Web support for e-learning: a constructivist computing approach

Timothy Monks*, Nicolas Pope*, Richard Myers*, Antony Harfield†*, Meurig Beynon* and Hui Zhu*
*The Empirical Modelling Research Group, University of Warwick, UK and †CS and IT, Naresuan University, Thailand

***Abstract*—The radical implications of emerging web technology for education motivate a reappraisal of the conceptual framework that surrounds computing. Empirical Modelling (EM) has been developed as a way of thinking about computing that emphasises the role the computer plays in supporting learning in a constructivist spirit. EM research on principles and tools over the last 25 years has established proof-of-concept for varieties of technology-enhanced learning particularly relevant to current and emerging educational environments. This paper reviews the progress towards developing web-based tools to support EM that can realise its full potential as a vehicle for learning.**

I. Introduction

Without doubt, the advent of the Internet has changed the educational landscape dramatically. Digital networks have promoted a culture such as was envisaged by Papert in 1980 [1], where users themselves create content, can share their knowledge and expertise, gain access to cultural artefacts or remix existing material. From a computer science perspective, the most interesting aspects of this culture relate to collaborative activities that involve more than the use of a computer simply as a technology for supporting otherwise conventional communication. But notwithstanding Papert's vision for constructionism [1], there are many problematic conceptual and practical aspects to exploiting the full potential for collaborative construction of computing artefacts as an educational medium. The popularity of Scratch [2] for instance seems to owe something to the radical simplification of the programming component, whereby the scope for user-specified procedures is limited, but hooks into a richer multi-media environment are supplied. By comparison, environments such as Net Logo [3] offer more sophisticated scope for end-user programming, but in a more tightly constrained semantic framework. And though the incorporation of 'Mods' in game environments such as World of Warcraft [4], [5] take the practice of user-development yet further in computing terms, this degree of learner participation in development is scarcely represented in technology-enhanced learning.

In this paper, we argue that the challenge of liberating learning through the collaborative development of artefacts can only be addressed fully through a radical reconceptualisation of computing itself [6], [7]. To this end, we propose Empirical Modelling (EM) [8] as a conceptual framework for computing that embraces the larger role that computing-related technologies play in the sense-making activities that surround traditional computer use [9]. The central concept in EM is that of using computing technology to make *construals* rather than to write programs. A construal is an interactive artefact that metaphorically exhibits the same states and transitions that are observed within its referent in the learning domain. An EM construal is developed incrementally through open-ended interaction in such a way that it embodies observables, dependencies and agency that are characteristic of its referent. This activity has the qualities of *construction*, as described by Latour in [10] (cf. [11]) - and is hence referred to as 'constructivist computing'. This paper discusses a vision for a computing environment to support constructivist computing.

Previous work on EM has established proof-of-concept in respect of support for primitive learning [12], exploratory experiment [13], [14], and distributed participatory design [15]. Tool development has a critical role to play in bringing these applications of EM to maturity. Our current focus is on developing a web-based tool that combines the qualities of several prototypes that have been introduced to support EM. These include the EDEN interpreter [16], which has been extensively used in model-building studies over the last 25 years, and Cadence [7], a more recently developed platform that addresses problematic aspects of EDEN. A common feature in all these tools is the maintenance of networks of dependencies between observables that serve as exceptionally powerful representations of state. In the case of EDEN, these dependencies have been expressed using families of definitions of observables, or "definitive scripts". Modelling with definitive scripts [17] has been the basis of interfaces to distributed and collaborative activity for model-building in a multi-agent and multi-viewpoint context. Where EDEN is a hybrid tool that is linked to a traditional operating system environment using conventional procedural mechanisms, Cadence exploits much more general dependency relations that are so expressive that they can address the issues of storage, communication and interaction with devices directly [7]. Our current research is directed at extending a new prototype implementation of EDEN using JavaScript, HTML-5 and other emerging web technologies [18], [19] so as to incorporate both the qualities of EDEN (as illustrated for instance by the Web EDEN interpreter [20]) and the support for structure and process that Cadence affords [7]. Such a tool promises to enable distributed collaborative learning activities that go yet further than existing technologies in realising Papert's ideals.

II. Issues for the emerging web

The development of web technologies stands in a most interesting relationship to *computer science* in its core academic sense. This point is well-made by Halford, Pope and Carr [21], who highlight the paradoxical fact that "despite the huge effect the web has had on computing ... computer scientists rarely study the web as a subject in its own right". They go on to observe that studies of the web "rarely breach the embedded binary divide between the natural and engineering sciences on the one hand and the social and human sciences on the other".

ISBN: 978-1-4673-4969-7 ©2013 IEEE

To some extent, the peripheral status of the web in the traditional computer science perspective has reflected the history of web use. Though Web 2.0-related activities have involved much more sophisticated use of computing technology and raised the possibility of harnessing the web to enable machine processing of human meanings, they have also raised the profile of social and human aspects of web use as *epiphenomena*. It is only latterly, as web technologies such as Adobe Flash, HTML-5 and SVG have begun to address the aspirations of users to play a deeper role as contributors and participants in development that the relation between the web and the core agenda of computer science has been highlighted.

Though the emerging web now shares many of the characteristics of a traditional computing resource, typical web use is oriented towards exploiting the computer in quite a different way. The relation of the programmer to the computer is traditionally framed with reference to a notion of crafting predictable behaviours on a reliable device. As Ben Ari argues in [22], in this traditional computer science viewpoint "the computer forms an *accessible ontological reality*". But conceiving typical interaction with the web as *program-like* specification of useful and interesting behaviours on such a device is problematic. In classical programming, only the programmer knows the computer code – it is framed in abstract and logical terms, is not intended for observation by the user, and is not easily connected with the domain of use. In the web context, it is much more natural for creation to be a public activity (cf. Papert's vision for constructionism [1]), for webpage development to involve distributed collaboration that reflects the insights and perspectives of many different agents, and for the resulting object to be a 'live' entity. And whereas the classical user sees the computer as defined by properties independent of its physical and social context, the web user is often explicitly conscious of being in touch with other minds. They consult the web in an exploratory spirit, wanting to know if somebody else knows the answer or has shared their experience, enlisting other people's involvement and interest, enjoying the sense of potential for evolution and possibly unexpected new developments (cf. the philosophy behind "the lean start-up" [23] that new technologies afford).

Providing technology well-suited to supporting learning in this context is challenging. For obvious reasons, much educational technology has been conceived within the same kind of philosophical and technical framework that classical programming affords. The whole concept of framing educational software with respect to pre-conceived learning objectives is consistent with conventional notions of computer programming. It can provide a good platform for staging and analysing predictable responses and can be engineered to some degree to allow for adaptation. But – as the well-recognised problems of adapting complex software systems to changing requirements show – (re)designing and (re)developing software to take account of open-ended human-oriented activities is hugely problematic. The key idea in this paper is that the problems of developing educational technologies for the emerging web and of developing complex software in a manner that supports radical design have a common root. Specifically, the new conceptual framework for computing sketched in the rest of the paper ('constructivist computing') is commended as essential for enhancing learning applications and the open-ended development of software.

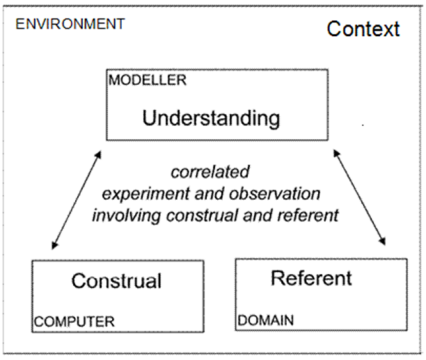

Fig. 1. Making an EM construal

III. Reconceptualising Computing

When alluding to "the embedded binary divide", Halford, Pope and Carr [21] echo the two contrasting paradigms associated with cultures in engineering identified by Brödner [24]:

> One position ... the *closed world* paradigm, suggests that all real-world phenomena, the properties and relations of its objects, can ultimately, and at least in principle, be transformed by human cognition into objectified, explicitly stated, propositional knowledge. The counterposition ... the *open development* paradigm ... contests the completeness of this knowledge. In contrast, it assumes the primary existence of practical experience, a body of tacit knowledge grown with a person's acting in the world. This can be transformed into explicit theoretical knowledge under specific circumstances and to a principally limited extent only ... Human interaction with the environment, thus, unfolds a dialectic of form and process through which practical experience is partly formalized and objectified as language, tools or machines (i.e. form) the use of which, in turn, produces new experience (i.e. process) as basis for further observation.

Traditional programming relies on engineering closed worlds within the application domain [25]. This has been a significant influence on one way of conceiving the Semantic Web: as a vehicle for automating cognitive processes. In their provisional manifesto for Web Science, Halford, Pope and Carr [21] stress the need to take account of a complementary view of knowledge such as is represented in Brödner's 'open development' paradigm. In keeping with the more central role for human cognition that this entails, Halpin, Clark and Wheeler [26] argue that the discussion of "the character and status of web objects such as websites and mash-ups" is more appropriately related to "what is sometimes called *4E (embedded, embodied, enactive, extended)* cognition". Embracing these complementary perspectives on knowledge and cognition within a single conceptual framework for computing is key to bridging the binary divide.

We propose to bring together these broadly 'scientific' and 'humanist' perspectives by adopting the principles of Empirical Modelling (EM), as outlined in the introduction above. An EM construal is an interactive artefact semantically similar in

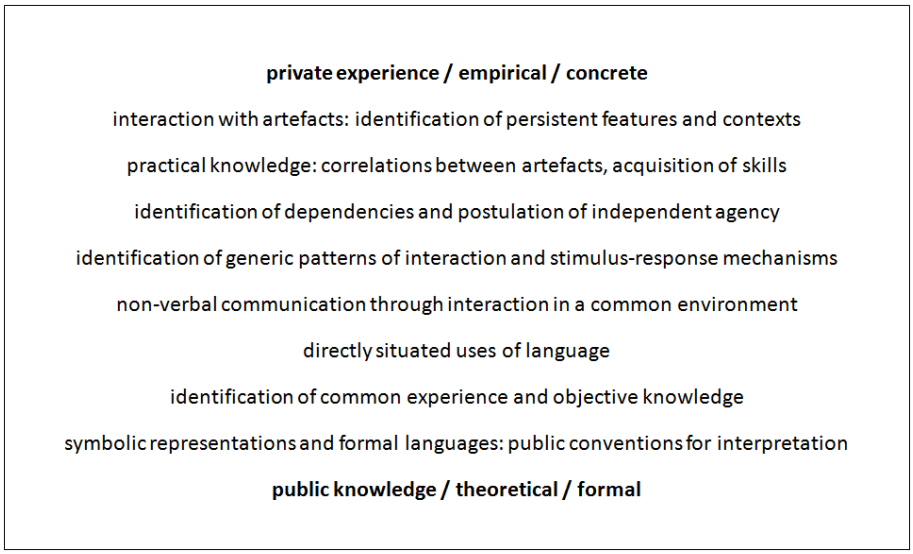

Fig. 2. The Experiential Framework for Learning (EFL)

character to a spreadsheet, in that it is meaningful only because a human interpreter can directly apprehend the correspondence between its state and an external situation (or current 'state of affairs') to which it refers (its *referent*). Figure 1 depicts the activity involved in making an EM construal. This entails the simultaneous 'construction' of several essential components: the physical construal, its referent, a context for interaction and interpretation, and a tacit understanding acquired by the modeller that takes the form of familiarity with particular ways of interacting with the artefact and/or its referent and interpreting the relation between them.

A full account of EM is beyond our present scope (cf. [9]), but it may be noted that the notion of an EM construal is well-matched to the four core concepts associated with Web Science in [21]. An EM construal is both a social and a technological artefact (cf. *co-constitution*), it involves a close integration of human and automated agency (cf. *heterogenous networks*), it is generated and exhibited through enacted processes (cf. *performativity*), and relies for its integrity and meaning upon ways of configuring the environment and exercising the construal and its referent that are always subject to revision at the discretion of the modeller (cf. *immutable mobiles*).

IV. EM AND LEARNING

The key concepts of EM are *observables*, *dependencies* and *agency*. In EM, the apprehension of particular configurations of observables, dependencies and agency is seen as characteristic of intelligent human interaction within an environment. In broad terms: an *observable* is an element of our experience to which an identity and current status can be ascribed; a *dependency* is a perceived causal link between a change made to one observable and a contingent change that occurs to another; an *agent* is any family of observables that can be viewed as a corporate entity to which state change can be attributed. Making a construal involves personal crafting of a correspondence between interaction with the computer artefact and interaction with its referent that is ideally so intimate as to be immediately apprehended by the modeller.

With reference to Figure 1, the correlation between the construal and its referent is mediated by observables and dependencies whose very existence and identity depends upon how the context for interaction can be crafted and maintained. Apprehending this correlation is always a matter of personal experience to be authenticated experimentally, but every kind of agency may be represented in its construction. To this end, the modeller projects herself into the role of other human agents, establishing valid assumptions about their capabilities and capacity to experience similar correlations (as when a composer writes music he cannot himself play for a performer, and a scientist sets out experimental procedures and observations they expect another scientist to replicate). Over and above this projection onto other human agents, the modeller may also adopt an anthropomorphic perspective on other agents (cf. musical and scientific instruments) and exploit the computer to automate counterparts of their agency in the construal.

Some of the many varieties of learning activity that feature in EM are set out in Figure 2, where the activities listed from top to bottom informally represent typical stages in the transition from 'knowing' in a subjective to an objective sense. This transition identifies EM as 'construction' of the kind discussed by Latour in [10] (cf. [11]). A construal can play an important role in communication: different agents can interact with the same construal, possibly exercising complementary skills and intelligence in its interpretation (cf. different instrumentalists reading an orchestral score). This characteristic makes an EM construal better suited to the role in constructionist learning first conceived by Papert than Logo programs [27], [28].

Much EM research to date has been concerned with educational technology and learning (cf. [29], [30]). The scope of such applications demonstrates the potential for bridging the binary divide: they include studies from mathematics and theoretical computer science [31], experimental biology [13], medicine [32], practical skills such as car parking and digital device usage, history of technology [33], and music appreciation [34]. (Spreadsheet use in education [35] offers independent evidence of the merits of 'EM principles'.)

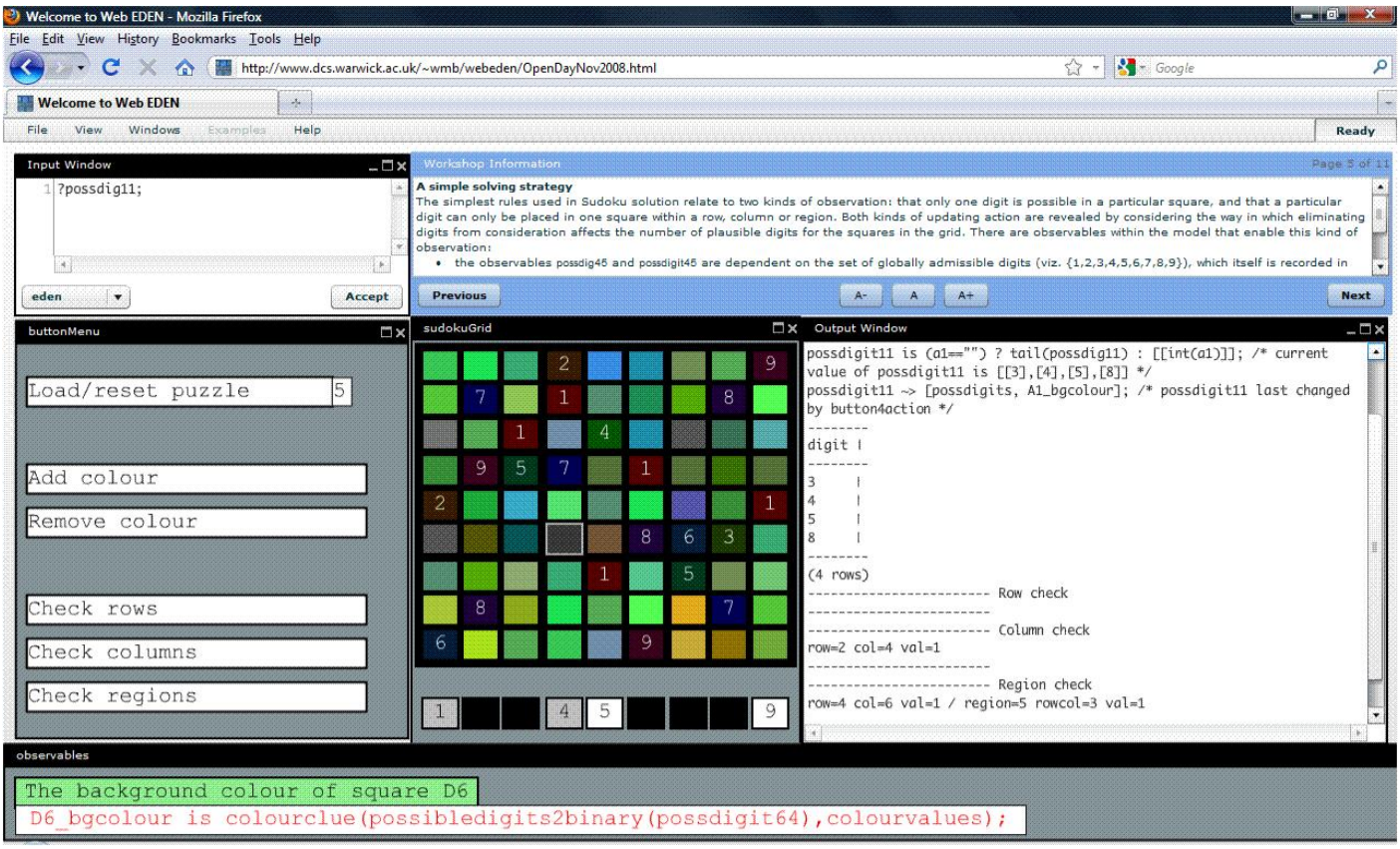

Fig. 3. A Sudoku-solving construal in the Web Eden environment accessible at **http://www.dcs.warwick.ac.uk/∼wmb/sudokuExperience/workshops**

V. WEB-ENABLED EM

In principle, the activities associated with transition from the top to the bottom of the EFL in Figure 2 have counterparts in the development of software using EM. This development begins with the making of a construal of the domain, and this construal evolves into an artefact that models the reliable interactions with automated devices expected of a software system. (The term 'construal' was borrowed from Gooding, who introduced it in connection with his accounts of Faraday's experimental work on electromagnetism [36]: a helpful parallel may be drawn with the way in which Faraday's construals led him to devise the first electric motor.)

The notion of developing programs from construals is well-aligned to the shaping of artefacts on the web. A web-page affords an experience of state out of which a behaviour can be crafted. In this process, as indicated in Figure 1, the roles of the human agents, the computing environment, and the contexts for interaction and interpretation are all subject to evolve.

In EM, every aspect of this evolution is interpreted with reference to observables, dependencies and agency. The activity depicted in Figure 1 is adapted so as to capture the perspective of any state-changing agent acting in the software environment, whether a human agent such as a user or developer (as in distributed participatory design [15]), or an automated agent such as a computer or device. This involves projecting the MODELLER perspective on to the interface between any such agent and its environment, and characterising its interactions with reference to *its* 'observables' and 'dependencies'.

In the early stages of development, EM takes the form of concurrent systems modelling to identify the relevant human and automated agents and the ways in which their interaction is mediated. To this end, the observables associated with an agent are classified into those that are directly apprehended ('oracles'), those conditionally under the control of the agent ('handles') and those defined by a dependency ('derivates'). The developer devises construals of the interactive context surrounding the individual agents that can then be integrated into a systemic view where the MODELLER in Figure 1 interacts as an objective external observer. Experiment and observation play a fundamental role throughout, and the ongoing crafting of observables, dependency and agency drives both the open-ended initial development and subsequent evolution (cf. the simulation of railway operation in [33]).

The EDEN interpreter mentioned in the introduction has been the implementation platform for most of the proof-of-concept educational studies cited above. Key ideas of EM software development have been illustrated using the web-enabled Web EDEN variant of EDEN [20], as deployed in an online activity for schoolchildren in the Sudoku Experience workshop in July 2008 (cf. Figure 3). Web EDEN as an environment for constructionism is discussed in [37].

Web EDEN has some practical limitations as an environment for learning and software development. The script for the Sudoku Experience workshop comprises some five

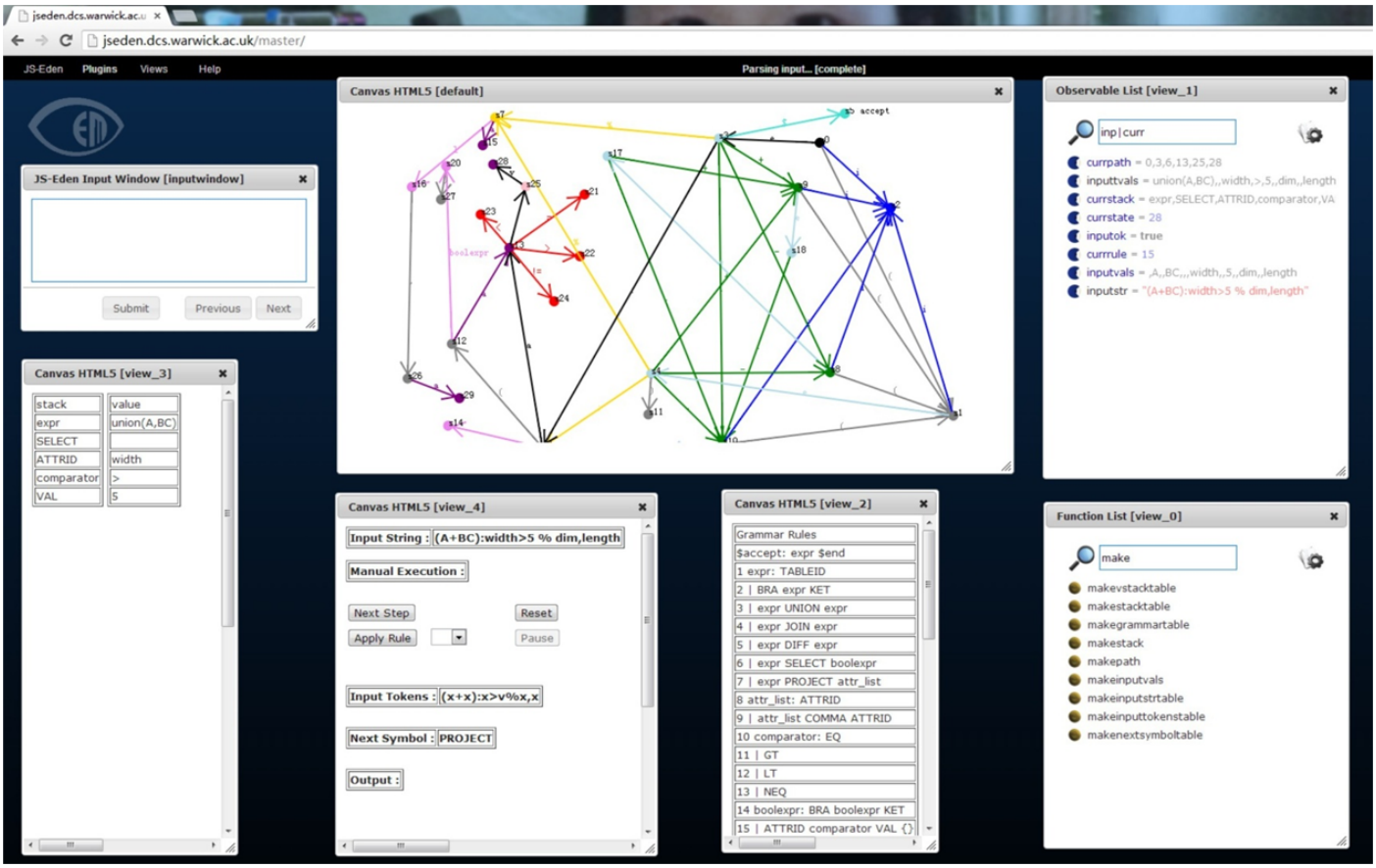

Fig. 4. A construal of a relational algebra expression translation developed by Hui Zhu: see **http://jseden.dcs.warwick.ac.uk/master/models/eddi2jseden0/**

thousand definitions of observables, many of which are syntactically very similar. Since EDEN makes use of various 'definitive notations' that address different aspects of the state of the construal (e.g. the screen layout, the graphics, the underlying relational database), there are several different kinds of observables. This poses problems for developers and learners alike in managing and manipulating the script. From a pragmatic perspective, a more promising approach to web-enabling EDEN is that introduced in JS-EDEN [19] (cf. Figure 4), where there is ready access to JavaScript objects and types well-suited to the Web, and where – unlike Web EDEN – the computational demand is distributed over the clients rather than being centralised at the server.

Some of the problematic aspects of EDEN are addressed in the prototype EM tool Cadence [7]. Cadence models the state of a construal in a homogeneous way that affords a form of prototype-based object-orientation. In this way, it meets the need in exploiting the computer for EM to make a realistic model of the computing technology itself – the processes, networking, and management of processors and peripheral devices. Whereas EDEN is a hybrid tool that supports dependency management and procedural code, Cadence is designed with an architecture that exploits dependency at every level in mind. This gives scope to exploit the capacity for parallel redefinition in a dependency network by ensuring that all definitions have the same granularity – as is crucial in relation to concurrency and synchronisation in distribution.

Current work is directed towards developing a JS-EDEN prototype that will combine the qualities of EDEN, Web EDEN and Cadence. An application of the latest prototype is illustrated in Figure 4. This depicts a construal of translation activity – in this instance such as is required to generate EDEN equivalents of expressions framed in the EDDI definitive notation for specifying dependencies between relational tables. As is characteristic of EM activity, the state depicted in the display can be construed as showing 'work-in-progress on developing the construal' and – simultaneously – a point in the automatic execution of translation of a specific input string. In the screenshot, the panel on the top left is an Input Window through which EDEN definitions can be entered. The two panels on the right display selected EDEN observables and functions (such as **k**, the number of input characters so far read, upon which many aspects of the current state depend). The other panels display the translation construal itself: they comprise a control panel within which the learner can trace the translation manually, the table of grammar rules, the current contents of the syntactic and semantic stacks, and the current state of the parsing process as recorded in conventional LR-parsing as a path in a finite state machine.

The translation construal potentially serves many roles. Adding an EDDI translator to JS-EDEN is an essential step towards porting existing EDEN models to JS-EDEN and making the construal informs the JS-EDEN developer's understanding of how such a translator can be implemented. The construal

ISBN: 978-1-4673-4969-7 ©2013 IEEE

is interesting in its own right as an educational application that allows a learner to trace the conventional syntax-directed translation process manually and appreciate the way in which issues such as shift-reduce conflicts are then addressed in automated execution. Its construction illustrates how the configuration of the model-building environment itself can be seamlessly integrated with model building. The construal already illustrates how object-orientation in JavaScript can be exploited. Future work aims to extend this to devise a template to apply to translating expressions in any definitive notation.

VI. Constructivist computing in practice

A conscientious reader can gain much insight into the vision for *constructivist computing* that motivates this paper by exploring the many associated references cited in previous sections. A more direct way to appreciate this vision is to get practical experience of EM tools. This section gives some pointers to practical activities that can be traced with the construals depicted in Figure 3, and relates these to key ideas described above.

The characteristics of the traditional EDEN interpreter as an EM tool [16] are most readily appreciated by consulting the *Sudoku Experience* workshops at the url cited in the caption to Figure 3. These workshops include exercises for the learner, but their initial goal is to provide orientation and acquaint the learner with what is involved in making an EM construal. With reference to Figure 1, the reader can inspect the source files for the workshops to see the scale and nature of the definitions that make up the construal, and refer to the "General Introduction to the Script" and the "Introduction to the Web EDEN environment" to appreciate how the script is interpreted as modelling a context for potential action resembling the network of definitions in a spreadsheet and how observables and dependencies in the construal have counterparts in the experience of a person contemplating a Sudoku puzzle.

These preliminary workshops introduce the learner to the activity through which the relationship between the construal and its referent depicted in Figure 1 comes to be established, exposing and interpreting the "correlation between experiment and observation with the construal and the referent" that informs the modeller's understanding. In these activities, the learner is cast in the role of a participant on a guided walk. With reference to Brödner's contrasting paradigms, it is quite apparent that the guide cannot communicate their own appreciation of such a walk fully as 'objectified, explicitly stated, propositional knowledge' but only through 'human interaction with the environment' that is 'partly formalised and objectified' with the help of commentary and demonstration.

As with a guided walk, the significance of the activity depends crucially on what part of the environment is being visited, the extent to which the guide is familiar with it, and what kind of observation and interaction is given priority. Each workshop has an associated minibook that provides commentary and direction. For the most part, the workshops exercise the construal in ways that focus on experiment and observation relating to 'Sudoku solving' or 'making a construal for Sudoku solving'. Some of the exercises on the other hand diverge from these primary themes to such an extent that they might be seen as elaborating independent construals. For instance: 'a basic exercise in EM' in Workshop 2A extracts two squares from the Sudoku grid and prescribes a sequence of interactions by way of redefinitions designed to illustrate the interplay between expectation, experiment and explanation that is characteristic of EM; Workshop 2B invites the reader to adapt the Sudoku grid to make a simple form of electronic blackboard on which to demonstrate primary school arithmetic.

The essential role of human interaction and interpretation in giving form and identity to a construal is entirely in keeping with its constructed nature. The Sudoku workshops illustrate how the fluid nature of a construal provides a basis for the 'constant innovation' favoured by Ries [23], as for instance in the conception and implementation of the *Colour Sudoku* puzzle variant [38] whose derivation is described in Workshop 2C. They also show how several different agent perspectives, such as those of the developer, teacher and learner in a constructionist vision of learning, can be closely integrated within one and the same environment [27]. This scope for integration stems from the support that interaction with the Sudoku solving construal gives to learning activities associated with diverse kinds of agency at every level in the EFL in Figure 2 – and most distinctively with personal subjective agency. This diversity is reflected in the many ways in which knowledge of the digits that might plausibly be entered in a given cell is conveyed – implicitly in the presentation of the grid in Figure 3, by a 1-dimensional relational table defined by a dependency, by the background colour of the cell, or through the exploratory interactions that the interface to the script affords (whether through direct redefinition of observables via the Input Window, or – in another variant of the construal – via the slider interface that features in [38]).

The model of agency that a construal supports is rich both in a 'vertical' and 'horizontal' sense. As a model of a moment in time, it admits observables that mediate agency in an open-ended extensible fashion. As a model of a trajectory of transitions from moment to moment, it records agent interaction in such a way that it can be replayed *as if* live. Both of these qualities are illustrated in the extension to the Sudoku Experience environment developed for a workshop presented at the Constructionism 2010 conference [39]. The steps in this extension, which was carried out in an intermittent fashion over several days and involved interleaving the modification of the construal with the solution of a Sudoku puzzle as a test case, were recorded in detail and documented using the Empirical Modelling Presentation Environment [40].

Developing a program from a construal involves constraining interaction in both vertical and horizontal dimensions. When appropriate constraints have been identified (cf. [25]), a suitable context for program-like interaction with the construal can be established (cf. Figure 1). This context may be respected through discretionary interaction on the part of the modeller (as is illustrated in Workshop 3A). It is also possible to provide an interface to restrict enrichment of the state through the addition of new observables and patterns of state-transition in preconceived ways (as is illustrated in Workshop 3B). The specific closed-world functionality associated with constrained interaction with construal can also be realised in a traditional program (cf. Harfield's Colour Sudoku program at [38]).

VII. Instruments for constructivist computing

Working through the *Sudoku Experience* workshops as outlined in the previous section is a good way to understand the aspirations for constructivist computing. In principle, making construals offers unprecedented scope for imaginative construction that exploits re-use and allows design experiences to be retrieved, retraced and reappraised. Given the richness of the cognitive activity that surrounds the making of a construal and the extended period of time involved in its interactive construction, it is unreasonable to expect the exploration of a mature construal to be a routine matter. We have grown accustomed to exploiting the computer as a device that reduces the burden of thinking, enabling us to act efficiently to achieve our goals without obliging us to reflect in depth upon what this involves. By contrast, working with construals is of its essence problematising, prompting us to probe our understanding in the spirit of McCarty's *Humanities Computing* [41].

In practice, the Sudoku Experience workshops also illustrate a range of problems that seem incidental to the primary agenda of making and exploring construals. Even for the authors of construals, it can be hard to meet the challenges of accessing and identifying relevant observables, finding a suitable organisation of files of definitions, and documenting their interactions and interpretations. A key issue is that both the vertical and horizontal components of a construal and the interplay between them should ideally be recorded. This is because of the 'spiral' nature of the elaboration of observables and dependencies, which become richer as the contexts for interaction with the construal become more mature.

The problems faced in making construals resemble those of developing and maintaining webpages in many respects. As web technologies become more sophisticated, webpages need to provide the context for interaction by many different kinds of agent, and crafting such contexts is much closer in spirit to EM than to programming in its classical sense. At the same time, the scope for conventional programming activity in web environments is ever increasing, and a satisfactory conceptual account of web development must accommodate this.

The design of JavaScript, "the programming language of the Web", reflects the way in which the creation of concrete artefacts and the specification of abstract behaviours come together in web development. This confluence presents challenges typical of the meeting of two cultures to which Brödner refers [24]. The scope and scale of Flanagan's 'definitive guide' to JavaScript [42] illustrates the difficulty of giving conceptual integrity to the many aspects of the language. As Crockford observes in his account of JavaScript as 'an outstanding object-oriented language', the reputation of the language is tarnished by association with the API of the browser, the 'quite awful ... poorly specified and inconsistently implemented' Document Object Model (DOM) [43, p.2].

JS-EDEN [18] aspires to address this conceptual and technical challenge by marrying JavaScript with EDEN. In this way, web support for EM can benefit from the contributions that JavaScript has made to the management of complex state by way of structural and interface mechanisms, and capitalise on existing resources for dealing with the experiential aspects of state. By a happy coincidence, the core syntax of JavaScript is so similar to that of EDEN that it makes sense to extend the EDEN interpreter to admit raw JavaScript as an auxiliary notation. This potentially makes it easy to blend the many different kinds of agency represented in the EFL.

By comparison with more established implementations of EDEN, JS-EDEN is as yet only a research prototype. A significant novelty is that, through the close connection with JavaScript, the entire web interface to the interpreter can be readily customised. This has been illustrated in JS-EDEN variants such as Harfield's simplified interface for educational use in schools (cf. [44], [19]). The 'master' variant of JS-EDEN whose use is depicted in Figure 4 illustrates the most effective way to exploit and generalise the scope for customisation. In this variant, the characteristics of the interface itself are within the scope of the construal. This is in keeping with the idea that the construal is accessible at one and the same time to agents with quite different status – such as the developer, teacher and learner in a constructionist setting [27] – whose roles are compartmentalised in conventional programming. The screen layout can be composed from a range of components that includes JS-EDEN input windows, HTML-5 canvases, symbol viewers and HTML views. The location, dimensions and visibility of each component are observables within the environment of the construal. By introducing several instances of input windows, canvases and symbol viewers, the modeller can reflect the essentially concurrent nature of the interaction with a script, creating separate interfaces for the distinctive oracles and handles associated with each kind of agency. As is illustrated in the panels on the right hand side in Figure 4, the symbols displayed within a view can be selected by specifying an appropriate regular expression.

These features of JS-EDEN support modes of interaction that are characteristic of an *instrument* rather a tool. Zhu's parsing model can be regarded as a prototype for providing interfaces to auxiliary definitive notations such as traditional EDEN supports. Other prototypes that have been developed provide means to generate scripts from templates and record these in 'script observables' that can take over the role conventionally played by files. Dependencies between script observables can then be used to express the subtle relationship between the script as developed horizontally over time and the script as established vertically as of the current moment.

VIII. Concluding remarks

The qualities of the alternative conceptual framework for computing that EM promotes are hard to appreciate without direct experience of its practice. Only by gaining such experience is it possible to grasp the distinctive meaning of its core concepts of observable, dependency, and agency. The aspiration in EM is "to realise software development as a lived experience" [45], and in the process to establish an intimate relationship between software construction and learning. EM thinking has much in common with Bret Victor's prescription for 'learnable programming' and his concern for a programming model "that [allows] for continuous change" [46]. In particular, Victor emphasises the crucial significance of the concrete and of direct perception of state ("People understand what they see") and endorses the idea that in order to get people to understand programming "we change programming". The objectives that inform Victor's learnable programming environments are similar to those that have guided the development of instruments

for EM, and both have drawn crucial inspiration from Papert's foundational thinking about the relationship between software construction and learning. A topical objective for our research on EM tools is to advertise the fact that open development in spirit very similar to that demonstrated by Victor in [46] – but essentially different in character – is a well-established feature of EM practice. The crucial difference is that *construals* rather than programs are the focus of EM development. And indeed, experience of EM gives grounds for scepticism about the extent to which Papert's constructionist vision can be realised without a radical reappraisal of traditional computational thinking [6].

ACKNOWLEDGMENTS

The contributions of the first three authors deserve more explicit acknowledgment. Special credit is due to Richard Myers for creating Web EDEN, to Tim Monks for conceiving JS-EDEN and building the first prototype, and to Nick Pope for his work on Cadence and the 'master' variant of JS-EDEN.

We are also indebted to Steve Russ for insights that have played an important role in developing the conceptual framework for constructivist computing.

REFERENCES

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

[2] http://scratch.mit.edu.

[3] http://ccl.northwestern.edu/netlogo/.

[4] http://www.warcraft-mods.com/.

[5] B. Nardi and J.Kalinikos, "Technology, Agency and Community: The Case of Modding in World of Warcraft," in *Industrial Informatics Design, Use and Innovation: Perspectives and Services*, J. Holmstrm, M. Wiberg, and A. Lund, Eds. IGI Global, 2010, pp. 174–186.

[6] 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.

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

[8] www.dcs.warwick.ac.uk/modelling.

[9] 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, pp. 197–228.

[10] B. Latour, "The promises of constructivism," in *Chasing Technoscience: Matrix of Materiality*, D. Ihde, Ed. Indiana University Press, 2003, pp. 27–46.

[11] W. M. Beynon and A. J. Harfield, "Lifelong Learning, Empirical Modelling and the Promises of Constructivism," *J of Computers*, vol. 2, no. 3, pp. 43–55, 2007.

[12] M. Beynon, "Towards technology for learning in a developing world," in *Proc. IEEE 4th International Workshop on Technology for Education in Developing Countries*, Iringa, Tanzania, Jul. 2006, pp. 88–92.

[13] D. Keer, S. Russ, and M. Beynon, "Computing for construal: an exploratory study of desert ant navigation," *Procedia Computer Science*, vol. 1, no. 1, pp. 2207–2216, May 2010.

[14] M. Beynon and S. Russ, "Experimenting with Computing," *Journal of Applied Logic*, vol. 6, pp. 476–489, 2008.

[15] M. Beynon and Z. E. Chan, "A conception of computing technology better suited to distributed participatory design," in *NordiCHI Workshop on Distributed Participatory Design*, Oslo, Norway, Oct. 2006.

[16] Y. P. Yung and Y. W. Yung, *The EDEN Handbook*, Department of Computer Science, University of Warwick, 1988, updated in 1996.

[17] J. Rungrattanaubol, "A treatise on modelling with definitive scripts," Ph.D. dissertation, Department of Computer Science, University of Warwick, Apr. 2002.

[18] T. R. Monks, "A definitive system for the browser," MSc Dissertation Report, Computer Science, University of Warwick, 2011.

[19] go.warwick.ac.uk/em/software/js-eden/.

[20] go.warwick.ac.uk/webeden.

[21] S. Halford, C. Pope, and L. Carr, "A Manifesto for Web Science," in *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010, http://journal.webscience.org/297/.

[22] M. Ben Ari, "Constructivism in Computer Science Education," *J. Computers in Mathematics and Science Teaching*, vol. 20, no. 1, pp. 45–73, 2001.

[23] E. Ries, *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Penguin Books, 2011.

[24] P. Brödner, "The two cultures in engineering," in *Skill, Technology and Enlightenment*. Springer-Verlag, 1995, pp. 249–260.

[25] W. M. Beynon, R. C. Boyatt, and S. Russ, "Rethinking programming," in *Proc. IEEE Third International Conference on Information Technology: New Generations*, Las Vegas, Nevada, 2006, pp. 149–154.

[26] H. Halpin, A. Clark, and M. Wheeler, "Towards a Philosophy of the Web: Representation, Enaction, Collective Intelligence," in *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010, http://journal.webscience.org/324/.

[27] M. Beynon and C. Roe, "Computer support for constructionism in context," in *Proc. ICALT'04*, Joensuu, Finland, 2004, pp. 216–220.

[28] C. Roe and M. Beynon, "Dependency by definition in Imagine-d Logo: applications and implications," in *Ivan Kalaš (ed.) Proc. of the 11th European Logo Conference*, Bratislava, Slovakia, 2007, EM paper #102 at [8].

[29] A. J. Harfield, "Empirical Modelling as a new paradigm for educational technology," Ph.D. dissertation, Department of Computer Science, University of Warwick, Jan. 2008.

[30] C. P. Roe, "Computers for Learning: An Empirical Modelling Perspective," Ph.D. dissertation, Department of Computer Science, University of Warwick, Nov. 2003.

[31] M. Beynon, "Constructivist Computer Science Education Reconstructed," *HEA-ICS ITALICS e-Journal*, vol. 8, no. 2, pp. 73–90, 2009.

[32] M. Beynon and W. Beynon, "Construals to Support Exploratory and Collaborative Learning in Medicine," in *Proc. Intelligent Support for Exploratory Environments*, Chania, Crete, 2012, pp. 12–20.

[33] M. Beynon and P.-H. Sun, "Computer-mediated communication: a Distributed Empirical Modelling perspective," in *2nd International Conference on 'Cognitive Technology'*, San Francisco, 1999, EM paper #053 at [8].

[34] M. Beynon, S. Russ, and W. McCarty, "Human Computing: Modelling with Meaning," *Literary and Linguistic Computing*, vol. 21, no. 2, pp. 141–157, 2006.

[35] J. Baker and S. J. Sugden, "Spreadsheets in Education – The First 25 Years," *Spreadsheets in Education*, vol. 1, no. 1, 2003, article 2.

[36] D. Gooding, *Experiment and the Making of Meaning*. Kluwer, 1990.

[37] M. Beynon and A. Harfield, "Constructionism through Construal by Computer," in *Proc. Constructionism 2010*. The American University of Paris, 2010.

[38] go.warwick.ac.uk/sudoku.

[39] go.warwick.ac.uk/constructionism2010-workshop.

[40] empublic.warwick.ac.uk/projects/empeHarfield.

[41] W. McCarty, *Humanities Computing*. Palgrave Macmillan, 2005.

[42] D. Flanagan, *JavaScript: The Definitive Guide*, 6th ed. O'Reilly, 2011.

[43] D. Crockford, *JavaScript: The Good Parts*. O'Reilly, 2008.

[44] A. Harfield and M. Beynon, "Empirical Modelling for constructionist learning in a Thai secondary school mathematics class," in *Proc. 9th International Conference on eLearning for Knowledge-Based Society*. Siam Technology College,Thailand, Dec. 2012, EM paper #118 at [8].

[45] M. Beynon, "Realising software development as a lived experience," in *Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Tucson, Arizona, 2012, pp. 229–244.

[46] http://worrydream.com/LearnableProgramming/.