Beynon, M. (2007). Computing technology for learning - in need of a radical new conception. *Educational Technology & Society*, 10 (1), 94-106.

Computing technology for learning - in need of a radical new conception

Meurig Beynon
Computer Science, University of Warwick, Coventry CV4 7AL, UK
wmb@dcs.warwick.ac.uk

ABSTRACT

Many have had high expectations for the impact of computer-based technology on educational practice. By and large, these expectations have not been realised. It has become evident that innovative technology alone does not necessarily guarantee progress - nor perhaps even significant change - in educational practice. This has led educational researchers to place greater emphasis on cultural issues that could account for the unexpectedly limited influence of technology-enhanced learning. This perception of the relationship between technology and learning is elaborated in the first section of the paper. It is complemented by a review of an alternative conception of computing, rooted in a methodology for modelling with dependency directed at the development of *construals* rather than programs, that is far better aligned to the demands of developing environments for learning. The paper concludes with a discussion of the potential implications of this approach.

Keywords

Educational technology, Learning, Constructionism, Dependency, Construal, Modelling, Computing

Introduction

In accounting for the limited impact of digital technology upon education, attention has shifted from the technology itself to the cultural context surrounding technology-enhanced learning. Education, like other areas of computing application, has endured too much hype to be sure of the contribution that further technological innovation can make to advancing educational practice. The best insights into how learning takes place all motivate a broad view of the educational context embracing psychological, social and political agendas.

Against this background, it may be surprising that this paper concerns a new approach to computer-based modelling, conceived within computer science, that has involved developing new software tools. To appreciate this as more than 'yet another technological innovation' involves recognising the deeper pretension behind this approach. Whilst strategic research interest in educational technology has shifted away from the technological perspective (witness the disqualification of 'technology-led' proposals from recent EU calls relating to Technology Enhanced Learning (Manson, 2004)), this paper argues for a radical reappraisal of the role of computers in learning based on alternative principles and practice for computing. Its thesis is that the accepted conceptual framework surrounding computing technology is critically ill-suited for creating environments for learning. Moreover, this conceptual framework has such an intellectual pedigree and authority, and such integrity when applied within its proper remit, that it has exercised a powerful and pervasive influence over our conception of human cognition, with far-reaching implications for business and education.

Rethinking computing technology

Whatever remains controversial in technology-enhanced learning, any technology that can truly assist learning must clearly stand in a very intimate relationship to human cognition and action. By this criterion, information and communications technology, as presently conceived, is ill-equipped to engage with the universal and profound issues encountered in human learning. What is required is a perspective on computing that integrates human understanding and interaction with the development of artefacts and systems in a dynamic and holistic manner. For sound historical reasons, the fundamental conceptual framework for computing that informs the development of software and the analysis of broader computer-related activities is oriented towards a quite different agenda - achieving functional efficiency, typically through automating human cognition out of the loop.

How computing technology is *conceived* may seem to have less practical relevance than how this technology is developed and applied. Yet Papert asserts - with feeling: "Wild imagination, passion, being close to nature, and

believing in magic ... I think these are all elements that we need to bring into the otherwise cold version of use of computers called 'ICT' ..." (Papert & Strohecker, 2005).

The lack of "magic" attending ICT is symptomatic of a more general and influential trend towards demystifying human affairs. In this connection, it is instructive that the primary proposal for countering the lack of excitement, mystery and imagination in the ICT curriculum (Lovegrove & Round, 2005 p.11) is to provide suitable materials, "concentrating on people and inventors, ideas and concepts, and applications of computing in everyday life (e.g. 'how your mobile phone works', 'how Google works', etc.)". There is a pertinent contrast with the mysteries of traditional science, which relate to unidentified phenomena and processes that are imperfectly understood. The meanings that classical computer science is best equipped to handle exclude such exalted concepts of ignorance and mystery - they are framed with reference to formal symbols whose identities and associated values must be made explicit, and to processes that follow prescribed algorithmic rules. Received understanding about computing involves exploiting what has already been learnt, rather than engaging with what is unknown, un-mastered or *as of now* being learned.

The accepted conception of computing technology is not only pertinent to the pervasive applications of computers; it has also exercised an exceptionally powerful influence over our whole conception of human activity. The tacit practical effect has been to privilege "rationalistic" notions (Winograd and Flores, 1987) like preconception, planning, formal languages, abstraction, methods, recipes, logic and rules *as if* they were the primitive ingredients of all human behaviour. In the process, activities that are of enormous significance in relation to human learning, concerned with creation, invention and "wicked problems" (Lansdown, 1987) in art, craft and design, and with experiment, discovery and "blind variation" (Vincenti, 1993) in science, mathematics and engineering, have been marginalized - or bowdlerized - in the educational and political discourse. Significantly, in such activities, it is not human ingenuity that primarily invokes a sense a mystery, but the subtlety of the phenomena and experiences that defy expectation and explanation.

The contemporary influence of computer science thinking on education is hybrid. The authentic intellectual legacy of Turing's remarkable characterisation of algorithmic thinking has spawned cognitive models based on a computational philosophy of mind. The limitations of the mind-as-machine metaphor as a foundation for theories of learning are now well-recognised. For instance, in *Acts of Meaning* (Bruner, 1990) - as paraphrased by an anonymous reviewer - the distinguished educational psychologist Jerome Bruner "elaborates on the failure of cognitive science in abandoning 'meaning-making' for 'information processing', and its attendant concentration on computational logic."

A key motivation for seeking a new conception of computing technology is that - whilst the authority of the classical theory of computation in studying and reasoning about algorithms is beyond question - the complementary aspects of computer science, concerned in particular with the problems of developing reliable, large or adaptive software, have by comparison most controversial credentials. These problems are moreover intellectually intimately connected with learning - they relate to human activities that precede the clarification of concepts, the identification of mechanisms and the comprehension of systems, and in general involve complex personal and social interaction between designers where issues of inter-subjectivity and conflict have a significant role. Though technologies such as objects and patterns have proved effective in certain contexts, the major limitations of current principles and techniques for general software development are well-documented (Brookes, 1987; Brookes, 1995). Significantly, development strategies involving objects and patterns are principally concerned with exploiting prior knowledge of application domains, and re-using artefacts and methods, rather than the more primitive and fundamental issues surrounding their initial construction.

In an educational technology context, more is required of a new conception of technology than a further contribution to the critiques of the 'rationalistic' traditions of computer science and artificial intelligence (e.g. McDermott, 1987; Winograd & Flores, 1987; Smith, 2002). The primary motivation is the practical development of useful software products to meet educational needs. Software development has long been centrally concerned with ways of circumventing the problems of premature specification and uncertain knowledge. The recent adoption of eXtreme Programming (XP) approaches (Beck, 1999) has sought to manage software production so that implementation and testing are prominent from the outset and the design is elaborated incrementally with no overriding initial preconception. The objectives of XP are well-aligned to those of constructivist learning: the developers aim to learn about the application domain and the user requirements in and through the process of developing a software product.

The key problem considered in this paper is that the fundamental conception of a computer program and the modelling apparatus surrounding traditional software development (in whatever idiom) offers very limited support to the discourse of primitive learning. From an educational perspective, the significant distinction is that between *representing* and *construing* identified by Mason (1987). Representing involves establishing conventions for associating formal symbols with entities; construing involves giving an account of the experiences that inform personal understanding.

In the traditional conception of computing, the separation between those elements admitting mathematical formulation and the informal experiential aspects is seen as sharp. Making computer representations involves committing to specific interpretations that cannot be revised in execution of the program or system. Human learning activity, in contrast, often begins in confusion, and may involve the reappraisal of interpretative conventions even as they are being employed. A major challenge in deploying computer technology in support of constructivist learning is that traditional representational principles do not accommodate new modes of interpretation that emerge through interaction with a computer artefact, nor support the high degree of integration of learner, teacher and developer roles that learning-by-constructing ideally presumes. Commitment to interpretations and goals is the price traditionally paid for optimization, and incurs notorious inflexibility when changing requirements. This is particularly relevant to educational software; because of the extreme variation in learner characteristics and capabilities, the most subtle changes in requirement are of the essence.

A radical new conception of computing technology

This section outlines a programme of research into a new conceptual framework for computing being pursued by the author and his collaborators. *Dependency maintenance* is the key technology we exploit. *Dependency* is familiar from spreadsheets, where an update to one cell leads to other cells being updated as if in one operation. Our previous research, spanning some twenty years, has elaborated on spreadsheet principles, and explored applications to many areas in addition to educational technology (For more background, consult the Empirical Modelling website at the Empirical Modelling website: http://www.dcs.warwick.ac.uk/modelling).

The general principles behind our modelling framework are briefly reviewed in the context of related research that has exploited dependency, and illustrated with reference to three modelling exercises.

Dependency in software construction

Dependency features widely in software. It is most conspicuous in spreadsheets for handling quantitative data, but spreadsheet principles have been applied to modelling scientific data and extended to support data visualisation. Functional dependencies are fundamental to relational database design, and in defining views. Dependency is exploited in modelling software for engineering and business, such as LabVIEW (National Instruments website: http://www.ni.com) and Attribute Explorer (Spence, 2000), where graphical representations are automatically maintained to be consistent with the underlying data. The potential role for dependency in interface design has been demonstrated in software such as Amulet (Myers et al, 1997). Abstract forms of dependency underpin concepts such as 'object linked embedding', style definitions in word processors and make utilities for complex software maintenance. Dependency analysis is now being deployed in applications such as maintaining large websites, and reducing the storage required to record the accessible assets in computer games (Carter, 2003). This has motivated the development of principles to support dependency maintenance within an object-oriented programming context (e.g. Perry, 2001). Dependency features prominently in mechanical devices developed for representational purposes prior to the modern computer, and is intrinsic to the analogue devices that pre-dated digital computers (Small, 2001).

In applications, dependency often serves a characteristic cognitive function of imitating relationships between external observables that help the modeller to recognise the intended meanings of variables and components within a model. For instance, there is direct correspondence between interactions with an examination spreadsheet and the way that the average mark on an examination paper is affected by discounting the absentees. This highlights the significant role for dependency in making computer models meaningful, and underlies the numerous applications of spreadsheet-related principles in education (Baker & Sugden, 2003), particularly - with constructionism in mind - in relation to end-user programming (Repenning, 1993). The use of dependency to which this paper specifically refers is intimately connected with ensuring that the meaning of a model can be directly apprehended by the human

interpreter, and relies upon principles that are not in general respected where dependency has been used in an *ad hoc* manner, or exploited merely as an abstract programming device in conjunction with conventional programming techniques. These principles define a methodology for model-building to which our use of the expression *modelling with dependency* refers.

Our methodology for modelling with dependency

Our applications of dependency in model-building rely upon a particular methodology for domain analysis. In this methodology, the modeller acts in a role resembling that of an experimenter applying the 'scientific method'. Model construction is incremental, and evolves along with understanding of the domain that emerges through identifying patterns in interaction within the domain. These patterns are expressed in terms of key *observables*, *agents* deemed to be responsible for changes to these observables, and *dependencies* reflecting the way in which changes to observables are linked. The construction of models that embody the emerging patterns of observation, dependency and agency is supported by special-purpose notations and tools.

The motivation behind our methodology is that - if we are to support learning in cognitive terms - our construction of software should be informed not merely by a narrow requirement, but - to the fullest possible extent - by domain understanding. In our modelling framework, this translates to "identifying the most appropriate way to conceive interaction within the domain with reference to the agents that we deem to be acting, the observables that we deem to mediate their interaction, and the dependencies that we deem to govern the synchronisation of changes to observables". For brevity, this is termed "identifying the most appropriate *construal* of interaction within the domain". Modelling with dependency is then interpreted as 'building construals' (cf. Gooding, 1990). The soundness of our methodology rests upon the thesis that such construals are an appropriate way of expressing fundamental understanding of domains of interaction.

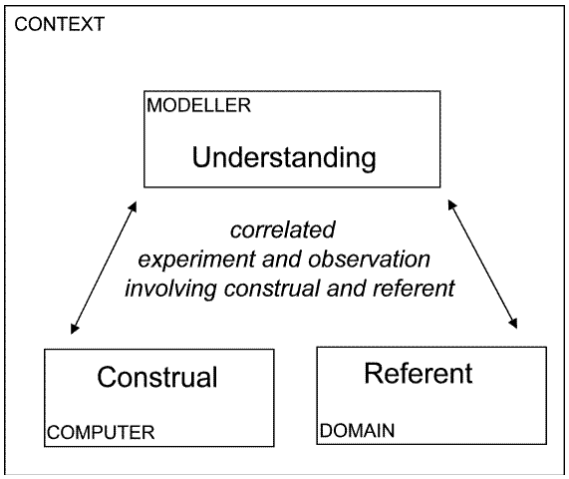

Figure 1. The archetypal setting for modelling with dependency

Our thesis about domain understanding informs a distinction between modelling with dependency and computer programming (as traditionally formally conceived). A program constructed merely to achieve a specific functional goal, does not - in general - require a deep construal of interaction in the domain. However, such a program will be unhelpful - if not obstructive - if new requirements or insights must be taken into account. There is only a loose and complex relationship between the separate concerns of developing deep understanding of domain interaction and devising effective recipes to solve specific problems.

Practical principles for modelling with dependency

Modelling with dependency relates to an archetypal setting for interaction (see Figure 1). The modelling activity depicted in Figure 1 is to be viewed from the perspective of the modeller, who has direct immediate personal

experience of an artefact that has been generated using computing technology ("their construal") and of some independent situation, phenomenon or more ill-defined focus of attention to which the artefact refers ("the referent"). At any moment, the modeller experiences the construal and its referent in just one of their many possible states. The modeller stands in the role of an experimenter who can perform state-changing actions on both the construal and its referent, and may also be able to configure them in context to exhibit some autonomous behaviour. The construal serves to mediate the modeller's understanding of its referent. This understanding stems from familiarity with interaction with the construal and referent that is attested by recollection of past patterns of interaction and reliable expectations about the consequences of future interactions. There need be no formal basis on which the construal relates to understanding of the referent. The presumption that guides the modeller is that the correlation between the current state of the construal and that of its referent can itself be experienced. The modeller aspires to bring about this state of affairs through engineering the context, making the construal, identifying the referent, and rehearsing interactions.

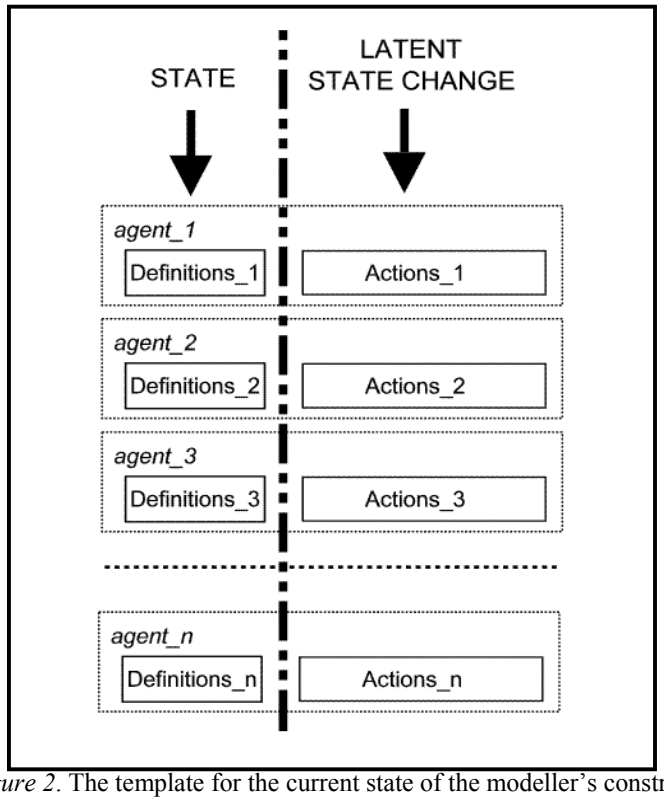

Figure 2. The template for the current state of the modeller's construal

As a familiar example, the modeller may be constructing a spreadsheet to reflect their personal financial situation. Simple as this example is, it illustrates many of the subtleties of the semantic relation between construal and referent and of the modeller's role in shaping this. For instance, interaction with the spreadsheet may be solely concerned with tracing prosaic transactions on an account. Alternatively, it might involve forecasting for a fictional future scenario. It might entail simulation of future financial circumstances over an extended period of time, perhaps with opportunistic intervention by the modeller aimed at accounting for expected but unquantifiable bonuses. Other extensions of the spreadsheet might involve extending the semantic range of the spreadsheet - dividing income into categories or reprogramming to accommodate changes in tax law. Note that the role the spreadsheet plays in mediating understanding is contingent upon tacit constraints on the modeller's interaction. Relevant issues include: *What are extreme values for an exchange rate? What components of income are not taxable? Does it ever make sense to base a computation of profit on the square of the income?* The depth of understanding that can stem from such modelling depends upon how much discretion the modeller has over the context. From an educational perspective, the scope for using such a model as a "learning artefact" is diminished essentially if - in the spirit of traditional programming - constraints upon interaction are formalised, rendered explicit and imposed. Nor is it

appropriate to regard the modeller as merely a 'spreadsheet user', since the interactive possibilities encompass significant and fundamental redesign.

In conceptualising the spreadsheet as a construal, both the dependency relations between cells and the visualisation and annotation of values within the grid are vitally important. The use of dependency can usefully transcend maintaining relationships between scalar quantities, as when negative quantities are displayed in red, or the spreadsheet is dynamically linked to a chart. Our framework supplies generalised support for modelling with dependency by enabling the modeller to formulate spreadsheet-style definitions linking the values of observables of a diverse kinds. For instance, definitions can connect scalar quantities with locations, attributes and textual annotations associated with drawings and screen displays. As in a spreadsheet, no explicit program code is required to ensure that the associated dependency relationships are maintained when values change. A family of definitions serves to express the state of the modeller's construal as it is currently presented for ongoing observation and experiment. A typical primitive interaction by the modeller involves modifying, adding or removing a definition. The semantic significance of such an interaction need not be preconceived - it may involve a "classification of experience" to be negotiated according to the modeller's intentions and conception of the context. Where the referent is perceived as acting autonomously, certain groups of observables may be viewed as ‘agents’ whose actions can be expressed as sequences of redefinitions and/or invocations/dismissal of other ‘agents’. A typical agent action then takes the form of such a sequence together with an appropriate cue for its execution, expressed as a relation between observables to which the agent can respond.

The idealised template for the current state of the modeller's construal in modelling with dependency is depicted in Figure 2. The association of definitions of observables by agents at first merely reflects the integrity of clusters of observables that are perceived to co-exist, being invoked and dismissed together. The actions associated with such a family of definitions are interpreted as actions attributable to the agent, and are subject to be performed automatically. In Figure 2, the set of actions or definitions associated with an agent may be empty if it is construed as having no capacity for state-changing action or significant observables. In principle, the modeller is free to modify, add or remove any definition whatever at any time, and such an action is meaningful in so far as it respects - or informs - the relationship between the construal and its referent.

Some illustrative examples

The above overview of the principles of modelling with dependency will be complemented by a brief description of how they are applied in the three modelling exercises. (For more details of each of these exercises, consult the online posters at the Empirical Modelling archive: http://empublic.dcs.warwick.ac.uk/projects; kaleidoscope, n.d.).

The JUGS model: The JUGS model was first conceived as a generic basis for developing variants of a simple educational program for a range of different computing platforms (Beynon et al, 1988). The original program simulated the actions of filling, emptying and pouring liquid between two containers ("jugs") of integer capacity. The program was designed to support a mathematical investigation into what quantities of liquid could be generated in this fashion (as is in fact determined by the highest common factor of the capacities of the jugs). The visual form of the JUGS construal is quite simple - each jug is represented by a textual pattern to represent the outline of the jug on which is superimposed a display window of variable height to represent the liquid content. This display is complemented by a set of buttons for performing “fill”, “empty” and “pour” operations whose responsiveness and display status depends upon the availability of these options. In relation to Figure 1, this construal can be interpreted either as referring to an actual pair of containers of appropriate sizes, or as giving concrete visual expression to abstract numerical quantities. With reference to Figure 2, the jugs themselves are passive agents that are represented by a family of definitions that express the dependencies between observables such as the capacities and contents of the jugs, the geometric characteristics of their display, and the status of the menu options. These passive agents are complemented by agents associated with the menu options whose actions are cued by the corresponding button presses. The JUGS construal is quite different in character from a typical program to realise the narrow functional specification implicit in its button interface. This is decisively demonstrated by the wide range of alternative technical and semantic purposes to which it can be readily adapted. Interaction with the construal illustrates how the notion of a container being *full* is context dependent, for instance. The construal can also be used in conjunction with alternative interfaces, adapted for more precise and realistic representations of liquid and containers by taking leakage and evaporation into account, and extended to explore such representations within the engineering limits of

the screen display. By acknowledging the possibility of radically different alternative views of the context and referent (cf. Figure 1), the underlying observables and dependencies can also undergo a metamorphosis whereby variants of the JUGS construal derived through relatively few redefinitions can refer to persons queuing at a bar, or to the strings and frets on a guitar (For more background, see Beynon et al, 1989; EMarchive:jugspresentationKing2005.).

The Clayton Tunnel model: This model illustrates how the principles of modelling with dependency can be applied to multi-agent simulation that is realised on a distributed network of workstations. The model serves to animate railway operation in the vicinity of the Clayton Tunnel in 1861. The construal in this case is directed at gaining a better understanding of the events that led up to a serious railway accident. The definitions and actions associated with the various agents (cf. Figure 2) reflect the realistic possibility that different agents have different perceptions of the "same" observable. The construal also incorporates a putative "objective" view such as might be attributed to an external observer. The model is a rich source of insight into the fundamental issues that surround reconciling subjective agent perspectives with an objective reality (For more background, see Beynon, 2005a.).

The Erlkönig model: This model was developed to illustrate the potential for applying modelling with dependency in the humanities. The model expresses one possible way of giving visual and dynamic expression to Schubert's treatment of classical harmony in his famous setting of Goethe's ballad Erlkönig. Dependency is used to associate a node in the cycle of keys and a colour (as determined by the current key) with each moment in a performance of the song. The pattern of the harmony throughout the song is displayed in the form of a ribbon of colour. Because the way in which Schubert exploits harmony is open to critical human interpretation, there is no convergence to a canonical interpretation, but an essential openness to revision and critique. This undermines the notion of the model maker as merely an objective observer of a system behaviour - the role of the modeller in relation to the behaviour of model is instead somewhat analogous to that of the performer (For more background, see Beynon, Russ, McCarty, 2006; Beynon, 2006b.).

Potential implications of modelling with dependency

Modelling with dependency is associated with a radically different conception of computing. It involves a reorientation that goes beyond developing a new programming language or even a new programming paradigm. In the terminology of the humanities, it entails "deconstructing" the notion of program, and initiates another "discourse".

Key characteristics of modelling with dependency

The conceptual core of current thinking about computing is concerned with controlling and predicting the corporate behaviour of agents whose capabilities for state-change and interaction are highly reliable, with its associated contextual overtones of system-like character and objective external observation. Even in areas where experiential aspects take precedence, as in research on artificial life, micro-worlds and simulation environments, or in applying alternative development techniques, such as genetic algorithms or neural nets, the computational focus is on crafting behaviour. The underlying metaphor for the interactive computer artefact is the machine or system, and the associated notion of human-computer interaction reinforces the divisive duality inherent in programming. In this - even though they express an aspiration to integrate human and computer activities - the notions of 'user-centred' and 'learner-centred' design are complicit. And whilst there is considerable interest in conceiving human interaction with technology as *process* (e.g. Warboys et al, 1999) and *narrative* (e.g. Turner, 1996; Papadimitriou, 2003) rather than formal symbolic communication, such abstractions also target behaviours.

Modelling with dependency, by contrast, puts the primary emphasis on the situation as experienced by the modeller. The modeller's subjective understanding of the situation is mediated by observation and interaction. Where appropriate, behaviours emerge as patterns of interaction that prove to be predictable within a contrived context, and some of these may be objective in that they can be executed by an external agent and can appear to be independent of the observer.

Within the symbolic frameworks of orthodox computer science, the difficulty of shifting the focus from 'behaviour' to 'situation' is illustrated by the problematic status of database theory (Ridley, 2003). In modelling with dependency, this difficulty is addressed by giving priority over any symbolic interpretative framework to the semantics associated with the modeller's direct experience of interaction. To illustrate how this semantics is invoked, imagine that all annotations by way of names and identifying labels were removed from a spreadsheet constructed for some familiar purpose, such as recording examination marks. Interaction with the spreadsheet would still enable us to identify the intended meanings of the rows, columns and cells through experiment, by observing how changes to certain cells affected the values of others according to certain patterns. Such identification might even be possible if the cells of the spreadsheet were randomly relocated whilst preserving their interrelationship. In effect, provided that the dependencies between its attributes are appropriately captured, a meaningful artefact can be presented simply by exhibiting any one of its possible characteristic states and allowing the human interpreter to change attributes freely. In this way, semantic connections are potentially being made and appraised live, here and now.

The appeal to semantics that is fundamentally rooted in direct correlation of immediate experience distinguishes modelling with dependency from other non-symbolic approaches for which the ground of knowledge is more ambiguous. Modelling with dependency resembles the grounded experimental activities that a scientist conducts prior to developing a theory, and that craftsmen and engineers perform outside a formal framework for understanding. By its nature, far from being an abstract transcendental concept, dependency is a matter of private experience conditioned by the performance and interface characteristics of the underlying hardware, the sensory, interpretative and manipulative skills of the modeller, and the specific context for the experimental interactions. As discussed in (Beynon, 2005b), this is consistent with the key tenets of a "philosophic attitude" espoused by William James (1912):

> *Knowledge of sensible realities ... comes to life inside the tissue of experience. It is made; and made by relations that unroll themselves in time. (p56)*
>
> *... the "truth" of our mental operations must always be an intra-experiential affair. (p202)*

Modelling with dependency applied to learning

The ontological distinction between modelling with dependency and traditional programming parallels the two research strands of 'learning and knowledge' and 'learning and cognition' alluded to by Manson (2004). This can be seen as reflecting the tension between what Brödner (1995) identifies as the 'open development' and 'closed world' cultures in engineering. In spirit, modelling with dependency has much in common with Vincenti's (1990) vision of engineering as an epistemological species distinct from applied science, where there is a role for blind variation - interaction 'without complete or adequate guidance' potentially leading to discovery.

This distinction is reflected in modelling with dependency and traditional modelling as they apply to learning. In the use of simulation tools such as STELLA to model systems dynamics (Doerr, 1996) or ScienceWare's Model-It tool (Soloway et al., n.d.), for instance, the computer serves to animate a behaviour determined by specifying relationships and setting parameters. Such an activity could only be regarded as "modelling with dependency" subject to the modeller being in some way equipped to directly apprehend the effect that changing parameters and relationships has upon system behaviour. Much mathematical modelling of complex systems by computer is in this spirit - it exploits theoretical insight for which the empirical basis is of historical rather than topical interest, as when the modeller routinely applies Newton's Laws of Motion.

The aspiration in modelling with dependency is similar to that expressed by Riley in his discussion of "learning by modelling" and "learning by making sense of computer models" (Riley, 2005a, 2005b), but the nature of the models - as construals rather than programs - is much better suited to realising the degree of subtlety and universality that is envisaged in the "virtual culture" he invokes. The informal and provisional way in which a model derives meaning from the human interpreter's immediate experience of its contextualized state contrasts with the way in which the formal semantics of computer programs is expressed with reference to comprehensive state spaces and behaviours. This makes it possible to create environments in which to explore subjective understanding, and embody tacit and pre-articulate knowledge. It also serves to situate the accepted conception of computing in a richer context of human computer interactions, where 'concepts', 'languages' and 'systems' are emergent rather than primitive. This supplies the appropriate setting in which to tackle questions, identified by Doerr (1996) as problematic in the educational applications of STELLA, that concern the validity of models, the appropriate use of 'stores' and 'flows', and the

relationship between different representations and their cognitive implications. Perhaps more significantly, it supplies a framework within which pragmatic context-dependent answers are most naturally accommodated and justified.

On reviewing the applications of spreadsheets in education, Baker and Sugden (2003) conclude: "There is no longer a need to question the potential for spreadsheets to enhance the quality and experience of learning that is offered to students." We attribute this potential to the fact that spreadsheets serve as construals, and see modelling with dependency as promising further enhancement to the learning experience. The illustrative examples mentioned above highlight several relevant features of modelling with definitions that are best appreciated through carrying out an extended exercise and observing the ways in which a model develops over a period of several months. They include:

- support for model-building in an open unsystematic fashion, where the order in which definitions are introduced is unconstrained, and the focus of attention can range freely in developing and interconnecting different components and semantic aspects;
- provision for recording partial and provisional relationships, for recovering previous states and versions, and for refining, updating and correcting definitions retrospectively without comprehensive redesign and possibly without extensive model redevelopment;
- means to extend the semantic range of the modelling tool by introducing new types and operators on-the-fly that make it possible to interpret and maintain dependency relations between new kinds of observable.

These practical qualities have implications of particular relevance to interactive environments for learning:

As a hybrid activity guided by domain understanding as it emerges, modelling with definitions enables the blending of propositional and tacit knowledge associated with development and use of models that is essential in active learning. This is in contrast to environments for active learning in which the learner either plays the role of a programmer (as in Logo) or of a user (as in interaction a microworld). Unpublished research by Roe (2005) provides more context for this claim by demonstrating the significant practical advantages and methodological implications of introducing dependency maintenance to state-of-the-art tools such as Imagine Logo (Kalas and Braho, 2001).

Modelling with dependency has potential application to collaborative learning. The benefits of spreadsheets as shared computer-based models for distributed and concurrent interaction in collaborative environments are well-recognised (Nardi, 1993). Construals can embody each individual learner's understanding of a situation, and be adapted to underpin novel processes of interaction and evaluation. For instance, in evaluating student performance in a model-building exercise, a teacher can keep a comprehensive record of their interactions with the model, and return this to the student as an auditable source of feedback.

Modelling with dependency can help to combat the acute problems of customisation and interoperability that arise in educational technology. The well-conceived use of dependency in model-building enables a degree of model interconnection and interoperability that is unprecedented in traditional software development. Dependency-based models can also potentially serve as a single source from which programs for specific target platforms and special learner needs can be derived by a process of specialisation and translation (Beynon et al, 1988; Beynon & Cartwright, 1997), as has been demonstrated in ground-breaking new approaches to cross-platform broadcasting for interactive digital television (BBC R&D Cross-platform Interactive Television: http://www.bbc.co.uk/rd/pubs/brochures/ibc2003/bbc-rd-cross-platform-interactive-tv.pdf).

Wider implications

The potential wider implications of our new conception of computing technology can be conveniently reviewed under three different themes identified by Price et al. (2005):

Political change - The insidious influence of machine metaphors for human cognition on the conception of human-centred activities such as commerce and education is one of the most unhappy side-effects of the insight into rule-based systems gained during the twentieth century. A surprising side-light on this development, as remarked by Emil Post (1965/1941), was that formalization in mathematics continued to attract the highest level of prestige and interest despite fundamental results by Gödel and Turing that demonstrated their inherent limitations. Writing in 1941, Post called for a reappraisal of the priorities in mathematical research, stressing "that mathematical thinking is, and must

be, essentially creative" and expressing the expectation - and hope - that mathematics would "reverse the entire axiomatic trend of the late nineteenth and early twentieth centuries, with a return to meaning and truth". Contemporary education policies, both in schools and higher education, seem to promote a culture with many characteristics of formal systems, and invite a similar critique. The new conception of computing technology proposed in this paper can be interpreted as lending support to an alternative culture, in which less emphasis is placed upon stereotypes both where the academic curriculum and the evaluation of performance are concerned (cf. Beynon & Russ, 2004).

Technology adoption - This paper argues that the conception of computing technology that underpinned the first attempts to exploit computers in schools, and that has dominated subsequent developments in educational technology (with the notable exception of spreadsheets), is ill-suited to supporting active learning. In particular, it involves modes of software development that make the dynamic and subtle changes to the requirement demanded in educational applications hard even for the professional to implement. The proposal outlined above aims to give teachers the levels of autonomy in developing educational software to which many originally aspired. Specifically, it identifies the need for educational software development to be a do-it-yourself activity that is intimately bound up with domain learning and understanding, and that ideally can be conducted incidentally in conjunction with other teaching preparation - that is, without requiring the extremely high levels of expertise, concentration and commitment currently demanded of the professional developer (Beynon, 2001).

Models of teaching and learning - Educational software is conceived or customised by the teacher; designed, implemented and maintained by the developer; used by the learner. These activities all take place in quite different contexts, and invoke sharply distinct perspectives and interfaces, each deliberately crafted to serve a specialist function. Such a disjunction of teacher, learner and developer activities results inevitably from a conception of computing that is predicated on separating the core formal computation from the rich penumbra of informal but essential interpretative activities that engage with meaning. Computer science and educational technology alike have adapted to this dualistic outlook, but such duality is not respected in practical computing or active learning, where construction and meaning are inextricably linked. To question the status of this duality is not to deny the significance of the classification of pedagogical activities proposed by Shaffer and Kaput (1999), after Merlin Donald (1991), that is further discussed in Riley (2005a). Computing technology will no doubt initiate a new stage in the evolution of human cognition and culture, but it is its dualistic conception that informs the notion of a virtual culture that "has its roots in the need to conduct more and more complex processing of information using formal mathematical systems" (Shaffer & Kaput, 1999). The conception of computing technology proposed in this paper has elsewhere been discussed in conjunction with a classification of learning activities that likewise ranges from what are perceived to be the most primitive to the most sophisticated modes of human interaction and behaviour (cf. the Experiential Framework for Learning (Beynon & Roe, 2006; Beynon, 2006a)). The significant difference is that, in that context, these learning activities are perceived as part of a holistic vision for learning in which modes of interaction at many levels, using a wide variety of supporting human and computing technologies, are typically concurrently invoked. It is such a vision that seems most appropriate if computing technology is to liberate the qualities of "wild imagination, passion, being close to nature, and believing in magic" that Papert has in mind.

Acknowledgements

I am indebted to Martin Oliver, Sara Price, Diana Laurillard and Carol Strohecker for stimulating me to write this paper, to Russell Boyatt, Charles Care, Eric Chan, Antony Harfield, Karl King and Ashley Ward for their contributions to the Kaleidoscope Showcase, and to Dave Riley and Steve Russ for valuable ideas and references. I also appreciate the most thoughtful and helpful suggestions of two anonymous reviewers, and the invaluable assistance of the editors in making revisions.

References

Baker, J. E., & Sugden, S. J. (2003). Spreadsheets in Education - the First 25 Years, e-Journal.*Spreadsheets in Education, 1* (1), 18-43

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*, Addison-Wesley.

Beynon, W. M. (2001). Liberating the Computer Arts, invited paper at 1st International Conference on Digital and Academic Liberty of Information (DALI), University of Aizu, Japan.

Beynon, W. M. (2005a). Computational Support for Realism in Virtual Environments. *Paper presented at the 11th International Conference on Human-Computer Interaction (HCII 2005)*, 22-27 July 2005, Las Vegas, NV, USA.

Beynon, W. M. (2005b). Radical Empiricism, Empirical Modelling and the nature of knowing. *Pragmatics and Cognition, 13* (3), 615-646.

Beynon, W. M. (2006a). Towards Technology for Learning in a Developing World. *Paper presented at the 4th IEEE International Workshop on Technology for Education in Developing Countries*, July 10-12, 2006, Iringa, Tanzania.

Beynon, W. M. (2006b). Mathematics and Music - Models and Morals. *Paper presented at the Bridges Conference: Mathematical Connectionsin Art, Music, and Science*, August 4–9, 2006, London, UK.

Beynon, W. M., & Cartwright, R. (1997) Empirical Modelling Principles in Application Development for the Disabled. *Proceedings of the IEE Colloquium: Computers in the Service of Mankind: Helping the Disabled*, IEE Digest No 97/117, 4/1-4/3.

Beynon, W. M., Halstead, K. S. H., & Russ, S. B. (1988). Definitions for the specification of educational software, *Report prepared for the Micro Electronics Support Unit*, UK Dept of Education & Science, University of Warwick.

Beynon, W. M., & Roe, C. P. (2006). Enriching Computer Support for Constructionism. In Alkhalifa, E. (Ed.), *Cognitively Informed Systems: Utilizing Practical Approaches to Enrich Information Presentation and Transfer*, Hershey, PA, USA: Idea Group Publishing, 209-233.

Beynon, W. M., Norris, M. T., Russ, S. B., Slade, M. D., Yung, Y. P., & Yung, Y. W. (1989). Software construction using definitions: an illustrative example. *CS-RR-147*, Warwick, UK: University of Warwick.

Beynon, W. M., & Russ, S. B. (2004). Redressing the past: liberating computing as an experimental science. *CS-RR-421*, Warwick, UK: University of Warwick, retrieved December 16, 2006, from http://www.nesc.ac.uk/esi/events/Grand_Challenges/gcconf04/submissions/26.pdf.

Beynon, W. M., Russ, S. B., & McCarty, W. (2006). Human Computing: Modelling with Meaning. *Literary and Linguistic Computing, 21* (2), 141-157.

Brödner, P. (1995). The Two Cultures in Engineering. In Goranzon, B. (Ed.), *Skill, Technology and Enlightenment*, Berlin: Springer-Verlag, 249-260.

Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering* (20th Ed.), Addison-Wesley.

Bruner, J. (1990). *Acts of Meaning: Four Lectures on Mind and Culture* (Jerusalem-Harvard Lectures), Cambridge, MA: Harvard University Press.

Cantwell-Smith, B. (2002). The Foundations of Computing. In M. Scheutz (Ed.), *Computationalism: New Directions*, MA: MIT Press, 23-58.

Carter, B. (2004). *Digital Asset Management and Processing for Games*, Rockland, MA, USA: Charles River Media.

Doerr, H. M. (1996). STELLA Ten Years Later: A Review of the Literature. *International Journal of Computers for Mathematical Learning, 1* (2), 201-224.

Donald, M. (1991). *Origins of the Modern Mind: Three Stages in the Evolution of Culture and Cognition*, Cambridge, MA: Harvard University Press.

Gooding, D. (1990). *Experiment and the Making of Meaning: Human Agency in Scientific Observation*, Berlin: Kluwer.

James, W. (1912). *Essays in Radical Empiricism* (Reprinted from the original 1912 edition by Longmans, Green and Co, New York, 1996), London: Bison Books.

Kalas, I., & Blaho, A. (2001). Imagine... a new generation of Logo: programmable pictures. *Paper presented at the 16th IFIP World Computer Congress (WCC2000)*, August 21-25, 2000, Beijing, China.

Kaleidoscope (n.d.) Concepts and methods for exploring the future of learning with digital technologies. *Technology-enhanced learning, Fact sheets of research projects under FP6*, European Commission.

Lansdown, J. (1987). Graphics, Design and Artificial Intelligence. *NATO Advanced Study Institute Series F: 40*, 1153-1174.

Lovegrove, G., & Round, A. (2005). IT Professionals in Education: Increasing the Supply. *Report on the North-East regional meeting on the HEFCE-funded initiative held at St James Park*, November 8, 2005, Newcastle-upon-Tyne.

Manson, P. (2004). Introductory talk at Information day on Technology-enhanced learning Workprogramme. *Information Society Technologies*, November 29, 2004, European Commission.

Mason, J. (1987). Representing representing: Notes Following the Conference. In Janvier, C. (Ed.), *Problems of Representation in Teaching and Learning Mathematics*, Hillsdale, NJ: Lawrence Erlbaum, 207-214.

McDermott, D. (1987). A Critique of Pure Reason. *Computational Intelligence, 3*, 151-160.

Myers, B. A., McDaniel, R. G., Miller, R. C., Ferrency, A. S., Faulring, A., Kyle, B. D., Mickish, A., Klimovitski, A., & Doane, P. (1997). The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Transactions on Software Engineering, 23* (6), 346-365.

Nardi, B. A. (1993). *A small matter of programming: Perspectives on End User computing*, Cambridge, MA: MIT Press.

Oliver, M., & Price, S. (2005). Establishing the impact of technology on roles and practices in Higher Education. *Paper presented at the Technology and Change in Educational Practice Conference*, October 5-6, London, UK.

Papadimitriou, C. H. (2003). MythematiCS: in praise of storytelling in the teaching of computer science and mathematics. *ACM SIGCSE Bulletin, 35* (4), 7-9.

Papert, S., & Strohecker, C. (2005). Catalyzing debate about fundamental change in education. *Paper presented at the Technology and Change in Educational Practice Conference*, October 5-6, London, UK.

Perry, M. (2001). *Automate Dependency Tracking, Parts 1, 2 & 3*, JavaWorld.com, Aug-Oct 2001, retrieved December 16, 2006, from http://www.javaworld.com/javaworld/jw-08-2001/jw-0817-automatic.html.

Post, E. (1965). Absolutely unsolvable problems and relatively undecidable propositions (Unpublished manuscript originally submitted to a mathematical periodical in 1941). In Davis, M., *The Undecidable*, Raven Press Books.

Repenning, A. (1993). AgentSheets: A Tool for Building Domain-Oriented, Dynamic, Visual Environments. *PhD thesis*, University of Colorado at Boulder, USA.

Ridley, M. J. (2003). Database Systems or Database Theory - or 'Why Don't You Teach Oracle?' *Paper presented at the LTSN-ICS Workshop on Teaching Learning and Assessment in Databases (TLAD)*, July 14, 2003, Coventry, UK.

Riley, D. (2005a). Learning by modelling: Reflections on rhythms and roles of instruction. *Paper presented at the Technology and Change in Educational Practice Conference*, October 5-6, London, UK.

Riley, D. (2005b). Learning by making sense of computer models. *Unpublished slides from seminar presented in Computer Science at the University of Warwick*, November 25, 2005, Coventry, UK.

Roe, C. P. (2004). Computers for learning: An Empirical Modelling perspective. *PhD Thesis*, Computer Science, Warwick University, UK.

Roe, C. P. (2005). On Dependency-based Programming for Education. *Paper presented at the Symposium at Centre for New Technologies Research in Education*, April 5, 2005, University of Warwick, UK.

Shaffer, D. W., & Kaput, J. K. (1999). Mathematics and virtual culture: an evolutionary perspective on technology and mathematics education. *Educational Studies in Mathematics, 37* (2), 97-119.

Small, J. (2001). *The Analogue Alternative* (Chapter 7), London: Routledge.

Soloway, E., Pryor, W., Krajcik, J., Jackson, S., Stratford, S. J., Wisnudel, M., Klein, J. (n.d.). *ScienceWare's Model-It: Technology to Support Authentic Science Enquiry*, Miscellaneous papers, Center for Highly Interactive Classrooms, Curricula & Computing in Education, University of Michigan, retrieved December 16, 2006, from http://hi-ce.org/research/index.html.

Spence, R. (2000). *Information Visualization*, New York: ACM Press.

Turner, M. (1996). *The Literary Mind*, Oxford University Press.

Vincenti, W. G. (1993). *What engineers know and how they know it: analytical studies from aeronautical history*, Baltimore, MD: Johns Hopkins University Press.

Warboys, B., Kawalek, P., Robertson, I., & Greenwood, M. (1999). *Business Information Systems: a Process Approach*, New York: McGraw-Hill.