **moment()** returns the moment of the clock about the angle specified by its third argument – and assigning **hA** or **mA** to **(hA+mA)/2** according to the sign of **momentHM**. This binary search can then be performed automatically by an agent that responds to the changing time. This illustrates how EM can be used as a way of specifying the functions that maintain dependencies themselves.

The use of this model to explore the dynamics of the rotating clock is another more familiar sense in which EM is concerned with the discovery of knowledge rather than its exploitation.

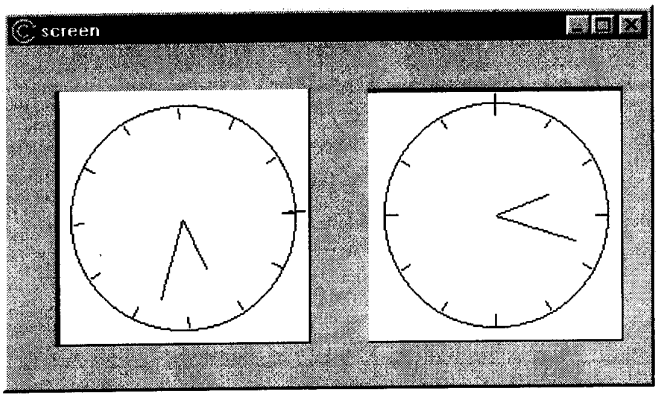

Figure 3: The rotating clock

- Representing nonsense

A much-neglected concern in our representation of the world is the way in which our minds impose relationships upon possible situations and events. We are accustomed all the while to organise our experience according to the degree to which it is familiar and 'makes sense'. This leads us to say "I don't think that's possible", or "if that can happen, then that is also plausible". Though nonsense suggests the antithesis of sense, it in fact refers to what is – in this mind-space – near enough to sense to blend with it in some respects. Perhaps the most important feature of the EM representation of a situation by a script is that it establishes such relationships: namely, nearness as assessed by the kind of redefinition that is required to transform one script to another, and by the kind of agency that would be involved. The extent to which these relationships match those that we encounter in the world reflects the quality of our construal, as determined by the

observables, dependency and agency we identify. The clock study is a useful source of examples.

It is easy to modify the clock so that the length of the second hand depends on the time:

```
_clock_size_secHand is (float(sec_mod_60)/60.0) * _clock_size_minHand;
```

In the days of the mechanical clock, this would have been implausible if not nonsensical, but it would seem unremarkable on a computer desktop. There are simple redefinitions to create a mirror image clock:

```
%scout
window clockwin = {
    type: DONALD
    box: [clockwinNW, clockwinSE]
    pict: "CLOCK"
    xmin: 30
    ymin: 370
    xmax: 370
    ymax: 30
    bgcolor: "white"
    border: 1
};
```

or an the upside-down clock:

```
%donald
clock/radius = -100.0
```

These examples highlight the role that agency of the observer plays in discriminating sense from nonsense: the conventions by which we read the time are in principle so arbitrary. The rotating clock with nothing to distinguish noon from other divisions would be more absurd as a timepiece.

Another extension of the basic clock model serves to illustrate how visual art exploits both the sense of space and 'the space of sense'.

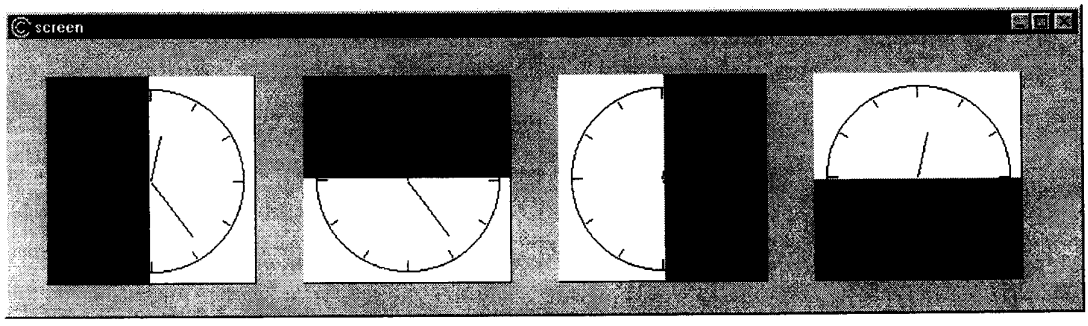

Figure 4: A model loosely based upon Richard Wentworth's The Warwick Dials (1999)

This model is loosely based on a construction on display in the Warwick Arts Centre. It is made by a simple extension of the original model in which the line drawing of the clock is displayed in several windows, each partially occluded by a blank window whose size can be altered by redefining the variable **blankedge**.

```
%scout
integer blankedge = 83;

window blank1 = {
    type: DONALD
    box: [clockwinNW, clockwinSE - {blankedge*clockwinScale, 0}]
    bgcolor: "black"
    border: 1
};

...

display screen =
<blank1/clockwin/blank2/clockwin2/blank3/clockwin3/blank4/clockwin4>;
```

In the space of sense, there is a distinction between nonsense and meaninglessness. In EM, this can be explored by transforming artefacts through random redefinition of variables. The arbiter in matters of sense is the observer, who may or may not be able to connect in any way with the experience offered by the transformation of an artefact. The redefinition of variables that correspond to observables beyond the control of any recognised agent, and redefinitions that subvert our physical intuitions about the permanence and reliability of objects are some of the most effective in destroying the semantic relation, as in the corrupted clock in Figure 5:

```
within clock {
      sixthpi = 1.0
      minHand = [centre + {0.75*radius @ minAngle}, hourHand.1]
      ...
}
```

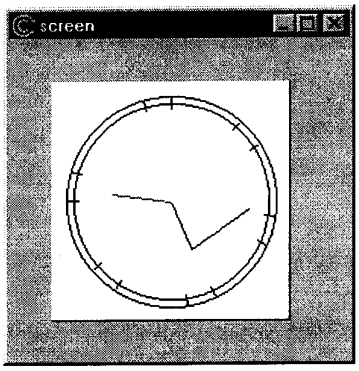

Figure 5: The corrupted clock

4.5. The culture of EM

EM has now been under development at Warwick for 15 years or so. The associated activities have embraced both philosophy [Bey99] and practice. The thinking and scholarship of Dr Steve Russ has been a major influence in framing the main principles and ideas of EM, and in relating these to other schools of thought and practice. To date, the study of EM has been largely confined to a relatively small number of undergraduate and postgraduate students working under the supervision of the author and Russ in computer science. There have been a number of research projects to address specific applications, and some significant related work elsewhere, such as the development of the LSD-engine by a team directed by Valery Adzhiev at the Moscow Engineering Physics Institute and ongoing research on applying Java-based tools for dependency maintenance led by Richard Cartwright, who gained his doctorate in EM at Warwick, at the BBC [Car01]. In so far as there has been an EM *culture*, however, it has been shaped over the last decade by the current group of graduate students, the 5 or 10 undergraduate students who have pursued final year projects on EM themes, and those who have worked on summer vacation projects. From 1992-9, an annual MSc module had a major influence, leading to a regular supply of project students and some significant results.

Most of the practical work on EM has made use of the particular software tool – the EDEN interpreter, an **e**valuator for **de**finitive **n**otations – whose use is illustrated above. The first prototype of EDEN was developed by Yun Wai Yung in his final year project in 1987, and was subsequently extended by his younger brother Yun Pui Yung. Further extensions have been made by Pi-Hwa Sun – to support distributed use, and by Ben Carter – to provide an OpenGL display interface. The current version of EDEN has been made available as open source on SourceForge [SF01] through the work of Ashley Ward.

The simple illustrations described in this paper are quite untypical of the scale of model-building that has been ventured using EDEN, but they do illustrate the essential principles that have underpinned the modelling activity. Though the documentation of tools and models has never been of a commercial standard, many students have been able to make use of them on the basis of a brief introduction to the principles and some exploration of the public directory of previous models. The objective of a project has often been uncertain at the early stages, and a theme has emerged as the model-building activity proceeds incrementally. The profile of work on the project is distinctively different from that practised in other paradigms, such as object-oriented software development. Students typically carry out significant model construction even at the early stages, and are guided by this in their strategic decisions. This mode of working resembles what Levi-Strauss has termed 'bricolage' [L-S66].

Re-use is without doubt an important feature of skill acquisition in EM. Though there is ostensibly scope for the development of a library of components, and there have been several attempts to simplify the task of building up scripts rapidly from components using a GUI , the low profile of functional specification in EM and the prolific number of minor modifications that every script affords make standardisation and categorisation

problematic. Typical re-use in EM has instead been characterised by extraction of script fragments and their customisation and assimilation into other models. Engagement with EM is not consistent with plagiarism. The themes and techniques represented in successful EM projects have invariably reflected the individual characteristics of the student, and the public directory of projects more closely resembles an art gallery than an engineering catalogue.

There are nevertheless some archetypes amongst the students. Each year, there have been a few students who have shown quite exceptional ability to make effective use of EM, overcoming the limitations and frustrations of the tools, generally pursuing a topic of strong personal interest, developing their own style, taking the tools to their limits and beyond, and leaving a legacy of interesting model fragments. For other very able students, especially those with an engineering or formal computer science background, the encounter with EM has been distinctly less rewarding: they have found the adaptation to an alternative way of thinking about model-building difficult – even stressful, been uneasy over the lack of a methodological framework and formal roots, and have not had the respect for the tools or the confidence to use them in the way in which they are intended to be used. In relative terms, EM has proved much more successful with weaker students, some of whom have been unable to come to terms with conventional programming, find formal approaches to analysis and organising knowledge uncongenial but revel in uncircumscribed exploration.

If in some respects these aspects of EM culture are redolent of art and craft rather than science, there is strong evidence for collaborative possibilities in EM model-building that would be unusual in art. The key to much of the interaction between modellers in EM is that comprehension of the models themselves is best acquired through interaction with the model. For this reason, it has been possible for relatively ambitious models to be built through the collaborative efforts of several people, more often than not with little direct communication with each other. Where communication is required, it typically centres upon a particular situation that can be established within the model and serves as a much more specific focus for corporate attention than a segment of procedural code, or a complex formal specification. Collaboration in this mode suits a particular social disposition, in which the idea of software as a shared resource is congenial and prominent. This contrasts with the concept of art as an exclusive product to be protected from reworking and valued for its authenticity.

Conclusion

The liberation of the computer arts is centrally concerned with a shift in philosophical stance. The computer arts cannot thrive in a culture that presumes that all experience is mediated by The 4R's Principle of science, and that – in consequence – no computer model can be developed without preconceptions about what is to be represented and why. Dwelling in experience without preconception is an essential feature of human creativity. It can only be fully enfranchised by an alternative philosophical outlook such as William

James adopts in his essays on Radical Empiricism [Jam96], where 'The World of Pure Experience' is prior to language and reason.

The rule-based orientation of today's computer science echoes the rationalist F H Bradley's critique of William James's 'philosophic attitude' a century ago, and still informs our social practice. To respond to our environment in ways that are neutral to purpose and invoke no preconceived mode of observation runs counter to many trends in modern culture. Good management practice encourages us to be purposeful, focussed, and to strive to see the world without illusion. Imparting an administrative ethos to teachers and pupils alike is deemed integral to good educational practice, and education itself is all too often perceived as "teaching the rules". The dramatic developments in neuroscience and genetics over the last few decades have brought the human being itself into the sights of scientific and computational reductionism.

The relationship between Empirical Modelling as an alternative approach to computing and classical Computer Science is similar to that between Radical Empiricism and Bradley's rationalism. Both EM and Radical Empiricism distrust the idea that systems of belief are essentially prior to experience, and both are associated with a philosophical stance, not an alternative system or method of operation. As seems appropriate to the theme of this paper, I have included a poem of my own that expresses for me this position – to the rational mind, somewhat paradoxical – that scepticism about systems of belief is not itself a system of belief (see Poem 2). Suspicion of systems as a foundation for our experience is not to be construed as cynicism. On the contrary, as a way of looking at the world that respects its mystery, it enhances our vision of the scientific, the artistic and the spiritual. This accords with the outlook of the German theologian Martin Buber [Bub66], for whom rationalising and articulating our experience was associated with a shift of deep spiritual significance: from the *I-Thou* to the *I-It* relationship. Such an outlook is not intended to subvert the aspiration to understanding, but to place it in its proper perspective. In the words of the Welsh poet John Ormond [Orm73]: "I await wisdom, wise enough to know it will not come.".

Acknowledgements

I am indebted to Steve Russ for many of the ideas that are developed in this paper, and to all the many students of Empirical Modelling, past and present, who have taught me so much about the subject.

I am most grateful to Carl and Jody Vilbrandt for their generous invitation to present my paper at this workshop, and for their personal encouragement.

I thank Islwyn Davies for an influence over my thinking that has extended over more than forty years, and that has found clearer expression in this paper than any before.

Above all, I am indebted to my father, William John Granville Beynon, to whose memory and inspiration this paper is dedicated.

Acknowledgement is also due to the Royal Society and the University of Aizu for their financial support.

References

[Bey99] W M Beynon, *Empirical Modelling and the Foundations of AI*, Springer LNAI 1562, 1999, 322-364
[Bro87]. Fred Brooks, *No Silver Bullet: Essence and Accidents of Software Engineering*, IEEE Computer 20(4), 1987, 10-19
[Bub66] Martin Buber, *I and Thou*, T and T Clark, Edinburgh, 2nd edition, 1966
[Car01] Richard Cartwright, *Distributed Shape Modelling with EmpiricalHyperFun*, also in these proceedings
[Don01] Nick Donofrio, Vice-President of IBM, Address at the University of Warwick, January 2001
[Goo90] David Gooding, *Experiment and the Making of Meaning*, Kluwer 1990
[Jam66] William James, *Essays on Radical Empiricism*, Dover Books 1996
[Orm73] John Ormond, 'Salmon', in *Definition of a Waterfall*, Oxford University Press, 1973
[L-S66] C. Levi-Strauss, *The Savage Mind*, University of Chicago Press, 1966
[Tur96] Mark Turner, *The Literary Mind*, Oxford University Press, 1996
[Wie52] Wiener, *The Human Use of Human Beings*, 1952
[SF01] http://sourceforge.net/projects/eden
[EM01] http://www.dcs.warwick.ac.uk/modelling

Wie Melodien zieht es mir leise durch den Sinn,
Wie Frühlingsblumen blüht es und schwebt wie Duft dahin,
Doch kommt das Wort und fasst es und führt es vor das Aug',
Wie Nebelgrau erblasst es und schwindet wie ein Hauch,
Und dennoch ruht im Reime verborgen wohl ein Duft,
Den mild aus stillem Keime ein feuchtes Auge ruft.

Klaus Groth, set to music by Johannes Brahms as op.105 no. 1

Like melodies, it drifts through my mind,
and floats away, like the scent of flowers in bloom.

If words try to grasp it and bring it into focus
It vanishes as into the mist; like a breath it fades away.

Yet in a rhyme can linger, hidden deep, a scent
that, remembered, some day shall make you weep.

Groth's poem illustrates the role that situation, ignorance and nonsense play in art. The poet's concern is with evoking a particular situation, familiar to us all, when impressions that we cannot capture come and go in our mind. He explicitly acknowledges the limitations of words as a representation for them. His preoccupation is with other media – smell and sound, and metaphors – melody, flowers, mists, sighs, perfume. The poem reflects an awareness of the elusiveness of knowledge; over and above this, the poetic representation that is recognised to be inadequate is also seen as delivering what was not explicitly represented – nonsensical and irrational as this would appear to be. The intensity of the imagery with which the poem concludes suggests that what has been expressed unconsciously is more emotive than could be grasped rationally in the original situation.

Poem 1: Illustrating the SIN principle for representation in art

Faith

You, in your super-laboratory,
With your carefully careless hair
And your precise eyes hinting genius,
If you should say to me,
Pouting at your tangled equations
In any number of dimensions:
This is the key to the universe.
I do not believe you.

You, in your priestly pinafore,
With your devilish seriousness
And your saintly sense of humour,
If you should say to me,
Ranting at your faceless gods
In parables and prophecies:
This is the key to the universe.
I do not believe you.

You, in your poetic pose,
With beauty at your elbow
And your brow in anger,
If you should say to me,
Taxing my vocabulary
With your obscure imagery:
This is the key to the universe.
I do not believe you.

Distrust the physicist, the priest, the poet:
This is the key to the universe;
Now you know it.

Do not believe me.

Poem 2: *Faith* - by the author - March 1986