Analysing the structuring of knowledge in learning networks

Lucila Carvalho and Peter Goodyear

Centre for Research on Computer Supported Learning and Cognition, Faculty of Education and Social Work, University of Sydney, lucila.carvalho@sydney.edu.au, peter.goodyear@sydney.edu.au

Abstract

Knowledge plays a significant role in networked learning. Epistemic activities, and the structures that support such activities, do not just spring into life when a networked learning resource is created. They exist prior to, and outside of, any specific networked resource. They go on in the world. So epistemic activities and structures can be seen as part of a broader social-cultural context. In this paper, we argue that people who design for networked learning benefit from having a richer repertoire of ways of understanding relationships between epistemic activities and the structures that support them, and of viewing these in their social contexts. Educational designers need to be aware there are different ways of expressing knowledge, associated with implicit values that underlie knowledge practices within any social context. These varied ways of expressing knowledge have diverse effects on learners' activities. Understanding such connections is useful when designing new networked learning resources, or when devising ways to improve existing learning networks.

Using an approach from the sociology of knowledge, this paper explores the structuring of knowledge in a case study of networked learning drawn from an undergraduate design course about graphics and programming. The course uses a computer-based platform called Peep to extend what would otherwise be a set of timetabled, lab-based learning and teaching activities. Students and teachers can interact using Peep, at any time, from any location. These networked learning activities involve a mixture of lecturer-led tasks and student-driven collaborations, including requests for and offers of help, and sharing and discussing code and graphical designs. Our focus in this case study is on variations in the degree to which the knowledge being dealt with is *dependent on its context for meaning* and *condensed*. Our analysis reveals that one of the key design elements of Peep – the code editor – facilitates a distinctive, important epistemic activity. The code editor enables the 'object of discussion' (programming and the visual and animated effects of it) to come to the fore. Instead of students having to describe phenomena, events or processes that happen elsewhere, they each have the object of their discussion in front of them, embedded within all of Peep's varied spaces for learning. Our analysis explores the relations between this design feature and Peep's support for students' discussions of complex concepts, sharing and exchanging views, and building on each other's ideas and work.

Keywords

Epistemic activity; Legitimation Code Theory; Design; Coding

An architectural framework for the analysis of learning networks

Designing for networked learning does not happen in a detached socio-cultural context. When an educational designer develops "feature x" or "learning resource y", or when participants engage in networked learning, they do so from a position they occupy in a social field of practice. Educational designers and networked learners engage in knowledge-related activities prior to, and outside of, the learning network that brings them together. Knowledge-related or epistemic activities are evident when people engage in discussions, or when they write essays, summarise, critique, investigate, memorise or do many other things, all of which are related to a knowledge topic or issue of interest to them. It is important that designers in networked learning understand epistemic activity as part of the broader social context in which it occurs, so that ways of structuring knowledge are addressed in their designs. Designing for networked learning requires, therefore, analytical tools that can support multiple investigations of complex intrinsically connected elements, which include, for example, the interplay between tools, tasks and divisions of labour, and an understanding of how combinations of these elements may influence learners' emergent activities (Goodyear, 2005). Recent research by the authors

ISBN 978-1-86220-304-4

examines complex learning environments through an architecturally inspired analytical framework (Goodyear & Carvalho, 2013; Carvalho & Goodyear, in press). This work investigates how a combination of heterogeneous structural elements supports learning; that is, how a particular assemblage supports and shapes, but does not determine, learners' activities. For example, one can have a plan for the tools learners will use, the tasks they will undertake or the arrangements of learners into groups. However, the resulting assemblages are likely to be adapted and reconfigured by learners. The structural composition of learning networks can be analytically divided into (i) structures of place (or *set design*), (ii) structures of tasks (or *epistemic design*) and (iii) social structures (or *social/organisational design*) (for other research involving empirical applications of these ideas see also Pinto, 2014 and Yeoman & Carvalho, 2014; for a theoretical discussion, see also Goodyear et al., 2014). This paper focuses on epistemic design. We analyse tasks and the structuring of knowledge in a learning network, exploring how a particular assemblage may shape and influence learners' activities, and we then connect our findings to design for networked learning more broadly. Drawing insights from an approach from the sociology of knowledge - Legitimation Code Theory (LCT) (Maton, 2014) - we discuss the analysis of the epistemic design of an undergraduate learning network. We argue that designers would benefit from using an LCT-informed approach in their work, as it provides a meta-language to understand the role of knowledge in design for learning: (1) it helps designers recognise that there are different ways that knowledge may be expressed within different knowledge practices, and (2) it provides a basis for designers to work from, where they can figure out ways of enacting these differences through their designs. LCT supports analysis by explicitly articulating the organizing principles of knowledge within a learning network, enabling one to explore connections between design elements and processes related to knowledge-building. In the next section of this paper, we introduce some concepts from LCT. We then provide an overview of the Peep case study and we briefly report our data collection. In the following section, we examine the structuring of knowledge in this particular network, in light of the LCT concepts, discussing one design element that appears to be key in our case study, and analysing the tasks proposed in relation to knowledge-building.

Insights from the sociology of knowledge

Education, including networked learning, involves the production, recontextualization, teaching and learning of *knowledge*. A key resource in the sociology of knowledge is Bernstein's theory of pedagogic discourse. Bernstein viewed pedagogic practices as not limited to classroom settings, including instead, any social context where epistemic activity takes place (Bernstein, 2000). His early work centred on understanding how pedagogic processes are shaped, proposing a way to investigate the forms of communication within these processes, which included analysing "how a pedagogic text has been put together, the rules of its construction, circulation, contextualization, acquisition and change" (Bernstein, 2000, p.4). Bernstein's later work suggested ways of examining different forms of knowledge, realised through an everyday, common sense knowledge (horizontal discourse) or academic, specialised forms of knowledge (vertical discourse) (Bernstein, 2000). Legitimation Code Theory (LCT) builds on and extends Bernstein's code theory and Bourdieu's field theory (Maton, 2014; Maton et al., 2014). LCT is associated with the "school of thought" of social realism, which views knowledge as *both*: (a) socially constructed, within cultural and historical conditions, *and* (b) something that is real in its own right and takes different forms which have effects on educational practices (Maton & Moore, 2010). Thus, knowledge changes and is influenced by relations of power, but importantly, knowledge claims are not necessarily all the same in different social contexts (Maton & Moore, 2010). In line with this perspective, we see the nature of knowledge as likely to differ amongst a variety of learning networks – what counts as relevant knowledge and practices will not be the same for every network. LCT views knowledge practices as underlain by implicit 'rules of the game' that affect and shape the way knowledge is expressed or communicated in any given educational or intellectual field. They may regulate, for example, who is "entitled" to participate in a certain profession, what is to be considered an interesting insight, whose voice is legitimate and so on (Maton, 2014). We argue, in this paper, that this is an important issue for designers in networked learning, because in order to identify key reusable design elements that may inform the development of better learning environments, it is crucial that one considers the various forms that an epistemic structure may take, as well as the related effects of such structures. LCT allows us to address two important issues. Firstly, there are different ways that knowledge may be expressed through design, offering a "language" to express these differences. That is, LCT offers analytical tools to identify the organizing principles that underlie the knowledge practices enacted through networked learning (the practices of designers, teachers and learners). Secondly, LCT allows us to theorize about ways in which a specific epistemic assemblage may affect those who participate in network activities, including ways in which network participants co-configure the space. In this paper, our focus is on investigating the structuring of specific epistemic assemblages, as we theorize about the role they play in terms of cumulative learning and knowledge-building within a particular learning network.

ISBN 978-1-86220-304-4

One of the key activities in networked learning is knowledge-building. Scardamalia and Bereiter (2003) define knowledge-building as "the production and continual improvement of ideas of value to a community, through means that increase the likelihood that what the community accomplishes will be greater than the sum of individual contributions and part of broader cultural efforts" (p. 1370). Maton (2013) sees knowledge-building as a process that generates "ideas that have utility or appeal beyond the specificities of their originating contexts" (p. 8). In a classroom, knowledge-building relates to students' building on their previous learning experiences and understandings, and reusing what they learned in new contexts (Maton, 2013). Both definitions emphasize knowledge-building as the result of cultural efforts. However, while Scardamalia and Bereiter highlight the significance of knowledge, they do not offer tools to analyze its organizing principles. LCT offers a means of conceptualizing knowledge-building that allows us to see knowledge as an object of study, so that one can then theorize its effects on social practices. LCT concepts may support researchers in exploring issues related to curriculum structures, analysing whether units of a course are knowledge bounded, or whether they extend and integrate knowledge from previous units; or one may examine processes of learning, investigating whether students are able to apply their understandings in new contexts and over time, and so on (Maton, 2009). An increasing number of scholars have been exploring such issues, in a variety of fields, applying LCT in empirical investigations of the organizing principles that underlie practices in, for example, history (Shay, 2011), academic literacies (Hood, 2012), design (Carvalho et al., 2009; Dong et al., 2014) and in many other fields. Our focus here is on exploring the structuring of knowledge in a case study of a learning network that is part of a course in design computing in higher education. In particular, we examine the "features" of knowledge within this environment, and we trace a trajectory for how knowledge is shaped over the sequence of tasks proposed. Understanding such issues is a necessary step, as we consider the potential effects of epistemic design on the activities of network participants. The next section introduces some key LCT concepts that we use in our analysis.

Legitimation Code Theory: Semantics and "semantic waves"

Legitimation Code Theory (LCT) is a sociological framework for researching and analysing knowledge practices. It is a practical approach, designed as an open-ended evolving framework (see Maton, 2014; Maton et al., 2014). Maton (2014) refers to LCT as comprising of a multi-dimensional conceptual toolkit that brings together a set of concepts for analysing organizing principles underlying practices, and referred to as *legitimation codes*. In this paper we focus on one of LCT's five dimensions – 'Semantics' – which regards social fields of practice as "*semantic structures* and whose organizing principles are conceptualized as *semantic codes* comprising *semantic gravity* and *semantic density"* (Maton, 2014). Semantic gravity examines knowledge in relation to degrees of context dependency, within a continuum of strengths. Stronger semantic gravity (SG+) denotes that meaning is more likely to be dependent on its context in order to make sense to people; whereas weaker semantic gravity (SG-) means that meaning is less context-dependent in order to make sense (Maton 2009, 2013). In the field of biology, for example, the meaning associated with the name of a specific *plant,* expresses stronger semantic gravity than the meaning of the name of a *genus of plants,* which in turn has stronger semantic gravity than *processes that are common to plants*, such as photosynthesis (Maton, 2013). Importantly, Semantics always views meanings in relation to "a context". The concept of semantic density articulates degrees of condensation of meaning, which may manifest as symbols, terms, concepts and phrases, for example (Maton, 2013). Again, within a continuum of strengths, stronger semantic density (SD+) is identified when more meanings are condensed; weaker semantic density (SD-), denotes that less meanings are condensed. When applying these conceptual ideas in a study conducted in secondary schools, Maton identified that *semantic waves* were key to understanding knowledge-building. He noted that the particular ways in which recurrent shifts in context-dependence and condensation of meaning occurred were of importance to knowledge-building (Maton, 2014). One key characteristic in knowledge-building is that it involves understanding how different forms of knowledge are related and change over time. When analyzing activities in classrooms, Maton observed that *teaching activities* often involved "a repeated pattern of exemplifying and 'unpacking' educational knowledge into context-dependent and simplified meanings" (Maton, 2013, p.9). This contrasted to educational *assessments*, where students needed to demonstrate that they had mastered certain pedagogic subjects, with the use of "relatively decontextualized and condensed knowledge" (p.9). Students needed to master not only how different forms of knowledge relate to each other but also how knowledge changes over time. The question then arises of what happens "in between" - how students get from "a" to "b", or how the "transformation of knowledge" takes place. Maton argues, that when teachers "unpack" technical or scientific terms, students are able to connect those to their everyday language and experiences, but it is crucial that the process does not stop there - concepts need "repacking". This is so, because academic subject areas require that students are able to see technical terms as part of a "web of meanings", which may involve, for example, compositional structures,

ISBN 978-1-86220-304-4

taxonomic structures and processes (Macnaught, et al., 2013). Assessment tasks often require that students demonstrate that they understand these technical terms within this web of meanings. When Semantics concepts were applied in the analysis of classroom knowledge-building activities, trajectories of knowledge practices related to *context-dependency* and *condensation of meaning* over time were identified (Maton, 2013) and some semantic profiles (mapped as "semantic waves"), seemed to be more likely to enable knowledge-building. Of particular interest were semantic profiles that show a specific pattern depicting growth over time, and starting from concrete going towards greater levels of generalization and abstraction. In the next section we discuss how the LCT concepts above were applied in the analysis of the epistemic design of Peep, presenting examples of elements that appear to support students in gradually moving from concrete to greater levels of generalization and abstraction, in the design of the sequencing of tasks proposed and elements that act as connectors within a "web of meanings". That is, we discuss design elements that seem to encourage movements from stronger semantic gravity and weaker semantic density (SG+, SD–) towards weaker semantic gravity and stronger semantic density (SG–, SD+). We show how a key design feature - the code editor, embedded throughout the environment - enables a certain pacing, enacting a specific way of dealing with knowledge within the network. We argue that such a combination of elements reflects a "semantic profile" that would favour opportunities for knowledge-building processes in networks.

Peep case study: supporting students to learn programming as a tool with which to design

Peep (Figure 1) is an online platform created to support one of the core courses in the Bachelor of Design Computing at the Faculty of Architecture, Design and Planning in the University of Sydney (Australia). Peep was designed with social network elements in mind, so as to promote knowledge sharing and knowledge-building amongst first-year undergraduate design students. Peep supports students' discussions about code and enables students to see, work with, comment on, and learn from other students' design work: code, images and animations. Processing.js is the visually-oriented programming language that students use. The course combines the creativity of design with technical knowledge of computing, where students have opportunities to explore aesthetic elements through computer-expressed works. Over a 13-week semester, students, lecturer and tutors meet weekly for one-hour lectures and two-hour lab tutorials. Students interact with Peep during tutorials, but as this is a web-based environment they can also access and use it remotely. The tutorials happen in the physical space of the labs but interactions also happen in the 'digital spaces for conversations' via the functionalities in Peep. Figure 1 (left) illustrates the main page of Peep, showing links to the pages that students can access during the course, recent activities and latest announcements. Figure 1 (right) is a snapshot of the Forum.

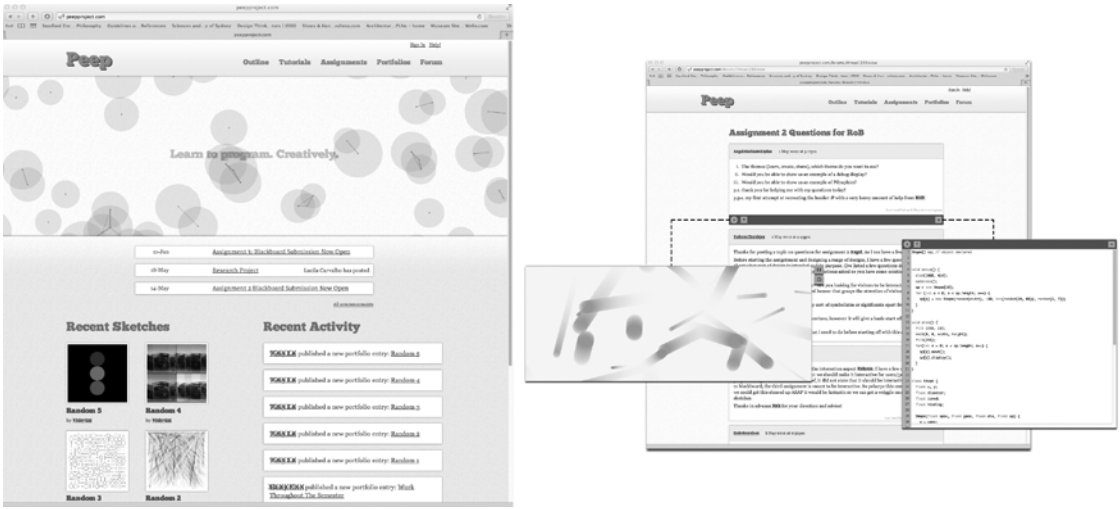

Figure 1: Main page (left) and the code editor showing code and visual image (right)

Data collection

Data collection and analysis involved interviews with the lecturer/designer of Peep, a developer, and one of the students enrolled in the course, and we observed students attending tutorials. We analysed the learning environment as an object of study itself, examining different elements of the network (set, epistemic and social design) in relation to the activities of students, collecting screenshots that illustrated moments we found

ISBN 978-1-86220-304-4

particularly interesting for our analysis (Carvalho et al., in press). At the time of our data collection, 63 students were enrolled in the course, and all of them used Peep during tutorials and for their assignments. In this paper we focus on epistemic activities and structures in relation to two spaces: code editors and tutorials.

What is the "context" in which "programming as a tool to design" is taught?

Semantic gravity conceptualizes how meanings relate to their contexts. At the macro level, the broader context of Peep is the field of design (design computing being the specialized area). In an earlier analysis of Peep, we discussed design elements that aimed to introduce first-year design students to the social context of the profession , exposing students to ways of practicing design that are grounded in practices within the field, encouraging support for, and critique of, each other's work (Carvalho et al., in press). Promoting "a sense of community" was highlighted by the lecturer/designer of Peep as one of its core functions. Although most tasks in Peep involved individual assignments, students engaged in conversations about each other's work. In the next section we explore the trajectories of knowledge that are enabled when students interact within Peep's varied spaces (Figure 1), using the concepts of *semantic gravity* and *density* to analyse its epistemic design. In each of these different spaces students are exposed to knowledge, for example: Tutorials space - where students have access to guidance for the completion of tasks; Forum space - where students discuss issues, ask for help, support each other; Portfolios space - where they can complete their assignments and also showcase their work.

Analysis and discussion

Code editor – a key epistemic element

One of the unique features in the design of Peep – the code editor – is described by the lecturer/designer as an element that brings programming to the forefront of the environment, highlighting code as a "first class object" - as something that is as important as text or images. Figure 1 (right) shows a threaded discussion from one of the pages in the forum. By clicking on one of the buttons (Figure 1, right) students open a window containing the images (or animation) produced with the coding. Clicking the button shown on the far right of the figure opens a window showing the code itself. The code editor, as a design element, allows students to navigate between different forms of representation (images/animation and code text). The code editor acts as a "translator device" supporting students in their learning processes, by visually offering a way of "unpacking" and "packing" different meanings associated with the use of programing to design. Such meanings would include, for example:

- What/how textual elements/characters together form a sequence of code
- The rules and norms, the order to be followed so that a sequence of code is formed - e.g. what text/characters needs to be typed and in what specific sequence
- Relationships of those rules to form a class of codes - e.g. "code a" belongs to the "class of codes x"
- Visualizations that show the effects of the code - e.g. a sequence of "code a" produces "image b"
- The rules and norms that link "code a" to "image b" - e.g. what code should be used for what effect
- Relationships of those rules to form a class of images - e.g. "image a" belongs to the "class of images x"
- Establishing connections between these relationships – e.g. between code-image and image-code

The code editor plays a key role in supporting first-year undergraduate students in understanding highly decontextualized and condensed ideas (SG–, SD+) such as those related to using code as a tool to design, or the use of textual elements and characters that will realize a certain image and produce a design. By enabling first-year students to navigate between two forms of abstract representations (code and images) it supports the recognition of relationships between those two, in the different associated meanings as listed above. The effect of the code editor in the networked context is that it brings the object of discussions to the forefront, for all to see, facilitating and supporting students' discussions of abstract concepts. When novice design students encounter a problem with their code, it may be difficult for them to verbalize what the problem is, due to the high degree of abstraction that programming requires. These novice students are thus dealing with concepts characterised by weaker semantic gravity and stronger semantic density (SG–, SD+) - they are context-independent and condense a range of meanings. With the support of the code editor, however, students do not need to say what sort of problem they "think" they have. Nor do they need to type the piece of code that they suspect is wrong into an email or threaded discussion. They simply have to make an entry in the discussion forum, with their code embedded in it, and ask for help. Unlike most text-based discussions in networked learning, where what is being discussed is not necessarily present or easily shareable, the code editor allows for the shared visualization of what is under discussion. Easy debugging of individual problems, however, is not the only benefit gained from using the code editor. It also facilitates learning by allowing others full access to the code. Other students see the types of errors that were made, and the proposed solutions. The discussion is easily

ISBN 978-1-86220-304-4

accessible and everyone can participate. Therefore, the code editor helps students converse about programming, supporting them in coping with complexities associated with learning a type of knowledge whose structure suggests meanings that display weaker semantic gravity and stronger semantic density. This is particularly important for novice programmers, as pointed out by the lecturer:

> and what's more important to me, actually, is that other students get involved then as well, because they don't need the patience to, you know, I have as an instructor to go through and work with the student to get that code and get their output, etc. But rather the code is right there, another student comes along and clicks on it and goes "oh yeah, I know what's wrong with that, I had that problem", and they can respond. So we build a community much, much quicker because … we reduce all of the barriers to sharing code problems and running that code and just showing off to each other as well, because that's part of what the portfolios do. (Lecturer interview)

As the code editor makes their design creations visible, students have opportunities to also explore how their peers have achieved certain effects in their work, and they may choose (or not) to use other people's work as the basis for their own designs. If so doing, students could be building on each other's activities and learning from each other's work. Our conjecture is that the presence of the code editor throughout the environment, and its use as part of tutorials, the discussion forum and portfolios, enables a trajectory that moves expressed knowledge upwards, allowing the lecturer to work with students at a "higher conceptual level".

> well, I have the opportunity now with Peep to take the lectures to a higher conceptual level, focus more on design issues and less on teaching students how to cope, which is something that I was getting frustrated by in the earlier teaching, was that I'd have to essentially teach a lot of coding in the lectures and then reinforce that in the tutorials, because the tutorials were essentially follow-along exercises where there wasn't a large amount of interaction and there wasn't a good support through a nice discussion forum like this. (Lecturer interview)

Tutorials – analysing the sequencing and rhythm of tasks

Students do not have to switch from one screen to another as the code editor is embedded in the Tutorials (see Figure 2): on the same screen they have a space to read about the task, and another to practice that task. So students can focus on the lesson at hand, rather than having to cope with different screens. Moreover, a specific lesson structure is repeatedly used in the tutorials: each lesson has the left side reserved for some textual and visual information, and the right side of the screen displays a code editor for students' activities. Once initial tutorials are completed, students know what to expect in subsequent lessons. Similarly, a consistent format is also used in terms of pacing the tasks (most tutorials have 4 to 6 proposed lessons) and sequencing (tutorials gradually build on previous ones). Students use the code editor to tackle the programming tasks proposed, and the editor automatically follows as they scroll down the page (Figure 2). Having the editor for their activities on the same screen is important to the epistemic design as it allows students to remain focused:

> Again, they don't have to context switch, there's no copying the code from one – from the discussion forum, putting it in to the Processing, and … reformatting it so that it will actually run and then – and then it doesn't run and then they have them switch problems. (Lecturer interview)

In their first tutorial, students undertake an "Introduction to Peep", where they are exposed to content related to very basic descriptive information - e.g. what is the environment, why they should use it, how to log in, etc. Such information reflects context-dependent practices with simpler meanings (they mainly refer to using Peep) (SG+, SD–). Students then learn about what characters and letters should be placed together to form a line of code, and rules and norms related to coding and its effects. These early tutorials also get students to work back from an image to figuring out the code that produced that particular image, and in so doing, they reinforce relationships between using code and image in their designs. In later tutorials, students work with moving images, which add complexity. They complete three assignments during the course. Each of the assignment questions is modeled on fictitious briefs, requiring students to demonstrate they have mastered programming skills for design. As the passage below suggests, students need to demonstrate their knowledge of programming to design in the form of "generative art and design" and expressing variations in "unique prints" under the themes of "movement and energy" or "balance and harmony", that is, in a context of weaker semantic gravity (they need to interpret the meaning and figure out how to express balance and harmony in generative art and design) (SG–):

ISBN 978-1-86220-304-4

> Design Brief: A printing firm is interested in experimenting with generative art and design and has approached you to develop software that will generate variations on a theme, such that every customer has a unique print. The designs will be printed on either apparel (e.g. t-shirts, shows, etc), posters or postcards. The client has proposed the following themes: (i) Movement and Energy; (ii) Balance and Harmony. Your job as a designer is to select one of these themes and one or more of the product categories and develop a sketch in Processing that will generate suitable design variations that express that theme. (…). (Peep webpage)

The trajectory of knowledge throughout the sequences of tutorials shows a pattern of: new programming knowledge is introduced, unpacked and repacked forming the basis for new tutorials. Thus, earlier tutorials build from simpler, more concrete understandings of code (SG+), where less meaning is condensed (SD–), taking students gradually into more complex meanings (SD+) within contexts that are more diffused (SG–) as the design brief above. Two excerpts from Tutorial 2 (depicted in the image on the left in Figure 2 and reproduced below for clarity) illustrate the unpacking of the elements of coding through an analogy with English language (SG+, SD–):

> An expression is like a phrase in English: an expression always has a value, determined by evaluating its contents. Some expressions can achieve complex results, but an expression can be as simple as a single number. Here are some examples of some expressions, with their values. [Image shown] (Excerpt 1, Tutorial 2)
> A set of expressions creates a statement, the programming equivalent of a sentence. Every statement ends with a terminator, the programming equivalent of a full stop / period. In the Processing language, the statement terminator is a semicolon [image surrounded by grey square]. You have to use a semicolon at the end of every statement that you make in Processing. [Image shown] (Excerpt 2, Tutorial 2)

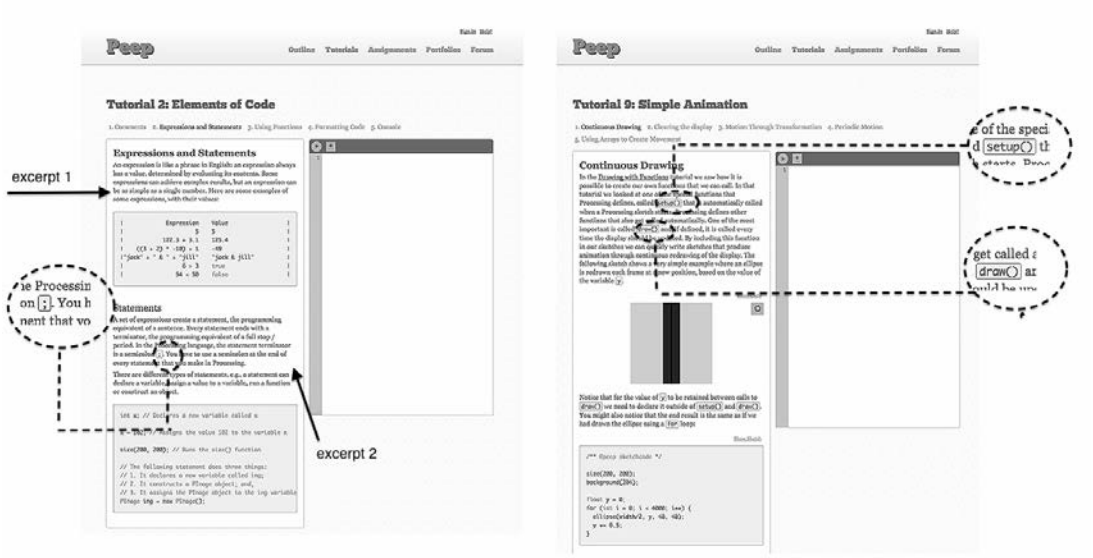

Figure 2: Tutorial 2 (left) and Tutorial 9 (right)

Excerpts 1 and 2 are accompanied by images that show the code in use, through a grey squared frame (see Figure 2, Tutorial 2). These larger grey frames within the tutorials act as markers for what eventually will be the space where a code editor will show the programming language and its visual effects (see Figure 2, Tutorial 9). Smaller grey squared frames are also used within the text, surrounding specific words or letters, and in this case they are markers of relevant technical terms in programming for that particular tutorial (dashed circles in Figure 2). They appear repeatedly throughout the tutorials, highlighting terms that are of importance in the text. Such elements are not just words in the screen they are key design elements in which more meaning is condensed (SD+) and they highlight to students relevant terms in the "web of meanings" within this context.

Conclusions

Design for networked learning can benefit from identifying and abstracting key elements that influence learners' activities. An LCT-informed approach may help designers explicitly articulate what is enabled within the

ISBN 978-1-86220-304-4

epistemic design of a learning network. In order to create effective structures of knowledge in a network, designers may analyse what and how tasks are proposed, their sequencing, pacing, and/or how other elements enable certain ways of communicating knowledge. LCT supported our analysis by mapping knowledge in a context, through the sequencing of the tutorials and in understanding the role of the code editor. We could then see what was achieved with the code editor as a design solution: allowing students to navigate between different forms of representation and group discussions, to accommodate higher abstractions than would otherwise be easy for programming novices. Such a design solution may benefit other learning networks where novices are required to work with concepts in similar ways. Our analysis suggests that a design solution that allows the object of discussion to come to the forefront of the environment helps students learn through discussions about their understandings, drawing on each others' work, and thereby engaging in processes of knowledge-building.

References

Bereiter, C., & Scardamalia, M. (2003). Learning to work creatively with knowledge. In E. De Corte, L. Verschaffel, N. Entwistle & J. van Merrienboer (Eds.), Powerful learning environments: Unravelling basic components and dimensions (pp. 55-68). Oxford UK: Pergamon.

Bernstein, B. (2000). Pedagogy, symbolic control, and identity: theory, research, critique. (Revised ed.). Oxford: Rowan & Littlefield.

Carvalho, L. & Goodyear, P. (Eds.) (in press). The architecture of productive learning networks. New York: Routledge.

Carvalho, L., Goodyear, P., Wardak, D. & Saunders, R. (in press). Peep: Peer support for programming. In Carvalho, L. & Goodyear, P. (Eds.) The architecture of productive learning networks. New York: Routledge.

Dong, A., Maton, K. & Carvalho, L. (2014). The structuring of design knowledge. In P. Rodgers & J. Yee (Eds.). Routledge Companion to Design Research. London, UK: Routledge

Goodyear, P. (2005) Educational design and networked learning: Patterns, pattern languages and design practice. Australasian Journal of Educational Technology, 21, 82-101.

Goodyear, P. & Carvalho, L. (2013). The analysis of complex learning environments. In H. Beetham & R. Sharpe (Eds.), Rethinking pedagogy for the digital age (2nd ed.). New York, NY: Routledge.

Goodyear, P., Carvalho, L. & Dohn, N. (2014). Design for networked learning: framing relations between participants' activities and the physical setting. Proceedings of the 9th International Conference on Networked Learning 2014, Edited by: Bayne S, Jones C, de Laat M, Ryberg T & Sinclair C.

Hood, S. (2012). Voice and stance as appraisal: persuading and positioning in research writing across intellectual fields. In K. Hyland & C. Sancho Guinda (Eds.). Stance and voice in written academic genres, Basingstoke: Palgrave Macmillan.

Macnaught, L., Maton, K. Martin, J. R. & Martruglio, E. (2013). Jointly constructing semantic waves: Implications for teacher training, Linguistics and Education, 24(1), 50-63.

Maton, K. (2009). Cumulative and segmented learning: exploring the role of curriculum structures in knowledge-building, British Journal of Sociology of Education, 30(1), 43-57.

Maton, K. (2014). Knowledge and knowers: towards a realist sociology of education, London, UK: Routledge.

Maton, K. (2013). Making semantic waves: A key to cumulative knowledge-building, Linguistics and Education, 24(1), 8-22.

Maton, K., Hood, S. & Shay, S. (Eds.) (in press). Knowledge-building: educational studies in Legitimation Code Theory. London, UK: Routledge.

Maton K. & Moore, R. (Eds.) (2010). Social realism, knowledge and the sociology of education: coalitions of the mind. London, UK: Continuum.

Pinto, A. (2014). Design and functioning of a productive learning network. Proceedings of the 9th International Conference on Networked Learning 2014, Edited by: Bayne S, Jones C, de Laat M, Ryberg T & Sinclair C.

Scardamalia, M. (2003). Knowledge building. In J. W. Guthrie (Ed.), Encyclopedia of education (2nd ed., pp. 1370-1373). New York, NY: Macmillan Reference.

Shay, S. (2011) Curriculum formation: A case study from History, Studies in Higher Education, 36(3): 315-329.

Yeoman, P & Carvalho, L. (2014). Material entanglement in a primary school learning network. Proceedings of the 9th International Conference on Networked Learning 2014, Edited by: Bayne S, Jones C, de Laat M, Ryberg T & Sinclair C.

Acknowledgements

Peter Goodyear and Lucila Carvalho are pleased to acknowledge the financial support of the Australian Research Council (Laureate Fellowship Grant FL100100203), as well as discussions with Rob Saunders and generous feedback from Nina Bonderup Dohn and Karl Maton on earlier versions of this paper.