# Constructivist Computing - a New Paradigm for Open and Distance Learning?

**Meurig Beynon**

Computer Science
University of Warwick
W.M.Beynon@warwick.ac.uk

**Hui Zhu**

Computer Science
University of Warwick
hui.zhu@warwick.ac.uk

Subtheme 6:
The supervision environment and strategy for creative, open and flexible learning

**Abstract:** Constructivist computing is a new conceptual framework for computing-in-the-wild. It is based on making interactive artefacts called *construals* using the principles of Empirical Modelling (EM). Making construals is a creative, experimental and social activity that promotes independent thinking in a constructionist spirit. We discuss the merits of applying constructivist computing principles to open and distance learning with reference to emerging EM tools to support digital education and the *Manifesto for Teaching Online* devised by teachers and researchers at Edinburgh University.

**Keywords**: constructionism, Empirical Modelling, web-based environments, self-paced, collaborative, flexible learning

This paper discusses open and distance learning in the light of our experience of teaching an unconventional course at the University of Warwick over the last twenty years. Empirical Modelling (EM) is a conceptual approach to computing-in-the-wild that goes beyond what can be characterised in a traditional way as "computational thinking" [2]. In its current form, our "Introduction to Empirical Modelling (IEM)" course is taught face-to-face through 20 hours of sessions, half of which are devoted to classroom sessions and half to practical laboratories. The tools and products of EM are the result of work largely carried out by computer science students at Warwick, both undergraduate and postgraduate, over the last twenty five years. Though the course itself has not yet been deployed online, it makes extensive use of online resources that - in addition to the traditional lecture slides, course documentation and background research papers - include web-enabled versions of our primary interpreter, interactive models demonstrated during classroom sessions, many archived student projects, and webpages that serve to integrate these under themes such as "EM for Educational Technology" and "EM for Concurrency" [6].

Our discussion will refer to an interesting recent contribution to thinking about open and distance learning - the *Manifesto for Teaching Online (MTO)* [7], produced at the University of Edinburgh in 2011. The MTO is notable for the positive way in which it presents online education not as the poor relation of face-to-face (F2F) education but potentially as "the privileged mode" [7]. The claims in the MTO are informed by practical experience gained from teaching on the MSc in Digital Education (MDE) at Edinburgh.

There are clear points of similarity between the IEM course and the MDE programme [10]. The average number of students completing our course each year has been about twenty. The principal coursework component for IEM involves carrying out a modelling study that is discussed in a paper suitable for a conference-style presentation. The submissions make up the annual "Warwick Electronic Bulletin on Empirical Modelling (WEB-EM)", now in its ninth year. Students choose the theme of their WEB-EM modelling study, and can adjust the weight given to the practical and written components to suit their purpose throughout the exercise. Their initial submission takes the form of brief details of a modelling study and an associated short abstract on which we give feedback.

In managing the WEB-EM assignment, we face challenges similar to those faced in the MDE. These relate to teaching in a setting in which active learning is encouraged. They will become more acute and topical as we move towards our goal of teaching IEM as an online course.

EM is centrally concerned with constructing interactive artefacts that serve a role in sense-making and learning. Such artefacts – so-called EM construals - invite free interaction and interpretation that is stimulated by the immediate experience they offer to the human agent. It is not appropriate in general to view construals as traditional programs that have a functional interpretation. They more closely resemble spreadsheets, in that they are meaningful only in their relation to specific external situations (e.g. managing my personal finances, examining a class of students, or analysing experimental data) to which they refer. The primitive ingredients of a construal are the *observables* - entities perceived to have an identity and current status - and the *dependencies*, which express expectations about how changing one observable affects others. The understanding that is associated with a construal is manifest in familiarity with patterns of interaction and interpretation that can have a highly personal character. Observables may refer to states of mind as well as to physical objects. The choice of observables reflects the knowledge of the modeller (e.g. what is involved in recognising a *matrix* in linear algebra). There is also a contextual aspect to making construals that reflects the modeller's current focus of attention and intentions. For instance, we attend to the state of a spreadsheet quite differently if our aim is to make its content more accessible by enabling the user to highlight selected cells, rows and columns in a dynamic fashion. Figure 1 depicts the way in which - at each moment in the making of a construal - the current status of the construal and its referent relate to the model-builder's understanding and the context for the model-building. The **construal**, its **referent**, the modeller's **understanding**, and the **context** are all subject to evolve moment-by-moment as the construction proceeds.

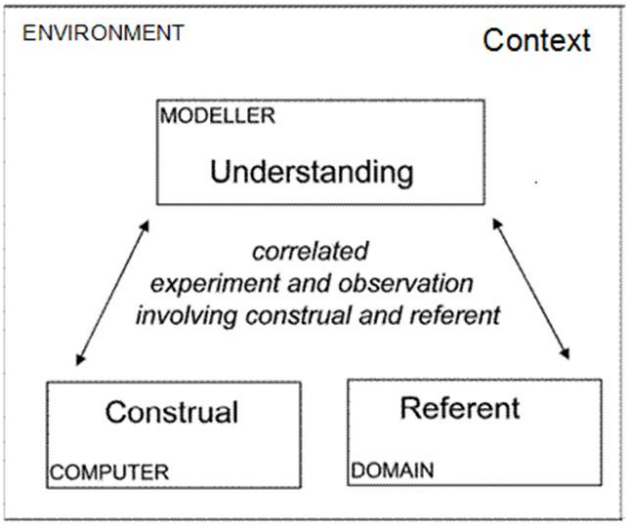


Figure 1: Making construals in EM

IEM already exhibits many of the characteristics of an online course identified in the MTO. From the outset, EM teaching has emphasised the essential need to engage in making construals. Traditional computer science promotes the idea that effective computing artefacts can only be built with the benefit of a functional specification that - if not actually preconceived - is subject to iterative revision. EM, in contrast, advocates experimental construction of a construal where the emphasis is upon identifying and imitating the patterns of observation, dependency and agency that we encounter in its referent. This foregrounds the role of computing technology as a source of experience of unprecedented richness and subtlety. In this respect IEM is a course that is "born digital" (cf. [7]). Such is the importance of practice that we are obliged to take *making a construal* seriously as an adjunct to academic writing (cf. "text is being toppled as the only mode that matters in academic writing"). A construal potentially gives a teacher access to knowledge about the learner's thinking that is absent in traditional assessment. In assessing the quality of a construal, we are obliged to exercise imagination and make pragmatic judgements (cf. "working harder"). It is vital that our preliminary feedback "can be digested, worked with, created from", since it can have critical consequences for the subsequent development. This is in keeping with what one of the authors of the MTO, Clara O'Shea [10], refers to as "feed forward". Some students select topics in which their knowledge of the referent surpasses our own - obliging us to discover and retrace their interactions whilst we adopt the role of learners. This challenges us to establish whether we have discovered all the relevant modes of interaction, and whether our trace does full justice to what is embodied in the construal (cf. "creative crisis"). Such a construal is then effective to the extent that it facilitates the *teacher's* learning.

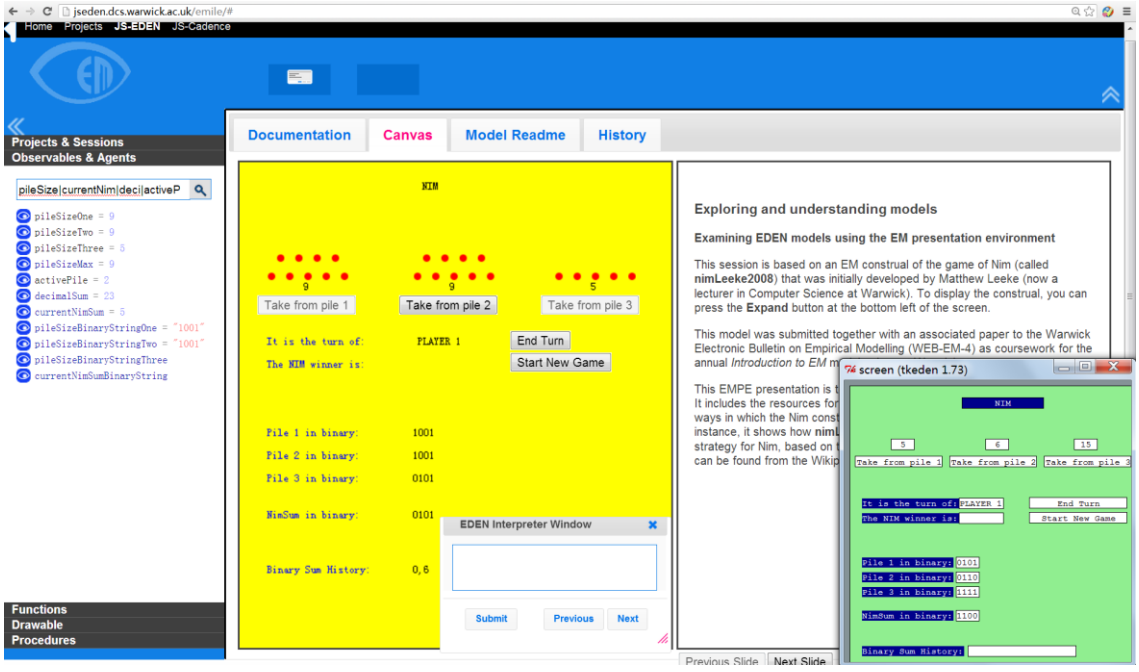


Figure 2: EM construals of the game of Nim

By way of an illustrative example, Figure 2 above depicts two EM construals of the game of Nim [11]. Nim is a simple game in which two players take turns to remove stones from an arbitrary number of piles. Each player is restricted to taking stones from just one of the piles on each turn, and the winner is the player who removes the last stone.

The initial source for this construal (nimLeeke2008) was a submission to WEB-EM-4 by Matthew Leeke, now a colleague in Computer Science at Warwick. Leeke's original version of the construal is displayed as an inset in the bottom right. The primary image in Figure 2 shows a new version of the construal generated using JS-EDEN, the latest web-enabled prototype of our primary EM tool. This is displayed within a presentation environment (the "EMPE") developed specifically for the purpose of presenting EM construals.

Superficially, Leeke's Nim construal resembles a traditional computer program to play Nim with three piles of stones. The buttons on the interface allow the players to take stones from piles in accordance with the rules. The binary representations of the numbers of stones in each pile can be used to compute the "NimSum" which underlies the winning strategy for Nim and these are also a prominent feature of the display. Making construals is quite different from conventional programming however, and this is critical where the potential for creative, open and flexible interaction is concerned.

The source for the construal and the accompanying slide presentation is a family of definitions of observables. Selected observables are displayed in the panel on the left-hand side of Figure 2. The construal remains live to modification at all times – a new definition for an observable can either be submitted via the Eden Interpreter Window, or modified by clicking on the observable name in the left-hand panel to bring up a textbox in which it can be edited. Conceptually, the entire state of the Canvas display in Figure 2 is captured in the observables and dependencies specified in these definitions. These observables include all the information required to specify the Nim display (e.g. the sizes of the piles, their binary representations, the layout of the buttons and display components, the status of buttons, the contents of textboxes and the colours of the display), to determine the current state of a game in play (e.g. whose turn it is, which pile is currently in play, whether the winner has been decided) as well as the current status of the slides in the presentation (e.g. how many slides there are, the contents of each slide, the slide currently being displayed, the way in which the slides are ordered). Simple examples of dependencies serve to link the number of stones in a pile to its binary representation, to change the colour of the rubrick on the "Take from pile 1" button according to whether or not pile 1 is in play, and to ensure that the option of taking a stone from an empty pile is unavailable.

The construal is built so that all the meaningful activities that take place in playing a game of Nim can be represented as changes of state that involve redefining one or more observables. For instance, when a player initiates a turn by taking a stone from Pile 1, the value of the observable **activePile** is set to 1 and the observable **pileSizeOne** is redefined so that it has the explicit value

**pileSizeOne-1**. Because these redefinitions are made within the context of the current family of definitions, the values of other observables are changed as if in one and the same action. For instance, the colour of the foreground text on the 'Take from Pile 1' button is changed, the visual and binary representations of the number of stones in Pile 1 are updated, as is the NimSum. Conceiving all state transitions as associated with families of redefinitions makes it possible to model exceptionally rich modes of interaction with the construal. In particular, we can conceive emulating a program for playing Nim in these terms.

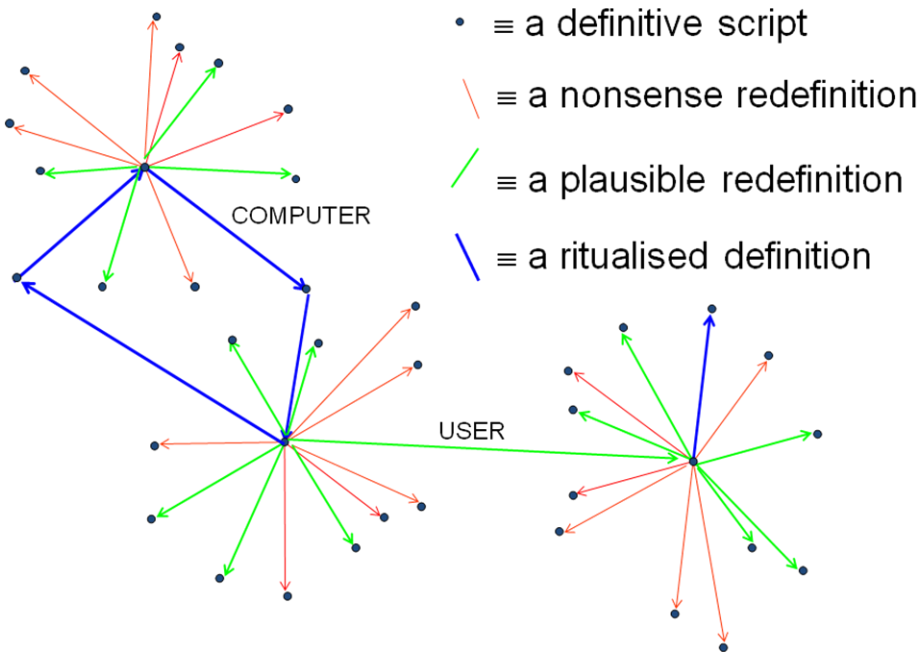


Figure 3: The semantics of construals

A visual aid for our mental model is helpful here. Following the conventions of automata theory, we can represent the current state (as determined by the current family of definitions of observables) by a single node, and draw a directed edge from one state to another to indicate a transition between states (as determined by any family of redefinitions that might be introduced). The virtual network of states and transitions that is set up in this way contains far too many edges and nodes to be drawn explicitly, but can be seen as the medium within which patterns of interaction and interpretation of interest can be traced. Figure 3 depicts the generic way in which a conventional program can be conceived in these terms. There are three ingredients in applying a construal in this way: engineering the states within which the agency of the user and the computer operate; crafting the behaviours which these agents then play out; and projecting meanings on to the agent actions.

In drawing the network of nodes and edges in Figure 3, we are asserting that certain families of observables and dependencies are experienced as concurrent in the domain of interest. With reference to the Nim construal in Figure 2, we might apply a strict criterion to determine which definitive scripts

correspond to nodes of interest in Figure 3. For instance, the horizontal green edge labelled USER might be associated with the redefinition of the mouse status that occurs when the "Take from pile 1" button is pressed at the beginning of a turn. In this case, the node at the left would correspond to the state immediately prior to the selection of Pile 1, where the observable **activePile** has the value 0. The node on the right would differ only in as much as the "mouse status" observable has a different value. This change in mouse status would in turn be the signal for the computer to make a transition such as is depicted by the blue edge pointing vertically upwards. This transition would be associated with the concurrently executed pair of redefinitions of the observables **activePile** and **pileSizeOne** that was discussed previously. Elaborating this process of attaching the meaning of the Nim playing program to the network of nodes and edges is conceptually straightforward, and generates the kind of state machine model that might feature in a program specification.

Many other interpretations are of course possible. Suppose that instead of repeatedly pressing a button to take stones from Pile 1, a player was able to select pile 1 then specify the number of stones to be removed from it via a drop down menu. The computer response in this case might trace out the sequence of three blue edges on the left-hand side of Figure 3, where each edge links a definitive script to one in which **pileSizeOne** has been decremented and a counter observable has been incremented. Alternatively, it might be that another element of skill was introduced into the game, whereby the computer cycled through states in which the options "take from pile 1", "take from pile 2", "take from pile 3", "take from any pile" were available, and a player selected the active pile at the start of each turn by interrupting this cycle at the most appropriate point. In that case, the sequence of blue edges might correspond to the four steps in the cycling process. Yet other possibilities involve changes to observables that remain invariant in traditional variants of Nim. It might be for instance that the radius of stones in one pile might diminish every time that another pile became active, and might then disappear entirely.

The elaboration of further examples of this nature is left to the reader's imagination. The alternative semantic possibilities are not of themselves the principal subject of interest. What matters is that making a construal opens up such a profusion of possible interpretations, stimulating the model-builder's imagination and creativity. Yet more important is the fact that the making of a construal is an open-ended activity that resembles organic growth rather than building to a specification.

In Figure 3, edges are colour coded according to whether they represent transitions that are so routine that they can be "ritualised" and executed automatically, transitions that are not within the normal semantic scope but *may* have a plausible interpretation, and transitions that are nonsensical. This classification is pragmatic and depends on the motivation for making the construal. If our objective in building a construal is to specify a computer environment to support the playing of Nim, the redefinitions that remove a stone

from a pile are primitive actions with entirely predictable effects. The modeller might establish the appropriate observables and dependencies needed to render such actions faithfully in the early stages of making the construal, framing them as definitions. In due course, once the modeller has rehearsed them manually enough to be satisfied that they have the appropriate effect, their interpretation takes on a ritual character, and execution may be automated. By contrast, a redefinition that transferred a stone from one pile to another would be nonsensical, whilst a redefinition that placed one stone on top of another might be plausible if we wished to elaborate the game by taxing the players' memory of the number of stones in each pile. It is when we develop a construal in a more exploratory fashion - perhaps with a view to devising an interesting variant of Nim incorporating the kind of unconventional features discussed in the previous paragraph - that the pragmatic nature of the characterisation of possible transitions is seen most clearly. It is then apparent that what is deemed plausible and nonsensical is itself shaped by the way in which the human exercises the emerging construal, and that it may not be self-evident from the outset which transitions will acquire ritualised status.

It is instructive to compare and contrast the JS-EDEN environment in Figure 2 with the *Learnable Programming* environment demonstrated by Bret Victor [12]. In Victor's environment, as in the JS-EDEN environment, a webpage is split into two panels so that the source of the RH panel is displayed in the LH panel. A learner can edit the JScript source of the webpage in the LH panel directly, and get immediate feedback as to how this affects the display.

In [12], Victor makes the case that the direct dependency between "editing the code" and "observing the changes to the webpage" encourages the learner to make a crucial conceptual shift from "thinking like a machine" to "thinking like a human". The learner programmer comes to see the programming text in an instrumental light, not as a sequence of operations to be carried out by a machine, but as a significant state-changing gesture. As Victor points out [12], this is analogous to the musical performer who internalises instrumental technique to such as a degree that she is only conscious of the musical line.

Our experience of EM strongly endorses the idea that the immediate feedback associated with synchronising state can establish a radically different conceptual frame. Though the process of editing the webpage source is more indirect in the JS-EDEN environment, the effect of maintaining dependencies between states is in other respects even more significant. This is because the correspondence between lines of programming code and their computational effect is notoriously difficult to comprehend, and can be challenging to imagine even for the expert programmer. Modifying a program is changing a recipe for action, and its significance can only be understood in general by imagining the side-effects this has on state to be played out by the machine. In EM, by comparison, there is - at any rate in the construal to which we aspire! - a direct correspondence between observables and dependencies in the construal and observables and dependencies in the referent (cf. Figure 1 and the qualities of

spreadsheets). Whilst this does not dispel the need for computational thinking in using tools such as JS-EDEN, it confines this activity to the role of specifying those special-purpose functions that have to be crafted by the modeller in order to express novel kinds of dependency.

Practising EM revises the mental models that learners bring to engaging with the computer. In place of procedural thinking directed at automating action to meet functional goals, the learner attends to what connections are encountered in immediate experience of the application domain. For instance, each of the sample dependencies in the Nim construal listed above establishes a connection that is a vital ingredient of what it means to be playing Nim with the winning strategy in mind. When used to formulate dependencies, the functional relations that traditionally play centre stage take on a different role - they express expectations about how making a change to one element of our environment will directly affect another. Informal evidence, both from the EM research programme and from studies of spreadsheet use in education such as [13], indicates that knowledge of this nature is well-matched to our commonplace conception of the worlds in which we interact.

The WEB-EM coursework illustrates several of the qualities of online education to which the MTO refers. Students are free to make artefacts that reflect their personal knowledge and experience, and have access to myriad resources beyond those that have been supplied by the teacher. This activity is in the spirit of Papert's constructionism [4], and represents a form of active learning. But as Clara O'Shea observes in her accidental MTO remix [8]: "different digital environments constrain and create different possibilities for engagement". As she remarks elsewhere [9]: "Neither learner centrism and teacher centrism will do – they are polarities on a continuum, the high arcs of a pendulum swing. Neither encompasses what happens in learning." This begs the question of which kinds of digital environment give appropriate support for engagement of a constructionist variety.

Victor's concept of Learnable Programming is in the same spirit as EM thinking. In [12], Victor emphasises the affinity between his approach and Papert's thinking [4]. From an EM perspective, attempting to link constructionism with traditional styles of programming, no matter what paradigm or interface is adopted, is problematic [3]. Traditional program development and learning about the domain are ill-matched, as becomes apparent when constructing software systems that entail "radical design". In our view, the qualities of construction by computer can only be fully realised by invoking a more radical reconceptualisation of computing.

Such considerations lead us to regard alternative approaches to resolving problems related to digital education in a similar light. In particular, though some of the claims being made in the MTO are endorsed by our experience in IEM, we are sceptical about the extent to which they are justified in respect of the techniques for digital construction and communication that have so far been applied in open and distance learning (cf. [1]).

A central concern in the MTO is that courses that empower the individual subvert the traditional role of the teacher. The students who flourish on IEM are those who can handle the exceptionally rich open-ended style of development that making construals affords. Because of the strong element of apprenticeship in the lab work for IEM, those who engage with the process must necessarily develop a relationship of trust with their tutors. In their WEB-EM coursework, they negotiate their own personal assignment, and – even when their work involves re-use – their sense of ownership is deep. Plagiarism is only an issue if one student's re-use of someone else's construal is so superficial that they do not create their own personal web of interaction and interpretation (cf. "A routine of plagiarism detection structures in a relation of distrust" [7]).

The problem for the teacher is to encourage engagement and participate in the student work whilst striving to maintain the objectivity that is essential for assessment. The student's task, as described above, involves building an artifact in parallel with building an understanding that is embodied in skilful interaction and interpretation with the emerging construal. The rehearsal of this understanding is initially a matter of private performance, but can later be tested and potentially communicated to others through demonstration. The teacher's role in this process is at first that of a listener who poses questions that elicit further elaboration, and later that of an evaluator who assesses the authenticity of the student's experience. Both roles demand the teacher's active participation (cf. "online teaching should not be downgraded to 'facilitation'" [7]).

The nature of EM construals is crucial in enabling this level of engagement between teacher and learner. As observed in [3], a critical problem in supporting constructionism in traditional programming environments, such as the myriad variants of Logo, is that the roles of developer, teacher and learner require conceptually totally different interfaces to the object under construction – interfaces respectively suited for software development, requirements specification, and use. In making construals, there is no such sharp conceptual distinction: the agency involved in building, shaping and exercising the construal is similar in nature, and differs only according to the skill and expertise that the human agent brings to the role. The effect is to promote ongoing collaborative development such as is illustrated in the Nim construal itself. First devised by Leeke, embedded in the EM presentation environment by the first author, then adapted by the second author to the web-enabled form displayed in Figure 2, the Nim construal has also served as the focus for a construal comprehension exercise for MSc students on IEM, and a session at a workshop on EM [6].

Making EM construals has two aspects: it is an individual activity that promotes creative, open and flexible learning; it also has a social dimension concerned with exploring the extent to which our understanding can be communicated and shared. That latter dimension is brought out when construals are demonstrated either informally, or by using the EMPE. In effect, the learner exposes key interactions that testify to their understanding, and invites others to follow their example. It is through such public performance that a degree of

"objectivity" can be crafted. This is in keeping with the idea of the objective world as a product of *construction* – hence "constructivist computing".

In the spirit of the MTO [7], the IEM course can be regarded as “philosophy and belief in action”. The strong affinity between EM and the radical empiricism of William James [2] provides an overall orientation that can maintain integrity whilst admitting plurality. Such an authoritative point of reference is invaluable in managing the “tension between randomness and intentionality” that characterises the course processes [7]. This tension affects both teachers, who must be open to a wide variety of perspectives, and students, some of whom find the lack of clearly specified structures and goals uncongenial. Amongst the most outstanding students are those who challenge the presumptions that inform IEM, and for whom the opportunity to turn their attention from making construals to critiquing EM principles and tools is vital (“assessment strategies … designed to allow for the possibility of resistance” [7]). This was most dramatically illustrated in the WEB-EM submission of Nicolas Pope, who deployed his own software in place of the established EM interpreter in his modelling study. Pope subsequently gained a doctorate for research that has laid the foundation for new EM tools that have also been exercised in the IEM course [5].

The boldest claims in the MTO concern the significance of place: "Distance is a positive principle … Place is differently, not less, important online. … By redefining connection we find we can make eye contact online." Our claims for EM are as bold, but different – and broader. One interpretation of Figure 1 is that EM *realises* its own external context. EM can be seen as “redefining connection” to achieve more than ‘eye contact’. And the lineage that connects construal with Faraday’s experimental practices [2] makes it plausible that in general more can be learnt from EM principles than can be learnt online.

REFERENCES

[1] W. M. Beynon, “Computing technology for learning - in need of a radical new conception,” Journal of Educational Technology and Society, vol. 10, no. 1, pp. 94–106, 2007.

[2] M. Beynon, “Modelling with experience: construal and construction for software,” in Ways of Thinking, Ways of Seeing, C. Bissell and C. Dillon, Eds. Springer-Verlag, Jan. 2012, 197–228.

[3] M. Beynon and A. Harfield, “Constructionism through Construal by Computer,” in Proc. Constructionism 2010. The American University of Paris, 2010.

[4 ] S. Papert, Mindstorms: Children, Computers and powerful ideas. Basic Books, 1980.

[5] N. W. Pope, “Supporting the migration from construal to program: Rethinking software development,” Ph.D. dissertation, Dept of Computer Science, University of Warwick, Dec. 2011.

[6] EM Website http://www.dcs.warwick.ac.uk/modelling

[7] Manifesto for Teaching Online http://www.swop.education.ed.ac.uk/manifesto.html

[8] http://claraoshea.wordpress.com/2012/03/01/my-accidental-manifesto-remix/

[9] http://claraoshea.wordpress.com/2012/02/29/why-is-it-a-manifesto-for-teaching-online/

[10] http://www.youtube.com/watch?v=SkHxG9ZU_E8

[11] "Nim" from Wikipedia https://en.wikipedia.org/wiki/Nim

[12] Bret Victor, "Learnable Programming" http://worrydream.com/LearnableProgramming

[13] “Spreadsheets in Education”, http://epublications.bond.edu.au/ejsie/