Making construals and the educational reconstruction of the computer science education curriculum

Meurig Beynon
Computer Science
University of Warwick
Coventry CV4 7AL, UK

ABSTRACT

The notion of 'educational reconstruction' of the curriculum for computer science education (CSE) has been an emerging focus of interest over recent years [6,7,14]. This paper considers how a new approach to computing, based on 'making construals', can contribute to our understanding of several challenging issues that are associated with the educational reconstruction of CSE. These include: seeking principles that are better matched to the broader vision of computing that this reconstruction promotes, finding more effective ways to represent the informal – and potentially confused – perspectives that must inform the reconstruction, and reconciling the constructivist spirit of educational reconstruction with the conventional rational perception of the computer as an 'accessible ontological reality' [1]. The concept of 'making a construal', as presented in this paper, which echoes David Gooding's introduction of the term 'construal' in his account of the principles behind Michael Faraday's pioneering experimental work on electromagnetism [10,11], also establishes significant links between computing and STEM disciplines.

CCS Concepts

• Social and professional topics→Computing education.

Keywords

Computer science education; science education; constructivism; making construals.

1. INTRODUCTION

A framework within which to conceive the 'educational reconstruction' of the computer science education (CSE) curriculum was proposed by Diethelm et al in [6] (see Figure 1). Understanding the conceptions - and misconceptions - that students bring to the study of computer science plays a key role in this reconstruction. In a complementary study with other co-authors [7], Diethelm illustrates how this applies to secondary school students' conceptions of how the Internet works. The aspiration in their study was "to find out the nature of these conceptions and possibly discover models which could be used for the planning of lessons". This aspiration is conservative in that it relates to the way in which the established "science of computing" might be more effectively conveyed. Yet the discussion in [7] implicitly raises a more fundamental concern: to what extent is educational reconstruction consistent with the accepted content and orientation of the CSE curriculum?

In [7], Diethelm et al refer to the 'constructivistic point of view' that informs their study of student conceptions of the Internet: "students create their own conceptions to explain the phenomena they perceive while acting in different contexts". The extent to which a constructivist perspective can be brought to bear on CSE is the principal theme of Ben-Ari's paper "Constructivism in computer science education" [1]. In considering the role that students' conceptions of an artifact such as a word-processor might play in a constructivist approach to CSE, Ben-Ari draws attention to the fact that "unfortunately, but perhaps inevitably, many users construct nonviable models". He concludes that "the creator of the artifact employed a very detailed model and the learner must construct a similar, though not necessarily identical, model" and that the knowledge associated with using such a software artifact "is not open to social negotiation". Computer science students must be educated to acknowledge the computer as "an accessible ontological reality".

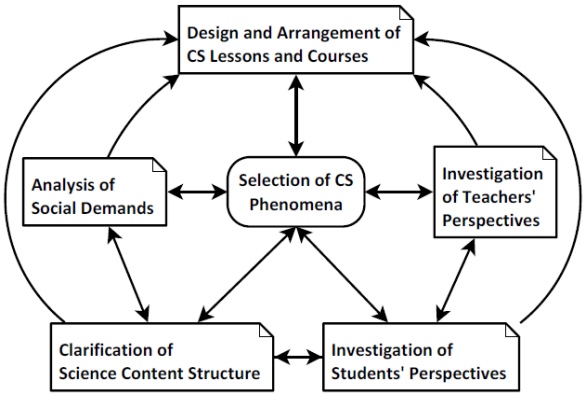

Figure 1: A Model of Educational Reconstruction for CSE, [6]

This paper builds on previous work that challenges the perception that CSE is of its essence ill-suited to a constructivist treatment [4]. Computer science has been dominated by the elaboration of Turing's computational vision for a 'mind following rules' [22]. Such a perspective of course has quite fundamental importance in contemporary computing and (for instance) motivates the significant trends towards new science that seeks to interpret empirical data in computational terms. Notwithstanding such important motivations for 'computational thinking', this paper argues that a complementary view of computer application is equally topical: one that takes account of the 'mind making sense of a situation'. Principles and instruments for 'making construals' are here promoted as the most appropriate approach to exploiting the computer in this role. The dissemination of the concept of making construals in schools education is the primary focus of an ongoing Erasmus+ project [3]. More background on this work can be found in other publications [2,3,4,16].

The paper has three principal sections. The first is an overview that revisits the question of whether CSE can embrace a constructivist outlook. The second introduces the concept of a construal and explains why making construals is relevant to the goals of educational reconstruction as applied to computer science education (CSE). The third section illustrates the practice of 'making construals' with reference to a simple example that has been deployed in a classroom setting at KS3-4. The potential implications for CSE are discussed in a concluding section.

This paper draws on work carried out in connection with CONSTRUIT!, a project that 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.

2. CONSTRUCTIVIST CSE?

In [1], Ben-Ari suggests that, where potential for a constructivist approach is concerned, computer science education is unlike science education. This section frames a counterproposal. It argues that the perception that computer science cannot adopt a constructivist approach is a side-effect of an established conception of computer science that is too narrow. The idea of the computer as 'an accessible ontological reality' is *itself* a construction. In effect, the science of computing has been developed in such a way that its core theory and practice is premised on presumptions about an objective agency for the computer and its associated technologies. Computer science is interpreted as unsuited to constructivist treatment in the first place because of the way in which the discipline is conceived and subsequently because of the way in which this conception has shaped its practice. For instance, the perception that abstraction might be the key to computing [19] is predicated on the notion that the agency of a computer is such that it admits formal interpretation.

Some relevant sources that inform this counterproposal (cf. [4]) will be briefly summarized here. They include:

- the distinguished software consultant Michael Jackson's reflections on the limitations of formal specification and verification of software [18], and the thesis about the essential role for empirical 'real-world' activities in software design that underlies his work on 'problem frames' [17]. Though Jackson does not advocate any fundamental shift in perspective on computer science, he highlights how problematic fundamental concepts of computational thinking, such as 'decomposition' and 'abstraction' can prove to be in practice.

- the controversial proposals for 'web science' made by Susan Halford et al [15], drawn up by computer science and social science academics in consultation. The internet is a domain whose semantics is difficult to conceive without invoking social construction. Finding a coherent way to reconcile the constructivistic perspective of social science with a conventional computer science perspective is challenging, A science of the web stands in a thought-provoking relation to Ben-Ari's observation in [1] about students having to learn the detailed model of "the creator of the artifact".

- new approaches to teaching programming such as have been proposed by Bret Victor [23] and Chris Granger [13], which – perhaps taking some inspiration from emerging practices in web development – seek to link the creation of software closely to the learner's direct experience of its effects. Granger's contention that 'coding is not the new literacy' is particularly relevant in this context, as it implicitly invites the question "what then is the new literacy?', to which this paper ventures 'making construals' as a plausible answer.

- the discussion that is emerging in social studies about the role that software has come to play in educational practices ("Code Acts in Education" [24]). An issue of particular concern to this community is the extent to which it is necessary and appropriate to understand coding in order to appreciate the implications of such developments. As digital technology comes to play an ever more pervasive role, as for instance in 'intelligent' devices that automatically regulate medical conditions such as diabetes [8], there is an urgent need for some account of the role of software that is more intelligible in human and social terms than program code.

The above concerns are reinforced by the author's many years of experience of teaching computer science at university, both from a traditional perspective and with a specific focus on making construals. A key motivating idea is that the 'scientific' account of a piece of software may be far from the most appropriate way in which to appreciate its significance. For instance, as a source of insight into its true nature and meaning, a formal account of the source code of a spreadsheet totally misses the mark. To make construals is to attend primarily to what the computer offers by way of 'something to be experienced'. A useful parallel may be drawn with what the performer attends to when playing a musical instrument – as contrasted with what the sound engineer studies via the digital representations that are exploited in a synthesiser.

From a pedagogical perspective, it is important to ask whether there are motivations for studying students' perspectives that go beyond what Diethelm [7] characterizes as "[being] aware of the conceptions (compatible with the scientific view or not) that the students bring into the classroom". In introducing the concept of educational reconstruction for science education in [9], Duit cites evidence that "Experiences show that the surprising and seemingly "strange" conceptions students own may provide a new view of science content and hence allows another, deeper, understanding". Is it conceivable that – contrary to Ben-Ari's thesis in [1] – similar benefits might be seen in taking student 'strange' conceptions seriously in a CSE context? Certainly, there are circumstantial reasons to suppose that a good grasp of academic computer science and skill in developing software are not necessarily well correlated: students can be very accomplished in software development and in other aspects of ICT practice on the basis of informal understanding (cf. the correlation between academic and performing musicians).

3. MAKING CONSTRUALS

This section introduces the notion of a 'construal' as an interactive artifact, potentially but not necessarily computer-based, that can be seen as embodying the understanding of its maker. In keeping with the use of the term introduced by Gooding in [10], the archetypes for such construals are the imaginative 'constructions' that Faraday developed in the course of his pioneering experimental work on electromagnetism. Gooding characterises such construals as "proto-interpretative representations which combine images and words as provisional or tentative interpretations of novel experience" [12] . It is quite natural to treat 'making construals' as an activity that can exploit computer-based devices in a fashion that is different from that associated with computational thinking. Developing a spreadsheet in an open-ended exploratory fashion – a practice that is widely used by experimental scientists who are trying to make sense of empirical data – illustrates basic principles behind making a 'digital' construal. Central to this is the idea that the novel experience (e.g. what is encountered through probing an unfamiliar phenomenon) comes to be directly associated in the experimenter's mind with familiar experience (e.g. reviewing and manipulating numerical data that is laid out in a grid and linked by dependency relations).

Making construals can be seen as a way of applying the computer that is complementary to conventional programming. Whilst there are undoubtedly many circumstances in which it is appropriate to use the computer to carry out a specific well-defined task (exploiting its capacity to serve as an 'accessible ontological reality' [1]) there are contexts where quite a different kind of technological support is needed. For instance, in the more creative and challenging aspects of software development, the developer must engage with new experiences and contexts for development

that are not fully understood (cf. Jackson [17,18]). Moreover, this is not typically a solitary activity: it is vital to be able to communicate personal understanding that is as yet partial and provisional to others who may have complementary insight and expertise.

There are clear connections between this broader view of computing activity and the goals of educational reconstruction. In the same spirit that Jackson [17] advocates that, in developing a complex software system, the choice of decomposition has to be informed by examining potential system failures from diverse perspectives, educational reconstruction advocates that, in CSE, we must take account of the broader context within which the basic science is understood and applied – and can be *mis*understood and *mis*applied. Making construals supplies an interactive environment that is well-suited to experimenting and challenging assumptions in this manner.

More significantly, introducing such an environment requires and promotes a fundamental change in the development paradigm. Software development that follows conventional principles, based as it is on the conception of the computer as an ontological reality, is neither well-suited for creating such an environment nor well-matched to the practice that is associated with it. This motivates a yet more radical aspiration for educational reconstruction of the CSE curriculum. It is not enough to survey the students' conceptions of key topics (their 'construals') so as to take them into account when teaching from a traditional computer science perspective. We should aspire to exploit the capacity of the computer to implement digital construals that can reflect students' construals and enable them to communicate, collaboratively critique and explore them.

4. AN ILLUSTRATIVE EXAMPLE

The basic concepts and principles behind making construals will be illustrated via a simple example. This uses a special-purpose web-based environment for making construals ("the MCE") that has been deployed in workshops with teachers and schoolchildren under the auspices of our ongoing Erasmus+ project.

There are three key concepts in making construals: **observables**, **dependencies** and **agents/agency**. By way of illustration, in making a construal of a sundial:

- **observables** might include: the position of the sun in the sky, the gnomon of the sundial, the shadow cast by the gnomon, the style of the gnomon, the shadow cast by the style, the dial, the hour lines on the dial etc. These are entities of which the maker has direct experience, to which an identity can be ascribed and whose status may change.
- **dependencies** might include: the relation that connects the position of the shadow of the style with the position of the sun in the sky and the relation that connects the time of day with the place where this shadow falls upon the dial.
- **agents/agency** might include: the sun, whose changing position in the sky affects the state of the other observables, the engineer who configures the sundial, a cloud that passes across the sun or a bird who lands on the gnomon and obscures the shadow of the style.

It is most important that these concepts should be seen as flexible and negotiable in their interpretation. This is in order to accommodate the many diverse perspectives that might be brought to bear on the interpretation of a sundial. The rich resources that may be found in an encyclopaedia entry for 'sundial' [25] highlight the many possible ways in which such interpretations may be elaborated. Setting up a sundial involves a quite different engagement with its observables from reading the time. A sundial of historical interest may be incorrectly configured or incomplete and no longer be functional. There may be controversy over whether a historical artifact was a sundial, and if so how it was interpreted. A different view of agency is appropriate for an analemmatic sundial: "The gnomon is not fixed and must change position daily to accurately indicate time of day. ... Analemmatic sundials are sometimes designed with a human as the gnomon." [25]. An aspiration relevant to the theme of this paper might be to develop a family of construals that could be used to supplement the Wikipedia page for sundials at [25] and enable the reader to interact with these in an open-ended exploratory and experimental way.

The MCE can be illustrated with reference to an introductory tutorial developed for schoolchildren at KS3-4 under the auspices of the EU CONSTRUIT! project. This tutorial gives pupils experience of framing observables and dependencies so as to create the simple construal of the solar system that is depicted in Figure 2. The tutorial has also been studied and reviewed by secondary school teachers. Other publications [2,3,4,16] can be consulted for more details than are given in this brief account.

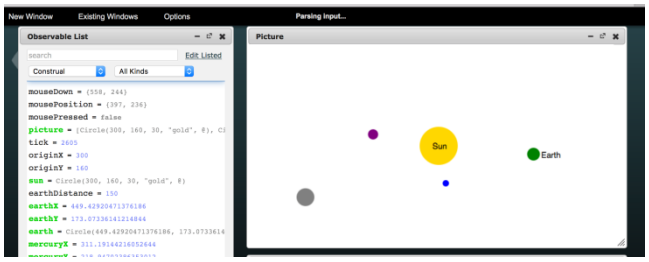

Figure 2: A basic solar system construal

The MCE, as first presented in simplified form to the school pupils, has three characteristic components: an **Input** window through which observables and dependencies can be specified (as in the middle panel of Figure 3), a **Canvas** on which visual counterparts of the observables are displayed (as in the right hand panels in Figures 2 and 3), and an **Observable List** window in which the current values of observables can be monitored (as in the right hand panels in Figures 2 and 3).

The principal activity in making a construal from scratch is the exploratory incremental introduction of observables via the input window. An observable can be assigned an explicit value, or be given a spreadsheet-style definition. The basic mechanisms required to define all the relevant observables and dependencies for the basic solar system construal in Figure 2 are set out in the tutorial. The sun is represented by a circle whose position depends on the origin for the space: moving the origin will relocate the entire planetary system visualisation. The current position of the earth depends on an observable 'tick': this is maintained by a clocking agent explicitly set up by the maker. Observables that are defined by dependency in this way are displayed in green in the Observable List (see Figure 2). The key principle in making a construal is to maintain a live connection in experience in the stream of thought between the various components in the MCE.

The experience gained from this tutorial, both by the school students and teachers, and ourselves in the role of facilitators, illustrates some of the qualities of construals in connection with the educational reconstruction of CSE agenda.

The elementary nature of a student's interaction with the emerging construal, which takes the form of a sequence of meaningful state-changing definitions, enables us to keep a complete record of the history. As described in more detail in [16], this has allowed us to analyse student's responses to the tutorial exercises and potentially gives insight into their thought processes. One of the intermediate tasks was to express the dependency that links the length of the hypotenuse of a right-angled triangle to the other sides within the MCE. By studying the steps taken by different groups, it was possible to identify problems that took diverse forms: failure to understand the mathematical relation, difficulty in framing this as a formula, and misconceptions about how to express such a relation in the MCE. The construal that was constructed to address this simple topic was recognized by educational consultants and teachers attending the workshop as a useful 'open educational resource' in its own right, with good potential for extension and customisation.

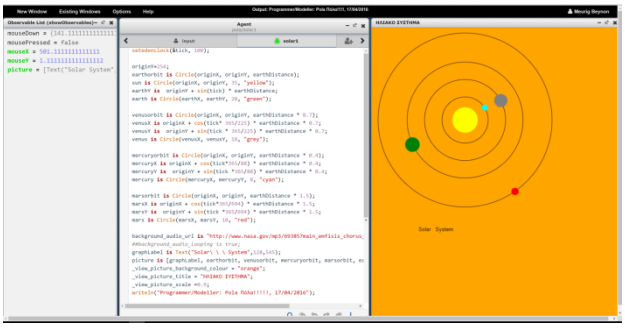

Figure 3: As adapted by a teacher at the initial workshop

The feedback from teachers has yet to be formally analysed. Informal interactions and comments suggest that it was the ICT teachers who best appreciated the motivation for 'making construals' as a way of connecting the process of developing software more intimately with acquiring understanding of the domain. Where standard construals are an established part of the discipline, as in many key topics in science, the process of creating simulations can be much more directly and simply addressed by making use of existing special-purpose educational software (such as Interactive Physics [26] or PheET [27]). For a physics teacher, the most appropriate basis for a model of the solar system is Newtonian mechanics. It is clear that, without elaboration, the principles illustrated in constructing the basic construal of the solar system are not particularly well-suited to this purpose. But whilst there are great educational merits in representing science as established knowledge of an ontological reality, there is also a crucial complementary role for making sense of novel experience in the spirit of Faraday. Similar considerations apply to Ben-Ari's perception of computer science as the study of an accessible ontological reality.

Some teachers with ICT experience were readily able to make effective practical use of modelling with observables, dependency and agency. Some devised and implemented simple extensions of the construal after their first acquaintance with the MCE. One example involved introducing a simple dependency to pause the simulation on mouse down. A more ambitious extension illustrated in Figure 3 included: depicting planetary orbits, adding sound-effects, and changing the colour, title and annotations on the canvas. Though such changes are relatively minor in character they were all carried out merely by making simple redefinitions of the kind that had been featured in the tutorial. They also include changes that you might not necessarily expect a teacher to be able to make to a simulation developed on conventional principles.

Making a construal is seen to best effect in encounters with experiences that puzzle and surprise. This was vividly illustrated during the experience of introducing the construal in the classroom situation, where the presenter himself serendipitously enhanced his personal construal of the 'epicyclic' motion of the planets as perceived from earth. The presenter motivated the tutorial worksheet by demonstrating simple examples of observables that were associated with the planetary system. These included an observable to represent the earth's orbital velocity which could be set to different values. He then intended to show that, simply by simulating the motion of the planets from an earth-centric perspective, we could observe the characteristic wandering epicyclic behavior of the planets (see for instance the trajectory of the blue circle representing Mercury in Figure 4a). In the practical demonstration, the observed simulated motion of the planet

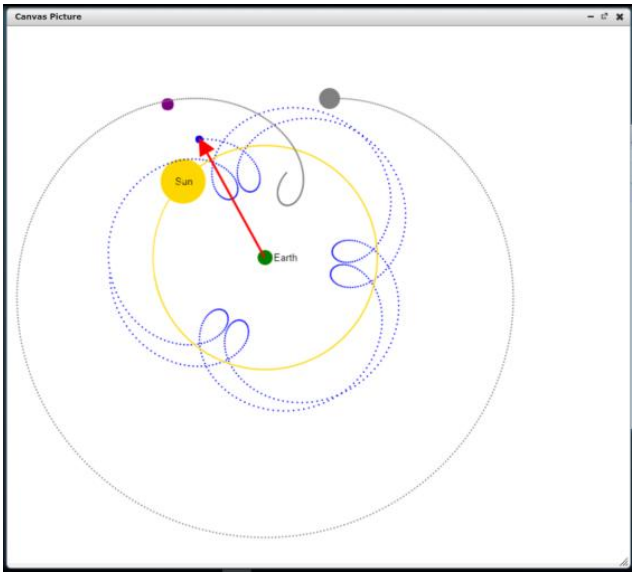

Figure 4a: The expected paths of the planets

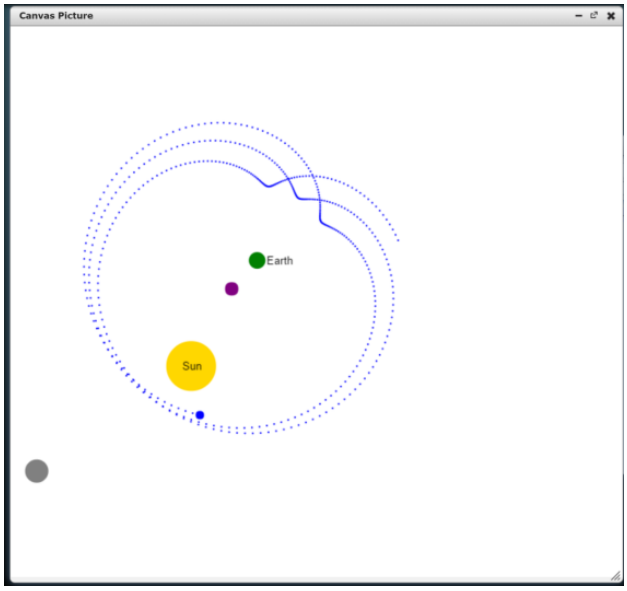

Figure 4b: The paths of the planets as displayed

Mercury was quite contrary to expectation – it did not oscillate from left to right and back again from the perspective of earth (cf. the red arrow in Figure 4a) but consistently moved in the same direction at varying speed (cf. the trajectory shown in Figure 4b). At the time, the presenter attributed this behaviour to an error in the construal, unaware that in fact Figure 4b was derived from a simulation in which the orbital velocity of the earth had been set to twice its correct value. The self-evident fact that the character of the observed motion of the planets depends critically on their orbital velocities relative to earth did not become evident to the presenter until he subsequently attempted to 'debug' the construal. The extensions of the construal illustrated in Figure 4a and 4b are a by-product of this process of more detailed investigation.

By comparison with many other construals that have been developed, the solar system construal is simple. The interactions with it that are described above nonetheless illustrate some key qualities in microcosm. A construal is unlike a program, whose relationship to its context is in aspiration prescribed in advance and in any event is conceived as matched to a stable ontological reality. The relationship between a construal and its referent is shaped by the interactions and interpretations that its maker projects upon it, evolving with the maker's understanding and current focus of attention. As illustrated above, whatever role is being played by the maker (pupil / teacher / presenter / developer) the essential character of these interactions is the same – they are all redefinitions framed in expectation of making connections in immediate experience. It is for this reason that a teacher was able to make changes to the basic construal of a kind that would normally be out of scope, such as translating the annotations on the MCE windows into Greek. Rather than being constrained by preconceived imposed realities, what we can sensibly change in a construal is determined by our capacity to make sense of change.

5. CONCLUSION

The notion of introducing making construals as a core feature of the CSE curriculum raises challenging – and at this point in time controversial – issues. To date, making construals has had modest investment and little exposure; it has only been incorporated into a computer science degree programme as an optional module for students in their fourth year of study at one UK university, where it was last taught two years ago. Against this backdrop, promoting its introduction 'for school education' is a bold idea.

Any such proposal for a new curriculum must be evaluated in relation to initiatives, such as Computing at School (CAS) in the UK, that are being introduced across Europe to promote the status and take-up of computer science in schools. The spirit of the CAS initiative is well-captured in Simon Peyton-Jones's keynote address at WiPSCE 2015 [20]. He recognises the need to emphasise *ideas* rather than technology and promotes computational thinking as an 'unplugged' conception that is 'not even primarily about computers'. The enthusiasm for computer science in this context stems from two complementary directions: appreciating the prodigious engineering feats that have established computers and associated technologies as a new 'accessible ontological reality', and delighting in the ingenuity of the abstract algorithmic processes that have been developed to exploit this. In this new vision, the place of the now deprecated skills that dominated the Information and Communications Technology ("ICT") curriculum in the UK in the 2000s is unclear.

In point of fact, the vision of computer science as a marriage of engineering reality with mathematical abstraction is an *old* vision. Contemporary computing supplies us with much more colourful furnishings in which to dress up the relationship between these two perspectives, but the relationship itself remains ill-understood. To those whose careers in CSE span several decades, the emphasis on launching CSE programmes for schools which are merely based on the traditional core university computer science curriculum conceived with this vision may induce some apprehension and a sense of *déjà vu*. For all its remarkable theoretical and practical achievements, computer science is an immature discipline that is ripe for educational reconstruction. This is most keenly evident in areas such as database design and management [21] and software design and development where the interaction between human and machine perspectives is too convoluted to be addressed by abstract separation of concerns. In such contexts, the foundational principles are not to be sought solely in a computer-like 'ontological reality' but also in the constructive craft practices that create this necessary illusion.

The anomalous status of the spreadsheet in the above discussion is highly significant. It would be entirely in keeping with the unplugged tradition of CSE to consider how to set the price of a commodity to a value that kept the expected profit below a particular threshold. This example task could be carried out in conjunction with a predefined spreadsheet to illustrate a technique such as binary search for instance. The learner would interact solely by assigning values to the price and observing the impact on profit. What gives clarity to this demonstration of an algorithmic concept is that the learner is working with concepts that they can directly connect with their experience. Suppose in contrast that a learner were to tackle the same task in a programming environment such as Scratch where, to establish the same semantic connection, they would be obliged to write the code that maintains a dependency between a price and an expected profit. Note that, from an experiential perspective, some critical aspects of this dependency maintaining behaviour (viz. the timeliness of the update of profit and the presentation of the values of the price and profit as display observables) are only implicitly and vicariously represented in the abstract program code. Such an illustrative example highlights the poverty of the pure computational semantic model in the broad context of applications where human and automated interpretations intersect.

The simplification that the spreadsheet affords in this context is clearly helpful in foregrounding the computational principle of primary interest, viz. binary search. The MCE is better suited to this purpose than a spreadsheet, in that one-off dependencies between observables can be expressed and visualized in a serendipitous open-ended way. Unfortunately, this strategy for 'simplification' does little to dispel the confusion of aspiring teachers of computing who have no specialist knowledge of the subject and want to explain to their pupils how 'unplugged' activities translate into practice. Rather than being itself a programming paradigm focused on making the most effective and efficient use of the machine, making construals is concerned with a human agenda: gaining the understanding of the application domain which informs software implementation. In practice, identifying and implementing dependencies amongst key observables in the domain plays a significant role in successful software design and development, but 'making construals' is not explicitly recognized as an independent digital skill. Why then might making construals be of interest to a teacher of computing?

The tutorial resources developed above illustrate the pragmatic solution to this concern that we have adopted in our EU project. In the first instance, a construal may be used as a simple educational resource in its own right. As outlined above, making construals can serve the somewhat perplexing role of providing electronic support for 'unplugged' activities. Teachers who make use of

prebuilt construals in this fashion may come to appreciate that, as open educational resources similar to spreadsheets, they can – to an unusual extent – be modified without specialist knowledge of programming. In due course, this may lead teachers to make construals of their own, thereby opening the door to the kind of educational reconstruction of CSE to which this paper points.

It may well be that neither CSE nor the science of making construals are yet ready to be united in a process of educational reconstruction. But whilst our computing culture is such that one of the most commited and inspirational advocates of unplugged approaches to computing freely confesses his dislike of computers [5], there is a clear need for an approach to CSE that better harmonises human and machine perspectives on computing. Without such a harmonization, our dreams for teaching and learning computing are unlikely to be realized.

6. ACKNOWLEDGMENTS

Thanks are due to Ira Diethelm for supplying valuable background references, to Stavroula Misthou for sharing the construal in Figure 3, to Jane Waite for valuable motivating observations on primary computing and to Nicolas Pope, Elizabeth Hudnott and Jonathan Foss for their technical support.

7. REFERENCES

[1] M. Ben-Ari. 2001. Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching* (2001) 20(1), 45-73

[2] M. Beynon, J. Foss, A. Harfield, E. Hudnott, N. Pope. Construing and Computing: Learning through Exploring and Exploiting Agency. In Proceedings of Constructionism 2016, February 1-5, 2016, Bangkok, Thailand, 69-78

[3] M. Beynon et al. Making construals as a new digital skill: dissolving the program - and the programmer - interface, Proceedings of iTAG 2015, 22-23 October 2015, Nottingham, UK, pp9-16

[4] Meurig Beynon, Antony Harfield and Richard Myers. Web Eden: support for computing as construction?. Proceedings of the 9th Koli Calling International Conference on Computing Education Research (ed. Pears and Schulte), Technical Report 2010-027, Information Technology, Uppsala University, Sweden, November 2010, 47-50

[5] P. Curzon. 2015. Teaching Computer Science "Unplugged". Computing at School conference, Birmingham University, UK. https://www.youtube.com/watch?v=brC4LgYefiA

[6] I. Diethelm, P. Hubwieser, and R. Klaus. 2012. Students, teachers and phenomena: Educational reconstruction for computer science education. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Koli, Finland, 164-173

[7] I. Diethelm, H. Wilken, and S. Zumbrägel. 2012. An investigation of secondary school students' conceptions on how the Internet works. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Koli, Finland, 67-73

[8] S. Doyle. 2014. How Digital Technologies Matter in Configurations of Professional Knowing: Coming to know insulin pumps in health care for paediatric diabetes. Presented at the 2nd ProPEL International Conference, University of Stirling, 25-27 June 2014.

[9] R. Duit. 2007. Science Education Research Internationally: Conceptions, Research Methods, Domains of Research. *Eurasia Journal of Mathematics, Science & Technology Education* 3, 1, 3–15.

[10] D. Gooding. 1990. *Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment*. Dordrecht: Kluwer.

[11] D. Gooding. 2007. Some Historical Encouragement for TTC: Alchemy, the Calculus and Electromagnetism. At url: http://go. warwick.ac.uk/em/thinkcomp07/gooding2.pdf

[12] D. Gooding, 2003. Experiment as an instrument of Innovation: Experience and Embedded Thought. Proceedings Cognitive Technology: Instruments of Mind: CT 2001 Coventry, UK, August 6-9, 2001 , eds. Beynon, M., Nehaniv, C.L, Dautenhahn, K. Springer, 15 May 2003, 130-140

[13] C. Granger. 2015. Coding is not the new literacy. http://www.chris-granger.com/2015/01/26/coding-is-not-the-new-literacy/

[14] A. Grillenberger, and R. Romeike. 2015. Bringing the Innovations in Data Management to CS Education: an Educational Reconstruction Approach. In *Proceedings of WiPSCE 2015*, ACM ISBN: 978-1-4503-3753-3, 88-91

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

[16] A. Harfield, R. Alimisi, P. Tomcsanyi, N. Pope, M. Beynon. Constructionism as making construals: first steps with JS-Eden in the classroom. In Proceedings of Constructionism 2016, February 1-5, 2016, Bangkok, Thailand, 42-52

[17] M. Jackson. 2000. *Problem Frames: Analysing & Structuring Software Development Problems*. Addison-Wesley.

[18] M. Jackson. 2006. What Can We Expect From Program Verification? *IEEE Computer*, 39(10):53–59, Oct. 2006.

[19] J. Kramer. 2007. Is abstraction the key to computing? CACM Volume 50 Issue 4, April 2007, 36-42

[20] S. Peyton-Jones. 2015. The dream of a lifetime: shaping how our children learn computing, Keynote presentation at WiPSCE 2015, King's College London, November 2015. http://www.wipsce.org/2015/Presentations/Peyton-Jones_WiPSCE_Presentation.pdf

[21] M. J. Ridley. (2003). Database Systems or Database Theory - or 'Why Don't You Teach Oracle". LTSN-ICS Workshop on Teaching Learning and Assessment in Databases (TLAD), Coventry, UK.

[22] A.M. Turing. 1936. On computable numbers, with an application to the Entscheidungsproblem. *J. of Math*, 5, 345–363.

[23] B. Victor. 2012. Learnable Programming: Designing a programming system for understanding programs. http://worrydream.com/LearnableProgramming/

[24] B. Williamson. 2013. The 'Code Acts in Education' seminar series. https://codeactsineducation.wordpress.com/seminars/

[25] https://en.wikipedia.org/wiki/Sundial

[26] Interactive Physics: Physics simulation software for the classroom. http://www.design-simulation.com/IP/Index.php

[27] PhET. Interactive simulations for Science and Math. https://phet.colorado.edu/