MAKING CONSTRUALS AS A NEW DIGITAL SKILL FOR LEARNING

Meurig Beynon[1], Russell Boyatt[1], Jonathan Foss[1], Chris Hall[1], Elizabeth Hudnott[1], Steve Russ[1], Erkki Sutinen[2], Hamish Macleod[3] and Piet Kommers[4]
[1]*University of Warwick, UK*
[2]*University of Eastern Finland, Finland,*
[3]*University of Edinburgh, UK,*
[4]*Helix-5, The Netherlands*

ABSTRACT

Making construals is a practical approach to computing that was originally developed for and by computer science undergraduates. It is the central theme of an EU project aimed at disseminating the relevant principles to a broader audience. This involves bringing together technical experts in making construals and international experts in educational technology in organising wide-ranging learning activities. This paper, which complements the tutorial and discussion on making construals in (Beynon et al. 2015a, 2015b), documents issues that arise from this encounter. Its central argument is that making construals offers a significant new contribution to educational technology.

KEYWORDS

Construal; computing education; educational technology; constructionism; software development; radical empiricism

1. MAKING CONSTRUALS

Making construals is a new digital skill that can be seen as complementary to 'writing computer programs'. The key characteristics of making construals are introduced in (Beynon et al. 2015a), a tutorial paper in which the use of JS-EDEN, a prototype online environment for making construals (the "MCE"), is illustrated by making an elementary construal of shopping activity (a "Shopping construal"). As discussed and illustrated in (Beynon et al. 2015a), many variants of the Shopping construal can be developed with relatively modest computing expertise. These include traditional educational resources that might be used to scaffold learning in teaching 'life-skills' and simple educational games.

Making construals is the practical core of a well-established research project (Empirical Modelling 2015) that was first developed for – and by – computer science students at the University of Warwick. It is currently the theme of an EU Erasmus+ project ("CONSTRUIT!") aimed at dissemination to a wider audience. Adapting resources that were originally conceived for computing specialists for this purpose has entailed a significant shift in emphasis. Though the technical development of the MCE and the construals described in (Beynon et al. 2015a, 2015b) is being led by members of the CONSTRUIT! team at Warwick, it has been informed through dialogue with expert international consultants in educational technology and by several collaborative project activities, including workshops with schoolchildren at SciFest in Joensuu, Finland and with schoolteachers in Athens. This abbreviated version of (Beynon et al. 2015b) summarises the diverse issues that have been raised in this process. These summaries do not necessarily represent a consensus view of the consultants, but inform an agenda for subsequent discussion.

The focus for CONSTRUIT! activity over the first year has been on showing that comprehending and making construals is potentially accessible to the non-specialist. The discussion of a Shopping construal in (Beynon et al. 2015a) illustrates the key principles and techniques involved. Our previous experience of teaching undergraduate computer students to make construals (cf. (Empirical Modelling 2005)) suggests that novices typically begin by exploring an existing construal that is connected with their chosen topic of investigation, and develop their skills through making adaptations. In some cases, this leads them to take ownership of the construal through making their own extensions. In any event, once students have developed

the ability to understand and extend someone else's construal, they have the basic competence to start making their personal construals from scratch.

As illustrated in the derivation of a simple game from the Shopping construal in (Beynon et al. 2015a), if we have some explicit extension, modification or specialisation of an existing construal in mind there are three stages in the adaptation:

- **conception**: without reference to the precise syntactic form and structure of their definitions, draw up a plan for the conceptual reorganisation of observables. This may entail re-disposing or replicating visual components, revising existing observables and actions, and possibly introducing new observables and actions.
- **identification**: identify the precise names and definitions of the observables and agents to be adapted through open-ended interaction with the construal. This activity exploits the range of viewers that are available in the MCE – such as the Observable- / Function- / Agent- / Symbol-Lists and Dependency Maps. Domain knowledge may also be helpful in guiding the search activity.
- **reconfiguration**: having identified the relevant observables and agents, modify their definitions according to the plan drawn up at the **conception** stage.

The most important consideration in this process of comprehension and adaptation is that the transition from the initial construal to the revised construal takes place in the maker's stream-of-thought. That is to say, all interactions with the construal, whatever their status (exploratory interactions, revisions, tests, experiments, corrections, aberrations, etc) are conducted with reference to the live construal and its real or imagined referent as depicted in Figure 1.

In practice, the above sequence of stages would not necessarily be followed strictly. Prior identification and exploration of observables and agents in the construal might be required in order to make a plan for revision. In an ambitious development there would also be many phases of the conception-identification-reconfiguration sequence. The MCE provides support for generating the current script (to be saved in the external file system) and for recording intermediate states which the maker can restore if needed.

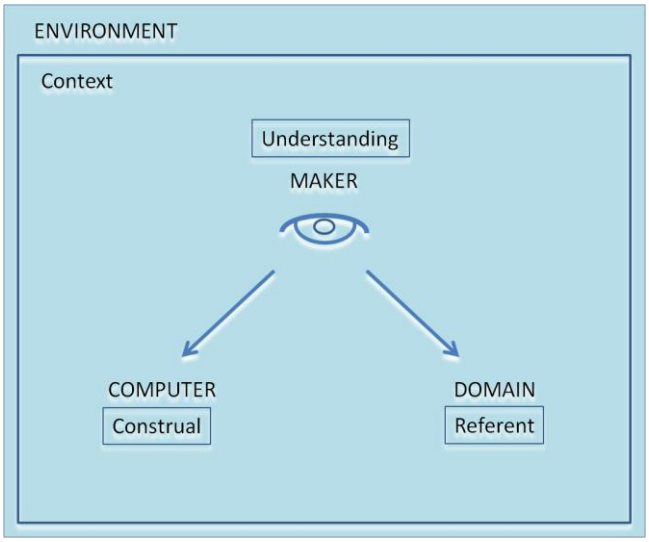

Figure 1. Making a Digital Construal

Figure 1 depicts the relationship between construal and referent as it is experienced by the maker as of one moment in time. The maker's understanding of this relationship, which gives the construal its informal meaning, is built up through interaction with the construal and its referent as these evolve over time. This understanding incorporates expectations about the impact of an interaction that are based on knowledge gained from prior experiment. The maker can in principle change observables in an unconstrained fashion, but in practice only makes such interactions as help to establish and refine understanding of the relationship between the construal and its referent. Refinement will typically involve exploring the perspective of different agents. In the Shopping construal, the customer is not in general able to fix the price of the items for sale for instance – this is a decision for the shopkeeper. The integrity of the construal relies on certain

assumptions about the stability of the context for interaction. For instance, we would not buy perishable fruit by mail order, and in some cultures the price of items would be negotiable. The most appropriate semantic framework for the making activity in Figure 1 regards all semantic relationships as pragmatically determined and rooted in the connections that can be made in the personal experience of the maker. As explained in (Beynon et al. 2015a, 2015b), this is the fundamental tenet of William James's radical empiricism:

"... the parts of experience hold together from next to next by relations that are themselves parts of experience. The directly apprehended universe needs, in short, no extraneous trans-empirical connective support, but possesses in its own right a concatenated and continuous structure."

The Meaning of Truth (James 1907, reprinted 1975), Preface: p.xiii.

2. MAKING CONSTRUALS AS AN EDUCATIONAL TECHNOLOGY

Making construals establishes a close link between how we think about a subject domain ('how we *construe* a situation') and how we might develop software to simulate behaviours and address problems drawn from that domain. In this respect, it represents a new way in which to realise program-like behaviours. When we write a program, its behaviour is prescribed in accordance with a specification. The specification need not be preconceived; it may be fluid and evolving. But – though it may be implicit – it is always present, if only (as in an agile development) as 'what the software does now'. A construal by contrast is an informal representation of the playground for agency within a situation as we currently conceive it. In the construal, there are counterparts for the patterns of agency and interaction that we encounter through experience and experiment. And where the interactions with a program are constrained by a user-interface that is closely aligned with its specification, the ways of interacting with and interpreting a construal are open-ended and informed by what 'makes sense' in the maker's imagination.

Previous papers have built on these ideas to argue that making construals is a 'radical new conception' in educational technology (see e.g. (Beynon 2007)). Educationalists, having seen so much hype around the latest technologies, have a proper scepticism about such claims to novelty. Their wariness may reflect the influence of traditional computer science thinking: from a perspective informed by elementary exercises in programming, developing software can be conceived as 'implementing algorithms to fulfil a preconceived formally specified functionality', and there is an abstract mathematical sense in which any two correct implementations are equivalent. In reality, orthodox thinking about software is very ill-suited to themes such as constructionism (Papert 1980; Beynon & Harfield 2010), adaptable open educational resources (Wikipedia 2015) and transformative practices in online teaching (Ross et al. 2013; Beynon & Zhu 2013). An aspiration for the CONSTRUIT! project is to give an account of computing that does fuller justice to these well-conceived and important educational initiatives and supplies a more appropriate grounding for their practices.

There are many different ways in which construals can contribute to learning about a domain. Making a new construal of your own can be a way of learning, as can interacting with an extant construal. There is much affinity between the skills required for making a new construal – as outlined in the discussion of learning *how to* make construals above – and those required for exploring an extant construal. In the latter context, the focus is on how we learn about a domain or a referent *by* making or adapting construals.

Making construals encourages the learner to adopt an unusual stance. Much technology to support learning is associated with bodies of knowledge that have been well-documented and formalised. The learner's interactions are staged within a frame that has been carefully prepared, possibly even so that there is an intended 'correct' path to follow. The archetypal context for making construals is one in which the learner is invited to act with greater autonomy. Making the connection between a construal and a referent that is depicted in Figure 1 is a personal matter that draws on different aspects of the learner's experience. The concern is not only with what the learner directly observes but with its resonance with everything else to which they are sensitive. And whilst the learner is led to identify observables and dependencies in the role of a model-builder, this is with a view to appreciating the subtlety and richness of their immediate experience of the domain more fully. In sharp contrast, the objective for traditional model-making is to find simplifying mechanisms that can be applied to the referent to serve the specific goal in hand. What is abstracted from the referent in this way is in effect detached from the domain.

In analysing how students learn to program in languages such as Lisp, Prolog, Smalltalk and Logo, three aspects of experience are significant:

- prior experience of implementing algorithms;
- what is being directly experienced in interacting with the tutorial environment;
- what intuitions have been acquired through experience of the subject domain.

In evaluating the quality of the learning experience, it is the way in which these different kinds of experience are brought together that counts. In particular, these three ingredients of experience guide the learner in making the links between familiar and new concepts.

In making construals, the above aspects of the learner's experience have counterparts that are respectively associated with the modeller's understanding, the evolving construal and its emerging referent. A distinctive feature of the MCE is the degree of active control it gives over the way in which these elements are brought together in the learner's experience (cf. the way in which visualisations of states and views of observables can be freely configured). This encourages the student to actively control and optimise the balance between prior and new ("restructured") concepts.

Learning to make construals entails a reorientation that presents a challenge of somewhat the same kind that is associated with adopting a declarative rather than a procedural approach to programming. Not all students will welcome this reorientation, but it potentially has unique value for learning. This stems from the emphasis on helping the learner to rethink topics that might initially appear trivial. This is particularly well-illustrated in the large body of project work that has been developed using the previous variants of the EDEN interpreter (Empirical Modelling 2006; Empirical Modelling 2013) and has yet to be replicated in the online MCE. By way of illustration, consider the construals of heapsorting (Beynon 2008a), elementary group theory (Beynon 2008b) and the Clayton Tunnel railway disaster (Chan & Harfield 2008).

3. THE TECHNOLOGICAL PERSPECTIVE

The relationship between making construals and other established technologies is complex. Many technologies appear to be addressing closely related agendas and delivering superficially similar products.

Making construals is significant as paradigmatic for a particular way of exploiting computer-based technology. Where computational thinking focuses on activities for which the archetype is 'a mind following a rule' (Hodges 2002), making construals is concerned with 'a person making sense of a situation'. The transition from informal personal accounts to formal objective understanding is characteristic of such sense-making. Deriving programs from a construal, as illustrated in (Beynon et al. 2015a), is just one instance of how making construals enables this transition to take place seamlessly. In keeping with the notion that "*The directly apprehended universe needs, in short, no extraneous trans-empirical connective support"* the distinction between the informal and formal components in this transition is in the mind of the maker. In one context (cf. Figure 1): 'this is what reliably seems to be the case' and, in another: 'this is what we can confidently deem to be the case'.

The transition from the empirical to formal perspectives is prominent in many settings. It has a critical role in expert systems (Jackson 1998) and design science (Hevner et al. 2004). Working in close proximity to the formal perspective attracts particular interest because of aspirations for automation and AI. By comparison, the human-centred activities associated with making construals serve a potentially less welcome role – making use of computing to provoke us into thinking more deeply.

Polya's reflections on teaching mathematics (Polya 2014) help to highlight the unusual qualities of construals. As Polya observes, though mathematics is done inductively, it is taught deductively. This can make the subject boring, alienating the learner from the researcher. As an environment that can enable the learner to cope with uncertain and chaotic elements, making construals gives valuable support for exposing the empirical roots of formal results. In keeping with Polya's remarks concerning the importance of reflection and review for learning, the history of interactions with a construal also provides an as-if-live record of the processes of rationalisation and problem-solving.

The narrative of making construals is likewise closely parallel to that of design science (Hevner et al. 2004). The learners are developing construals ("design artefacts") and thus constantly relating their work to emerging demands – perhaps from their own imagination (their "local contribution" in design science terms) and finally, and probably throughout the process, adding to the science ("global contribution"). This is a

plausible narrative, but makes challenging demands where the nature of the design artefacts is concerned. In (Hevner et al. 2004, p.89), Hevner discusses a mode of describing the design artefact that is based on searching for solutions to a family of design constraints. He then observes that in general finding such a solution is computationally infeasible and proposes to address this by relaxing constraints. Approaches of this nature, which operate with formal specifications and use sophisticated algorithms, give nothing like the delicate control over details of design and nuances of meaning that making construals affords. As in other contexts, claims concerning the character and quality of methods become rhetorical in the absence of appropriate representations.

The problematic aspect of incorporating the principles of making construals alongside more orthodox approaches is that it creates a fundamental ontological conflict. Though the maker may retain the prerogative to say 'this is what we can confidently deem to be the case' this is of no consequence if the integrity of the enclosing environment is predicated on the assumption that it *is* in some absolute sense the case. This helps to explain why merely adding dependency as an extra feature in an otherwise orthodox programming environment introduces problematic semantic issues (cf. (Roe & Beynon 2007; Tomcsanyi 2003)).

One of the objectives for CONSTRUIT! is to enable people other than computing specialists to make their own software applications. This is in the spirit of Mark Hatch's Maker Movement Manifesto (Hatch 2014), which promotes 'making' as an activity that is accessible to all. Hatch's focus is on making physical things, but computing and its associated technologies have a significant implicit role. As he observes: "The tools of making have never been cheaper, easier to use, or more powerful." ... "It may take some practice to get good at some kinds of making, but technology has begun to make creating easy enough that everyone can make.". Curiously, a better understanding of how technology for computing can support *making* is required if Hatch's thesis is to extend to making virtual products. Making construals is helpful in this respect, as might be expected given the character of the applications of spreadsheets in education in (Baker & Sugden 2007).

In the conception phase, making construals somewhat resembles the activities that have been conceived as ways of teaching computer science principles and concepts without using the computer (cf. 'computer science unplugged' (CS Unplugged 2015) and 'Barefoot' (Barefoot Computing 2015)). In effect, describing computing activities in non-technical human terms abstracts away the technically sophisticated apparatus that enables machines to manipulate and 'interpret' real-world observables and dependencies that a human interpreter takes for granted. This shifts the focus of attention from the purely technical to high-level issues more central to good design practice in computing. By way of illustration, consider how adapting the Shopping construal to allow many instances of each item (Beynon et al. 2015a) highlights a limitation in the representation of the stock of items – it does not allow the maker to record the identities of individual items.

4. CONCLUSION

This paper has argued that making construals is a practice, and an approach to software development, that is more accessible and intelligible for the non-specialist than coding (cf. Granger 2015). It has also argued that making construals is better suited to supporting learning and major themes in educational technologies than are the conventional methods and environments for programming.

ACKNOWLEDGEMENTS

We are much indebted to Tim Monks, Nick Pope, Antony Harfield and Joe Butler for their work on the development of the JS-EDEN environment. This project has been funded with support from the European Commission under the Erasmus+ programme (2014-1-UK01-KA200-001818). This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

REFERENCES

Baker, J. & Sugden, S., 2007. Spreadsheets in Education –The First 25 Years. *Spreadsheets in Education (eJSiE)*, 1(1). Available at: http://epublications.bond.edu.au/ejsie/vol1/iss1/2 [Accessed July 8, 2015].

Barefoot Computing, 2015. primary computing Barefoot Computing. Available at: http://barefootcas.org.uk/ [Accessed July 8, 2015].

Beynon, M., 2007. Computing technology for learning - in need of a radical new conception. *Journal of Educational Technology & Society*. Available at: http://www.jstor.org/stable/jeductechsoci.10.1.94 [Accessed July 8, 2015].

Beynon, M. et al., 2015a. Making construals as a new digital skill: dissolving the program - and the programmer - interface. In *International Conference on Interactive Technologies and Games (iTAG)*. Nottingham, UK: IEEE.

Beynon, M. et al., 2015b. Reflections on making construals as a new digital skill. University of Warwick, UK, September 2015. Online. Available at: http://go.warwick.ac.uk/em/construit/year2/c2/itagtutorial/reflectionsontutorial/ [Accessed 25 September 2015]

Beynon, M. & Harfield, A., 2010. Constructionism through construal by computer. In *Constructionism 2010*. Paris, France. Available at: http://eprints.dcs.warwick.ac.uk/242/ [Accessed July 8, 2015].

CS Unplugged, 2015. Computer Science Unplugged. Available at: http://csunplugged.org/ [Accessed July 8, 2015].

Granger, C., 2015. Coding is not the new literacy. Available at: http://www.chris-granger.com/2015/01/26/coding-is-not-the-new-literacy/ [Accessed July 8, 2015].

Hatch, M., 2014. *The maker movement manifesto*, McGraw-Hill Education. Available at: http://cds.cern.ch/record/1643837 [Accessed July 8, 2015].

Hevner, A.R. et al., 2004. Design Science in Information Systems Research. *MIS Quarterly*, 28, pp.75–105.

Hodges, A., 2002. Alan Turing. Available at: http://plato.stanford.edu/entries/turing/ [Accessed July 8, 2015].

James, W., 1975. *The Meaning of Truth (1907)* New Ed., Harvard University Press. Available at: https://books.google.com/books?hl=en&lr=&id=FVRvjj7LRAQC&pgis=1 [Accessed July 8, 2015].

Papert, S., 1980. Mindstorms: children, computers, and powerful ideas. Available at: http://dl.acm.org/citation.cfm?id=1095592 [Accessed July 8, 2015].

Polya, G., 2014. *How to Solve It: A New Aspect of Mathematical Method: A New Aspect of Mathematical Method*, Princeton University Press. Available at: https://books.google.com/books?hl=en&lr=&id=X3xsgXjTGgoC&pgis=1 [Accessed July 8, 2015].

Roe, C. & Beynon, W., 2007. Dependency by definition in Imagine-d Logo: applications and implications. Available at: http://eprints.dcs.warwick.ac.uk/252/ [Accessed July 8, 2015].

Tomcsanyi, P., 2003. 39. Implementing object dependencies in Imagine Logo. Available at: http://matchsz.inf.elte.hu/colabs/Porto/pubs/TP.pdf [Accessed July 8, 2015].