

Constructionism through construal by computer

Meurig Beynon, wmb@dcs.warwick.ac.uk
 Department of Computer Science, University of Warwick, UK

Antony Harfield, ant@dcs.warwick.ac.uk
 Department of Computer Science, University of Warwick, UK

Abstract

Traditional computer *programming* is not well-aligned to the needs of constructionism. Orthodox programming principles are oriented towards prescribing processes that address clearly specified uses. Functional specification and optimised execution do not encourage interactive exploration and open-ended interpretation. We propose making *construals* by computer using *Empirical Modelling* principles as an alternative to conventional computer programming. The merits of this approach are discussed and illustrated using construals for Sudoku solving.

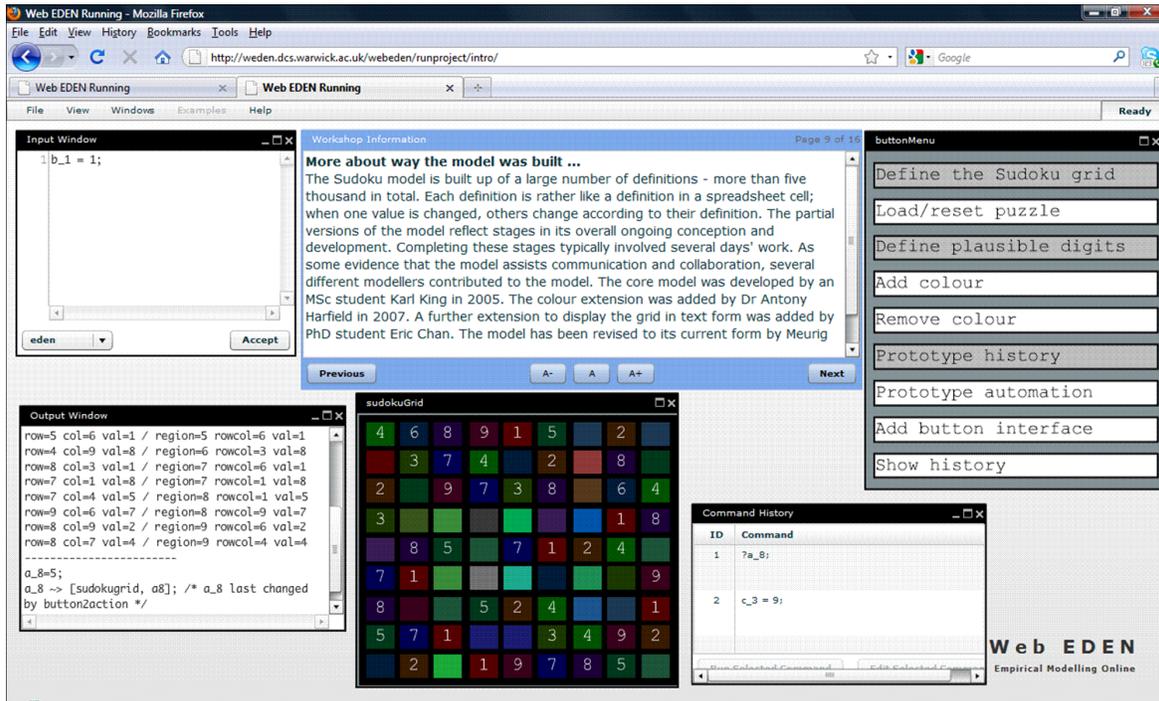


Figure 1. A screenshot depicting the online Sudoku solving construal

Our Sudoku solving construals are made up of definitions that express *dependencies* between *observables*. Many kinds of human agency can be expressed through modifying the current set of definitions. The construal serves as a shared artefact with which developers, teachers and pupils can all interact concurrently in essentially the same way, each according to their role and experience. Our preliminary experiments with schoolchildren highlight potential for rich and radically new kinds of learning experience and unprecedented scope for recording, monitoring and intervening in support of constructionist learning. Further empirical study is a vital next step.

Keywords

Constructionism, programming, learning technology, Empirical Modelling, construal, dependency

Constructionism and Computing

As Richard Noss (2008) observes in his *Open Letter to Logo and EuroLogo Communities*, Seymour Papert introduced **constructionism** as a name for "a pedagogy based on building and sharing physical, virtual and intellectual structures". This characterisation makes no reference to computers, but in practice it was the computer, and specifically the use of Logo programming, that launched the concept of constructionism. No educational movement has since contributed more to the cause of constructionism than the Logo community. Yet, as Noss also remarks, "the fundamental hope of Logo's creators and its later adherents remains largely unfulfilled".

The challenges for constructionism as a computer-based activity mirror those facing computer programming within computer science. The *Call for Papers* highlights the fact that "The developers of Logo and similar computational environments have ... encouraged learners to better understand the world and their place in it by building their own meaning-making models based on iterative, interactive exploration and testing of ideas and notions." It endorses a vision for constructionism in which, in the words of one conferee: "I don't see any hard edges between creating, sharing, consuming and learning. I want a system that allows people to shift effortlessly between doing these things."

As computing technology matures, and computing systems become ever more complex, so "constructionist" activities in this spirit become more relevant for software development. Agile development methodologies embrace the notion of "iterative, interactive exploration and testing of ideas and notions." They demand environments in which developers can play many different roles - building, learning, communicating - and "shift effortlessly between doing these things." But even if we look beyond programming paradigms and languages based on Logo, the vision for efficacious software engineering of this kind also "remains largely unfulfilled". Even within established computer science, the problem of giving computing support to constructionist principles is unresolved (Ben-Ari, 2001; Beynon and Harfield, 2007; Beynon, 2009).

We believe the underlying problems faced in educational technology and in complex systems development have a common root. They both relate to the difficulty of aligning computing programming with learning, as is essential both in the classroom and in the software house.

In this paper, we illustrate an alternative approach to developing software that we believe offers much better prospects for supporting constructionism. To reflect the radically different orientation of this approach, we conceive constructing software as 'making a construal' rather than 'developing a program'. A **construal** is a computer-based artefact similar in character to the structures conceived by Papert, but with essential qualities that are not apparent when we interpret it as a conventional "computer program". A construal admits myriad interactions that are not preconceived, for instance, and for which there may be no specified *a priori* use or interpretation. The interpretations it supports depend in general on the experience and the skill of the individual human agent who is interacting with it. On this account, they also depend critically on qualitative and experiential aspects of the construal – how its state is communicated and perceived, how it can be manipulated and how quickly it responds.

One reason why conventional programs - such as the Logo programs originally studied by Papert – fail to support constructionist learning as powerfully as we might like is that they are built with quite different objectives in mind, and according to quite inappropriate principles. For instance, in order to make practical progress, computer programmers must specify the interactions that their programs support and align these to specific purposes. To this end, the programmer abstracts automatable patterns of interaction and interpretation ('uses') from the environment. By contrast, a construal is a source of concrete open-ended experience that is not typically intended to be understood in isolation from its environment. In this respect it resembles a spreadsheet or a database in which the symbolic data stands in an intimate relation to meaningful – and current - entities in the external world. When separated from the external

referent, or no longer current, such symbolic relationships lose their significance. Interactions with spreadsheets and databases thus directly address sense-making.

A construal for Sudoku solving

The construal we shall study relates to the process of human solving of Sudoku puzzles. Of its nature, a construal is an organic artefact that evolves over time as it is deployed by different interpreters. We build construals using Empirical Modelling (EM) principles that we have developed over many years. A full discussion of EM is beyond the scope of this short paper – interested readers can consult the EM website for more details. The principal construal used for illustrative purpose is available online, and other variants of this Sudoku construal can be downloaded from the EM archive.

The key elements in a construal are **observables**, which represent meaningful entities that have an identity and current status or value in the referent, and **dependencies**, which reflect perceived connections between observables similar to those that link cells in a spreadsheet. As the word ‘perceived’ suggests, the construal is to be understood in conjunction with a human agent acting within an environment that encloses or evokes an external referent. There is no clearly specified set of appropriate processes and states associated with a construal. The states of the construal are intended to evolve in intimate conjunction with the states of mind of its human interpreter – or more precisely with those of its human *interpreters*, since there is no absolute constraint on the interpretative role that human agents can adopt in interaction with its current state. Meaningfulness to the human interpreter is all that constrains the evolution of the construal, which lends an open-ended exploratory character to its construction. This does not rule out the possibility that patterns of interaction and interpretation emerge, some of which may be automated to realise program-like functionality.

The Sudoku construal – or to be more precise, a *state* of a Sudoku construal – is depicted in Figure 1. Because of the highly interactive nature of construals, it would be helpful at this point for a reader unfamiliar with EM to invoke this construal via the online ‘General Introduction to the Script’ at <http://www.dcs.warwick.ac.uk/~wmb/sudokuExperience/workshops/>. As explained in that introduction, the underlying structure of the construal is a moderately large set of definitions (some five thousand in all), each of which specifies the value of a different observable either explicitly, or by a formula in terms of other observables. The size of the set of observables is largely due to the fact that each observable associated with a particular cell of the Sudoku grid has a counterpart in each of the other 80 cells. Examples of observables include:

```
d_3 = 7;          ## the contents of the cell D3 in the 3rd row and 4th column of the grid
d3_fixed = 1;    ## the status of the value in this cell – given in the puzzle, so fixed
d3 is mkstr(d_3); ## the string that is displayed to indicate the value in the cell
## ... the x and y coordinates of the cell D3 in the screen display:
D3_X1 is grid_startx + column(3.0) + (3.0 * spacer_x) + 2.0;
D3_Y1 is grid_starty + row(2.0) + (2.0 * spacer_y);
## ... the foreground and background colours of the cell D3:
D3_fgcolour is d3_fixed ? SD_fixed_fgcolour : SD_fgcolour;
D3_bgcolour is colourclue(possibledigits2binary(possdigit34),colourvalues);
```

The construal is built using the EDEN interpreter, depicted in Figure 1 in its online variant. Definitions are first entered through the Input Window at the top left, either individually, or through file inclusion. The current values and definitions of observables can then be queried and displayed in the Output Window at the bottom left. Initially, before any definitions have been entered, the interpreter affords only these two windows; other components of the display in Figure 1, such as the Sudoku grid and the ButtonMenu panel at the top right are themselves specified by sets of definitions via the Input Window. In Figure 1, the grid and button menu

their location on the screen. We can also see that the background colour of cell D3 depends on the observable `possdig34`, which records the set of *plausible* digits for a cell, given its status and that of cells in the same region, row or column. This dependency is the focus of a later section.

The dependency net may appear to be static and structural in character, but in fact it is dynamic and fluid. At first sight, it seems obvious that the dimensions of a cell should be independent of its location, but – in fact – it might assist a partially-sighted person if the grid could be moved around so that the cell at the centre of the screen was enlarged. Of course, what changes can be made to the dependencies are constrained if the construal is to represent a Sudoku grid. This is no more than what is expected of structure that is negotiated through experience and established by convention. In this respect, a construal provides a representation that not only favours a constructionist stance, but is in the broader sense *constructivist* in spirit (Latour, 2006).

Simple redefinitions play a vital – though sometimes hidden – role in the Sudoku construal. For example, when a solver focuses on a specific cell (say E2) and enters a digit (say 6) from the keyboard, they in effect instruct automated agents to redefine the current cell, thereby defining an observable (`e2_focus`) to indicate that E2 is currently selected, and to make an appropriate redefinition (`e_2 = 6`). More elaborate sets of redefinitions are associated with a shift in perspective on the solving task, such as is involved in switching in and out of the ‘colour Sudoku’ mode (cf. Figures 1 and 3). The complex reconfiguration of dependencies invoked when the ‘Remove colour’ button is pressed can be seen by contrasting 2(a) with 2(b). Setting up these different modes and patterns of agency for interaction involves the iteration and testing of ideas characteristic of constructionism. Once created, they can then be recorded and replayed.

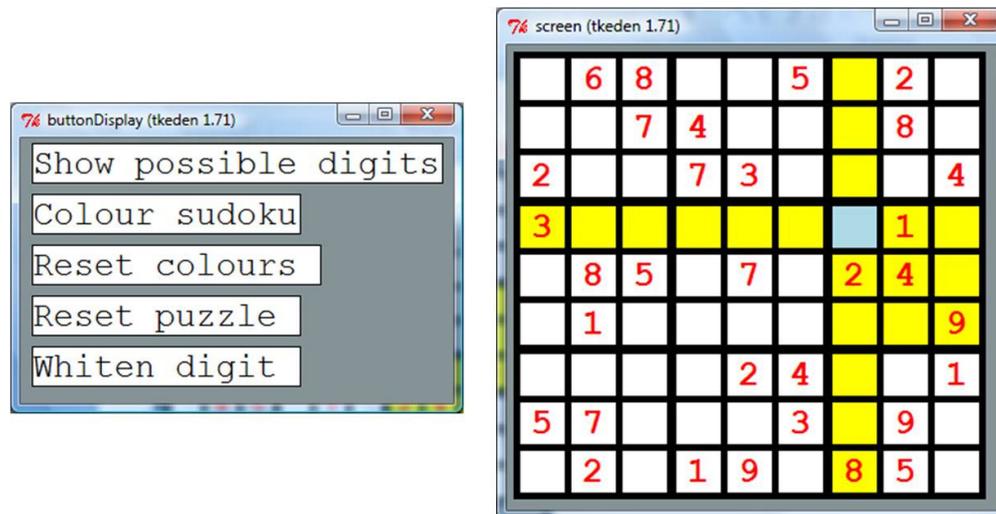


Figure 3. The initial state of the Sudoku solving environment for the ACE sessions

Blending learning, teaching and development

At the heart of constructionist education is the idea of shifting effortlessly between the work of the learner, the teacher and the developer. The distinction between these roles is very stark in conventional programming. In construals, by contrast, the net of observables and dependencies serves as a playground where many agents can act, potentially even concurrently. What distinguishes the roles of agents is their expertise and level of privilege where interpreting and modifying definitions is concerned.

The potential for blending roles that construals afford has been illustrated informally in educational activities we have carried out at the University of Warwick using our Sudoku-solving construals. The first of these was in connection with two short visits by local schoolchildren

under the auspices of the "Aiming for a College Education (ACE)" programme in January and February 2008 (see Figures 3 and 4). The second, in July 2008, was a week-long workshop entitled "The Sudoku Experience" for pupils on the UK Young Gifted and Talented (YGT) programme (see Figure 1). Further feedback on this workshop has been given by Daria Antonova, a high-school student affiliated to the Nokia Toijala Center scheme in Finland.



Figure 4. The state of the Sudoku solving environment after colour has been introduced

The variant of the Sudoku construal used in the ACE activity highlights some of the ways in which the perspective of developers, teachers and learners can be blended. The educational objective was to expose pupils to abstract reasoning and problem-solving in a way they might find entertaining. The basic construal on which the activity was based was initially developed by an MSc student Karl King in 2006 (cf. Figure 3), and subsequently elaborated to include a cell colouring by Harfield in 2007 (cf. Figure 4). These were combined into a single artefact by Beynon for the ACE workshop, when the simple button interface in Figure 3 was added. This process of re-use and adaptation is characteristic of our construals and illustrates the scope for sharing and communication in development that EM affords.

The adaptations of King's and Harfield's construals for the ACE activity are typical of those that a teacher might make. They introduce interfaces that disclose observables and allow them to be manipulated in appropriate ways. The basic affordances in the button menu in Figure 3 make it possible to inspect the plausible digits for the currently selected cell and to reset the puzzle. In a routine solving process, the solver looks for cells that admit *just one* plausible digit, or instances of rows, columns or regions in which there is only one cell in which a specific digit can be placed. We introduced these abstract rules to the pupils before they used the construal.

Several skills are engaged in applying these simple rules. The yellow highlighting of cells in Figure 3 helps to maintain focus on the digits that lie in the same row, column and region. A teacher who wished to assess a pupil's skill in identifying such relevant cells might remove these highlights by redefining a single observable – `SD_relevant_colour`. Once pupils are skilful in such identification, colour can be exploited in another way, as illustrated in Figure 4.

The basic principle behind "colour Sudoku" is that a colour can be associated with each empty cell so as to reflect the 'plausible digits' for the cell. For the cell B7 in Figure 4, for example, the set of plausibles is [3,9] because 3 and 9 are the only digits not found in its row, column or region. Each empty cell can then be coloured according to its set of plausibles. One way of doing this is to assign a unique colour to each digit and then colour the empty cell as a mix of the

colours of its plausibles. In the Sudoku construal the resulting colour for a cell is obtained by adding together the RGB values of the assigned colours for each of the possible digits.

The two basic rules for Sudoku solving identified above have interpretations in colour Sudoku. The colour of a cell having only one plausible digit (such as E1 in Figure 4) has the same colour as that digit. The fact that there is only one location in which a particular digit can be placed within a row, column or region can be disclosed by modifying the colour associated with that digit and observing which cells are then affected. The interface in the left panel of Figure 4 enables the colour associated with any specific digit to be modified. An application of this rule is then illustrated in Figure 4, where the R component of the colour associated with the digit 4 is being enhanced, and the impact on cells of the grid observed. In this case, the cell A1 is the only cell in the top row that changes colour.

Extensions to King's original construal of the above kind are of interest to the teacher in several ways. They can be used to enrich the learning experience for the pupil, or to probe the nature of a pupil's difficulties. They can also be used in gathering insight into Sudoku solving prior to framing activities for pupils. When solving puzzles without computer support, a solver may survey the plausible digits for cells and commit them to memory, or record the plausible digits as a list in each cell. To some degree, efficient solution of puzzles appears to rely on good fortune in identifying cells to which one of the basic rules applies. Studying colour Sudoku helps to expose issues that relate to perception, cognition and memory in the solving task.

Because of the brief and cursory nature of their visit, we had no opportunity at the time to study the pupil's reactions to the ACE sessions in a formal way. A feature of the Sudoku construals is that the entire history of interactions associated with each individual pupil is recorded as a sequence of redefinitions of observables, which can then be replayed. Figures 3 and 4 are screenshots from just such a replayed sequence of interactions on the part of one pupil. What is more, because each state constructed from the history is retrieved as a set of definitions, it serves as an interactive environment in which the analyst can also interact. This is also illustrated in Figure 4 – the display of plausible digits at the bottom left was not in fact invoked by the pupil who created the history, but was added when the history was replayed.

From our preliminary studies of these histories, and our informal observation at the time, it was apparent that the range of pupil reactions was broad. For some pupils familiar with Sudoku, the elementary puzzle in Figure 1 proved too simple. Others – who perhaps lacked experience and motivation for puzzle solving – did not understand the task, but seem to have completed the grid in a random manner so as not to lose face. One pupil chose not to use the colour Sudoku interface, maintaining that they preferred the presentation in Figure 3 to that in Figure 4. A virtue of the openness of the construal is that it enabled us to adapt to special circumstances. For instance, our button interface made no provision for changing the underlying Sudoku puzzle, but we could show the more advanced pupils how to load different puzzles via the Input Window.

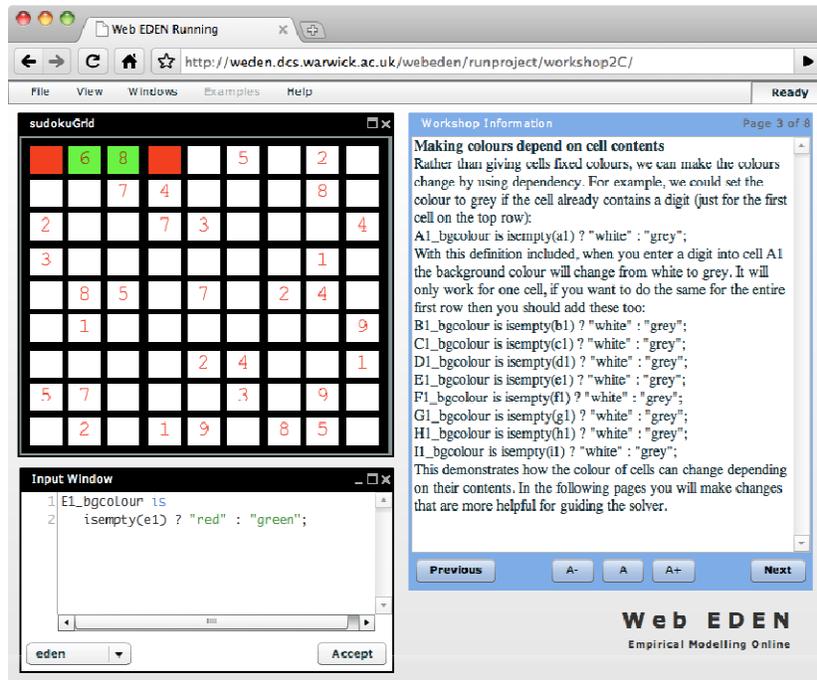
The *Sudoku Experience* workshop

The *Sudoku Experience* workshop consists of a number of activities to be undertaken inside the online Web Eden environment. Each activity starts with the Sudoku construal loaded in a specific state together with a guidebook for the student to follow or work through (see Figure 1). For some of the activities the guidebook resembles a tutorial (e.g. in early activities the student is taught how to use the environment). For other activities the guidebook is a guided exploration of the construal (e.g. to give insight into essential elements of the construal). In other contexts the guidebook proposes open-ended creative tasks for the student to undertake (e.g. building an extension to the construal by modifying dependencies). In this way, the environment is utilised for both 'instructionist' and 'constructivist' learning. In order for a student to progress to a deeper level of learning in these workshops it seems to be necessary for exploratory activities to be preceded by activities of a more closed and tightly prescribed nature.

The workshop had three core elements: playing, exploring, building. Students began on day one by playing Sudoku. This could have been on paper or using the Sudoku construal. In its basic state, the Sudoku construal appears to be nothing more than an interface for playing Sudoku (much the same as the many Sudoku programs that are freely available). However, the vanilla construal provides an interface for adding layers from which a student can begin to explore ways to solve a puzzle. For example, a student might choose to show the plausible digits for a particular cell as shown in Figure 4. The set of plausible digits for a cell is an observable in the construal (cf. `possdig34` in Figure 2) that is defined by a dependency. This is one of many dependencies that are already given to the students in the Sudoku construal for them to explore.

An important feature of a construal is that it is always live. There is no separation between a 'build' mode and a 'play' mode. A construal is open to exploration and redefinition at the same time as a student is using or playing with it. In the Sudoku construal, a student might start playing by entering some digits into the puzzle, but then explore what this means in terms of the underlying dependencies, or try to add new dependencies to derive new insight into the puzzle. One of the activities that involved adding colour to assist solving serves as a good example.

By midway through the workshops, the students were familiar with solving Sudoku and able to identify and write dependencies that describe simple rules for solving Sudoku. They had also been introduced to the colour version of the Sudoku construal. At this point we gave the students an opportunity to construct their own version of colour Sudoku.



The screenshot shows a web browser window titled "Web EDEN Running" with the URL `http://weden.dcs.warwick.ac.uk/wabeden/runproject/workshop2C/`. The interface includes a menu bar (File, View, Windows, Examples, Help) and a "Ready" status indicator. The main content area is divided into three sections:

- sudokuGrid:** A 9x9 grid with some cells containing numbers. The top row has cells with numbers 6, 8, 5, and 2. The grid is partially filled with numbers, and some cells are highlighted in red and green.
- Input Window:** A text area containing two lines of code:


```
1 E1_bgcolour is
2 isempty(c1) ? "red" : "green";
```

 Below the text area is a dropdown menu showing "eden" and an "Accept" button.
- Workshop Information:** A panel titled "Making colours depend on cell contents" with the following text:

Rather than giving cells fixed colours, we can make the colours change by using dependency. For example, we could set the colour to grey if the cell already contains a digit (just for the first cell on the top row):

```
A1_bgcolour is isempty(a1) ? "white" : "grey";
```

With this definition included, when you enter a digit into cell A1 the background colour will change from white to grey. It will only work for one cell, if you want to do the same for the entire first row then you should add these too:

```
B1_bgcolour is isempty(b1) ? "white" : "grey";
C1_bgcolour is isempty(c1) ? "white" : "grey";
D1_bgcolour is isempty(d1) ? "white" : "grey";
E1_bgcolour is isempty(e1) ? "white" : "grey";
F1_bgcolour is isempty(f1) ? "white" : "grey";
G1_bgcolour is isempty(g1) ? "white" : "grey";
H1_bgcolour is isempty(h1) ? "white" : "grey";
I1_bgcolour is isempty(i1) ? "white" : "grey";
```

This demonstrates how the colour of cells can change depending on their contents. In the following pages you will make changes that are more helpful for guiding the solver.

At the bottom of the workshop information panel are navigation buttons: "Previous", "A-", "A", "A+", and "Next". The footer of the interface reads "Web EDEN Empirical Modelling Online".

Figure 5. The Sudoku construal as used for the colour creation activity

The colour Sudoku activity acquaints students with the way in which the visual colour clues in Figure 4 are specified, and equips them to explore alternative ways in which these could have been specified. It begins by introducing the observables for defining the background colour of a cell. Students then start by experimenting with the dependencies that define the colour based on the value in the cell, as shown in Figure 5. In previous activities, students had already explored the observables for determining the possible digits of a cell. Using this knowledge we set them the task of creating dependencies that colour the cells in a similar manner to the colour Sudoku game they played at the beginning of the workshop.

The colour Sudoku activity illustrates how the Web Eden environment supports two key aspects of constructionism: helping students to achieve new knowledge and skills through disciplined guidance, and offering students opportunities to actively explore and experiment in order to develop knowledge based on their own experiences.

In the colour Sudoku activity, before attempting to ‘actively explore and experiment’ a student must be equipped with some knowledge of how to change the colour of cells and to access the plausible digits of a cell. The information window on the right hand side of Figure 5 provides disciplined guidance to colour specification in the colour Sudoku construal. It functions as a guidebook to the construal in much the way that a tourist guidebook assists a foreign visitor, pointing out places of interest, things to try, and essential facts that you cannot get by without.

With the guidebook as a reference for preconceived interactions that might be useful, the student can explore the construal and experiment by *changing* observables or dependencies. Such modifications might be achieved through a graphical interface (provided by the teacher or developer) or by entering definitions into the Input Window. In Figure 5, a new definition for the background colour of cell E1 is entered via the Input Window. In the colour Sudoku activity, the student is encouraged to experiment by setting cells to different colours in this way. An alternative colour scheme might associate the brightest colour with individual digits, and hence represent cells with many plausible digits by dark colours, for instance. Many students were able to progress to derive definitions for the colour of an individual cell based on plausible digits and thereby eventually build up a specification for the full colour grid, as shown in Figure 6.

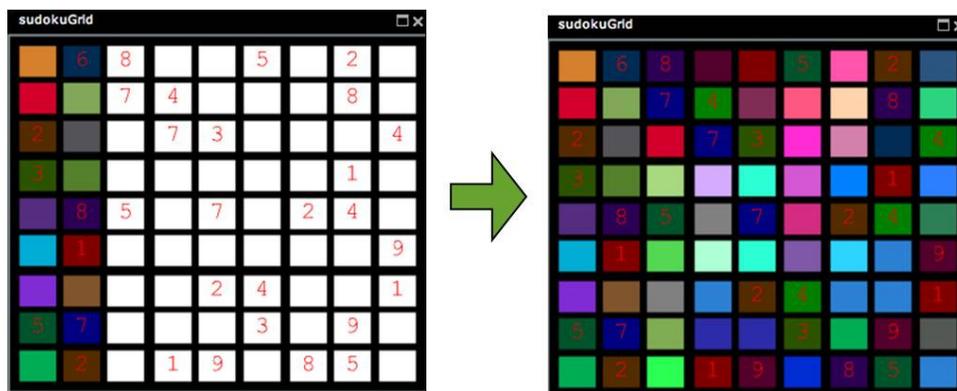


Figure 6. The Sudoku construal as used for the colour creation activity

Some feedback and reflection

Positive feedback from students during the workshop indicated that there is potential in the approach. Comments such as “I did it! It works!” suggested that students enjoyed the activities. Several appreciated the power of the metaphor of *making and following a guided walk* we had invoked. One student expressed what he liked about the activities: “It was amazing to see what we have actually done to the sudoku board and it was good that you said we could ‘wonder (sic) off the path’ a bit, e.g. changing colours and numbers, which was good fun.”

Some of the most encouraging comments were directly related to the exploratory nature of the environment: “A very good idea to have a ‘path’ but it was flexible.” Others saw the value in building up towards a larger goal: “A good building progress from each of the tasks, so at the end you can put all of them together.” Students found experimenting valuable, even when it was the smallest change: “Having lots of stuff to do in every workshop, even if it looks easy, you still feel a small achievement when you actually change a square to blue or make the number 7 green.” Students were also willing to try things out for themselves and take their learning into their own hands: “Lots and lots of colours recognized, i decided to have a little play and started

putting random colours in like magenta, turquoise etc. and was really good fun making it multicoloured". The environment enabled some students to go beyond the set tasks: "I know that the sudoku grid is supposed to give blanks when 0 is entered into a square but this does not allow other programs such as the addition program to work properly. A solution to this would be to set a key such as 'c' to a blank and tell users to press 'c' or clear to empty a box."

Many of the critical comments related to literacy demands. One criticism was that there was too much material to read: "There was a lot of writing to read during the first two tasks 18 pages and 14 pages, maybe other people feel differently, but the majority of the task was reading." And whilst some students remarked that: "it was well written and easy to understand", others struggled: "I had difficulties to knowing how to do things, as I don't think it was explain very well. In the introduction I got confused straight away but then when I went onto workshop 2 I worked out what to do. I think it needs to be made clearer how to do things." Another said: "I really didn't understand a lot of it - some of the basic stuff made sense and I think I got some of the stuff about the colours, but a lot of it went over my head. A bit of getting to know the basics first would have helped, I think if I knew more about programming, I would have enjoyed it more."

The wide range of notations in the environment confused one student: "I have run into more problems. Which (thing) should I be using; eden or scout". Another was mystified by aspects of the environment: "I also found the output box confusing and don't quite know why we need it." These issues point to a need to refine the environment for educational use. Although restricting what tools the learner has available may be undesirable from a constructionist viewpoint, it may sometimes be better to hide parts of the environment that are unlikely to be used.

Overall, the breadth of the issues addressed in the *Sudoku Experience* workshop and the open-ended interaction it provokes go beyond blending developing, teaching and learning, blurring the very boundaries between what is being taught, learnt and constructed. The student feedback points to the difficulty some face in adapting to the fuzzily-defined objectives and rich and messy interactions that discovery in a constructionist spirit can entail. Looking to the future, it is vital to distinguish construal-by-computer from the sharply-defined goals and neat rationality of programming. In these respects, Antonova's independent verdict on the workshop activities is encouraging: "They turned out to be pretty interesting and dont really require programming skills or previous knowledge of programming language, just some logic. I had to think quite a while about some of exercises to find answers but after you find them, exercises don't seem hard."

References

- Ben Ari, M. (2001) Constructivism in Computer Science Education, *Journal of Computers in Mathematics and Science Teaching* 20(1), 45-73
- Beynon, M. (2009) *Constructivist Computer Science Education Reconstructed*. HEA-ICS ITALICS e-Journal, Volume 8 Issue 2, June 2009, 73-90
- Beynon, M. and Harfield, A. (2007) *Lifelong Learning, Empirical Modelling and the Promises of Constructivism*. *Journal of Computers*, Volume 2, Issue 3, May 2007, 43-55.
- Latour, B. (2006) The Promises of Constructivism, In Ihde, D. (ed.) *Chasing Technoscience: Matrix of Materiality*, 2006, 27-46
- Noss, R. *Open Letter to Logo and EuroLogo Communities*, at the url: http://www.aup.edu/news/downloads/constructionism2010_Noss.pdf (accessed 28/03/10)
- The EM website at <http://www.dcs.warwick.ac.uk/modelling/> (accessed 28/03/10)
- The EM archive at <http://empublic.dcs.warwick.ac.uk/projects/> (accessed 28/03/10)
- The Web Eden interpreter at <http://go.warwick.ac.uk/webeden/> (accessed 28/03/10)