CONCEPTUAL UNDERSTANDINGS OF NOVICE PROGRAMMERS

Roland Gesthuizen
Department of Education and Early Childhood Development, Victoria

Paul D. Chandler
Australian Catholic University, Melbourne

Abstract

The need for computer users to have a conceptual, compared with surface-level, understanding of computers has been argued by various authors over many years. Conceptual difficulties are not, of course, specific to the computer programming domain, and indeed are often the focus of Science education practices. This investigation concerns the understanding of secondary school students who are novice users of the Python programming language. A series of different tasks were developed to probe their understandings of relevant programming concepts. As Science education in recent times has often favoured probes of understanding and student-centred representational approaches, we argue for creative teaching and learning strategies which make visible and explicit their understandings, making them open to clarification and elaboration. In short, we contend that there is opportunity for creative pedagogy by bringing some Science education practices into 'Computer Science', thus helping students resolve misconceptions and identifying pedagogical approaches which may have unwittingly reinforced such views.

Introduction

Conversations online

An interest in constructivism since the early 1980s has galvanised an interest in learner's understandings in a range of subject areas. This has led to a paradigm change in the teaching and learning of various various school subjects, and the change in Science education practices is particularly notable (So, 2002; Tobin, 1993, p. ix). In terms of a theory of mental and conceptual models (Cardinale, 1991), there is a 'target system', and the 'mental model' is what the user presently has in his/her head about the target system, and a 'conceptual model' is one which is invented to provide a teachable representation of a target system. The interest, therefore, has been with shifts in mental models and leveraging such shifts through student-constructed conceptual models.

In contrast, the interest of Information and Communications Technology (ICT) educators with learner understandings of the technology has been different. Yan and Fischer (2004) have observed that insufficient attention has been given to how people learn to use computers from the perspective of cognitive development. Hammond and Rogers (2007) have also observed the relative lack of research into children's understanding of computers and computing concepts – particularly when compared with the very large literature on teaching and learning with ICT. Ben-Ari (1999, 2001, 2002) has been critical of the widespread application of minimalism, a methodology for designing manuals for software documentation and for using these manuals in training users of the software. Trained in the more behaviorist style of minimalism, he argues, when faced with an unfamiliar situation, the user will not attempt to employ or expand conceptual knowledge, but rather will attempt to find and recycle a task that was 'actively learned'. In short, Ben-Ari expresses concern with an insufficient attention to conceptual understanding.

Nevertheless, practitioners have been steadily implementing more learner-centred or constructivist-compatible teaching and learning approaches. For instance, Chesñevar, Maguitman, Gonzáles and Cobo (2004) have used such ideas to develop innovative approaches to the teaching of highly abstract ideas of theoretical computing and Chen (2003) has done similar with the teaching of computer networking. Whereas the emphasis has tended to be on the conceptions that students construct whilst in the computing classroom, not the conceptions that they bring to the classroom door (Powers and Powers, 2000, p. 1), there are a small number of insightful studies and it is work which take a 'student first' approach. Ben-Ari (1999) considered the mental models of word processing of academic staff in a university. Hammond and Rogers (2007) considered children's perspectives on issues such as 'What is logging on?' and 'How does a mouse work?'. Young student's perspectives in explaining the 'behaviour' of a mechanical, autonomous robot were studied by van Duuren and Scaife (1996) and Levy and Mioduser (2008). Kafai (2008) explored students' conceptions of a computer virus. Papastergiou (2005) and Diethelm and colleagues (Diethelm & Zumbrägel, 2012; Mesaroş & Diethelm, 2012) investigated high-school students' conceptions of the internet. The ImpaCT2 and related studies (ImpaCT2, 2002; Mavers, Somekh and Restorick, 2002; Pearson and Somekh, 2003) considered several thousand students' understanding of 'What is a computer?'. Across all of these studies, there is little consistency of findings. ImpaCT2 researchers concluded that students had detailed and complex cognitive representations of technologies, whereas Papastergiou (2005) found widespread simplistic and utilitarian mental models.

The authors of this paper are both Science teachers as well as ICT teachers, invested in the conceptual change model of teaching, and to some extent side with Ben-Ari (1999), troubled by the more minimalist approach which seems to often permeate ICT education, and seeking for a more conceptual basis for our work, and give greater respect to the thinking that students bring to the computer classroom. Our earlier forays into this territory (Chandler, 2010; Chandler and Gesthuizen, 2010) considered 'common place' computing activities. In this paper, our focus is on the more specialised work of teaching programming.

Focus for investigation

Roy Pea's (1986) work *Language-independent conceptual "bugs" in novice programming* is unquestionably the early and seminal work in the field. Google Scholar indicates that it has been widely quoted, but readily-located similar studies are not easy to find (e.g. Fleury, 2000; Pane, Ratanamahatana, & Myers, 2001; Spohrer & Soloway, 1986). Amongst them, the focus has not been at the upper secondary level nor in relation to more recent programming languages such as Python. Pea's investigation identified the following three misconceptions in the work of novice programmers:

- Parallelism - the assumption that different lines in a program can be active at the same time
- Intentionality - the attribution of forsightedness to the program
- Egocentrism - the assumption that there is more of the programmer's meaning for what he or she wants to accomplish than is actually present in the code

The focus of the investigation is, in the context of programming in Python, exploring the ideas about computing that students present to a teacher about what is happening inside a computing device. Therefore, to contribute to an extension or confirmation of Pea's work with respect to the specific contact of teaching of programming through Python to Australian upper secondary students.

Methodology

Participants

The participants were students from the second author's information technology classes in a secondary school in the city of Melbourne, Australia. Whilst there were three classes in total involved, they were small classes and the total number of participants was 29. Both genders were represented and participant ages spanned from 15 to 17. Students had a broadly different exposure to computing varying from no contact, some limited programming with Scratch to some more intensive programming experience with the Python[4] programming language through the GROK[5] Programming Challenge. All had some experience using computer applications such as word processing and spreadsheets. Each had their own

[4] http://python.org
[5] https://groklearning.com/challenge/

personal netbook computer.

Probes for understanding

Following the value of Science educators to use visual representations to help students represent abstractions (Tytler, Haslam, Prain and Hubber, 2009), and similar work in ICT education (Kafai, 2008), our earlier investigations (Chandler, 2010; Chandler & Gesthuizen, 2010) took a similar approach. We designed some simple questions to prompt students to draw representations of how variables 'work'. Only results for one question is considered in this paper:

Figure 1: Sample Question

> Here are two variables in a Python program:
>
> X = 16
> Y = "Cat"
>
> If you could "draw" what this looks like when this information is stored 'inside' a computer, what would your drawing look like? You can annotate your drawing to add some notes that describe the different parts of your drawing. Use your imagination and ideas about what could be happening inside a computer and how these variables might be stored.

Data collection

Students were presented with this questionnaire to answer independently on paper. Their teacher (the first author) provided considerable encouragement to record and submit an answer but not provide any clues or hints about what should be answered. No information technology or programming instruction was given to support or scaffold their answers. Submissions were codified to remove any identifying information then digitally scanned for further analysis.

Data analysis

Whilst Pea's (1986) research was available as an interpretive framework, we took an approach more aligned with grounded theory (Glaser and Strauss, 1967). Both researchers independently read over the responses. Initial classifications that we made were then discussed, and then we worked together to sort the responses into broad groupings. Our interpretations were then compared with Pea's work.

Results and Discussion

The responses were group into four broad classifications, which are explained below, which is organised as an approximate taxonomy ranging from least abstract to most abstract.

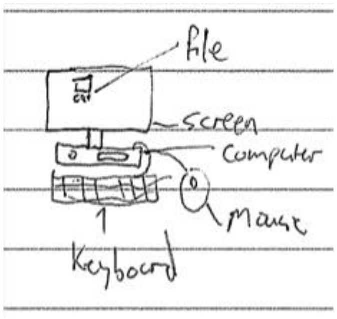
No representation

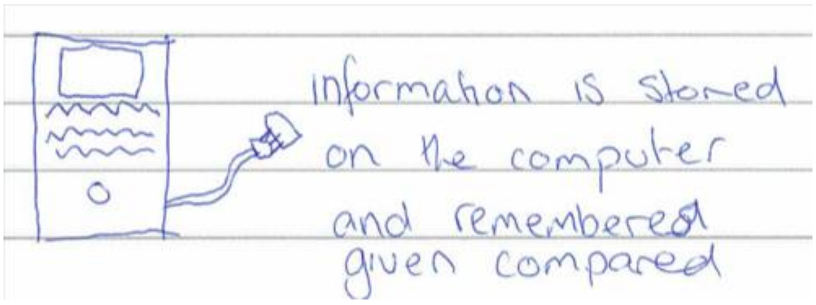
Two students responded with no answer. They indicated that that had absolutely no idea, even when pressed to use their imagination and despite considerable encouragement to submit an idea what may be happening.

Exterior view of the computer

Some responses indicated a recognition of the computer as a "system". The diagrams sometimes labelled the parts and indicated that information is entered and stored and perhaps manipulated inside the computer.

Figures 2 and 3: Student Responses indicating 'exterior view'

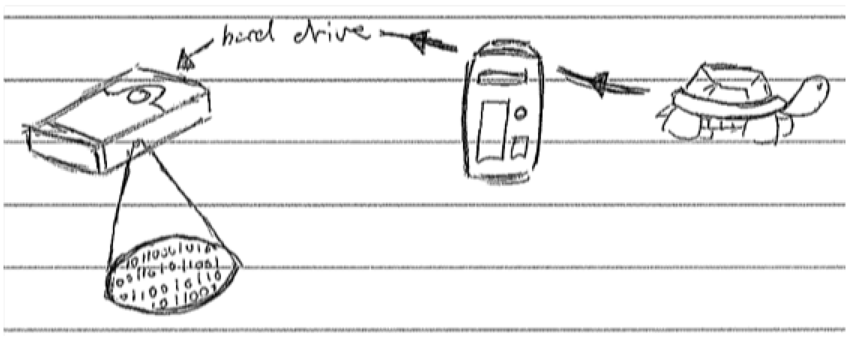



information is stored on the computer and remembered given compared

Naive symbolic understanding

Some responses suggest a notion of the “inside” of the computer as symbolic. One student describe a Turtle device connected to a computer full of chips that contains binary numbers. There is some sense of Pea’s notion of egocentrism (more meaning attributed than is actually specified) as it is otherwise unclear how the idea of the turtle arose.

Figure 4: Student Responses indicating 'symbolic understanding'



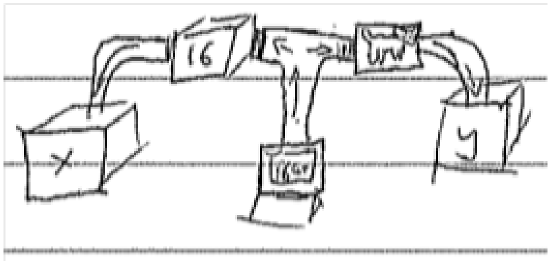
More detailed presentations of the “inside”

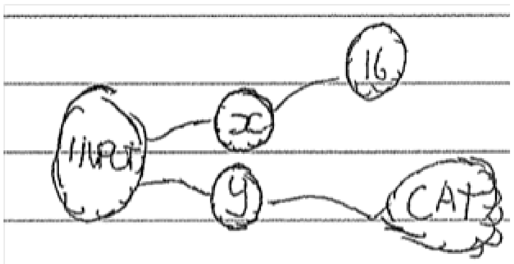
When examining the responses that explored what was probably happening at a ‘deeper’ level, there were four broadly different ways of visualising how information could be represented inside a computer. They varied from the fantastical notion of abstract animals, information dynamically flowing from boxes through different paths, static binary code and physical reality of dots of data or magnetic field lines. We will here consider each in turn.

Data ‘flow’

Some students submitted a model that illustrated the flow of information along pipes to different boxes or spaces.

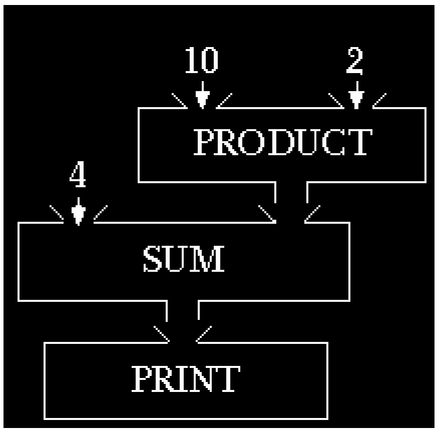
Figures 5 and 6: Student Responses indicating 'data flow'





This is reminiscent of the “plumbing diagrams” used by Brian Harvey (1997) to develop understandings of variables such as:

Figure 7: Sample 'plumbing diagram'



Notions of "flow" also connect with high-level mathematical thinking such as cellular automata (extensions, really, of Conways 'game of life'), which have been modelled to represent low-level digital structures such as logic gates (Schiff, 2005, pp. 97-100). It is important to note, though, that students had not been exposed to "plumbing diagrams" in their classes, so their use of this abstract representation is entirely of their own making. It is possible that these responses embody some degree of Pea's misconception of parallelism (elements of the program being active at the same time), but even so, it can be argued that this 'data flow idea' is the principle objective of the programming teacher: that values must be stored in a 'container' somewhere that they must be combined with other values in order for computation to take place.

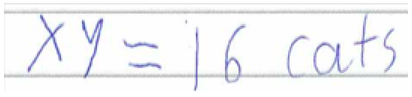
The response is therefore very important as it suggests that those students who constructed diagrams which successfully reinterpret information derived from their teacher and other sources and developed a representation which is highly productive, close to the canonical representation of the field and allied with high-level mathematical thinking. If this was the mental model (or interim mental model) of most students in a programming class, a teacher would have reason to be very happy.

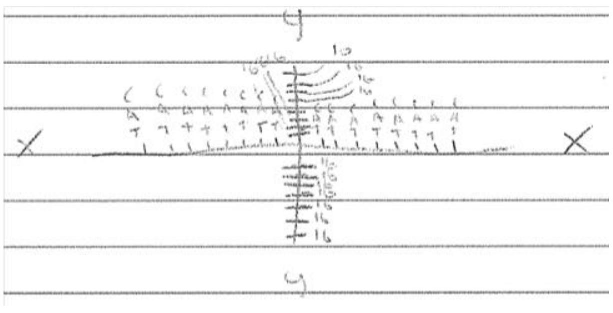
Mathematical

What can be seen from the response below is that students draw upon their prior experience strongly when they interpret the question. For instance, one student produced a response with cartesian coordinates has taken note of the symbols "X" and "Y" in the question and connects that with certain types of mathematical work. Likewise, other students who have produced something which looks like a mathematical equation; one of those students has possibly seen the same symbols and connected that with other elements of mathematical experience.

Some degree of Pea's notions of parallelism and intentionality is probably present in these responses.

Figures 8 and 9: Student Responses indicating 'mathematical understanding'



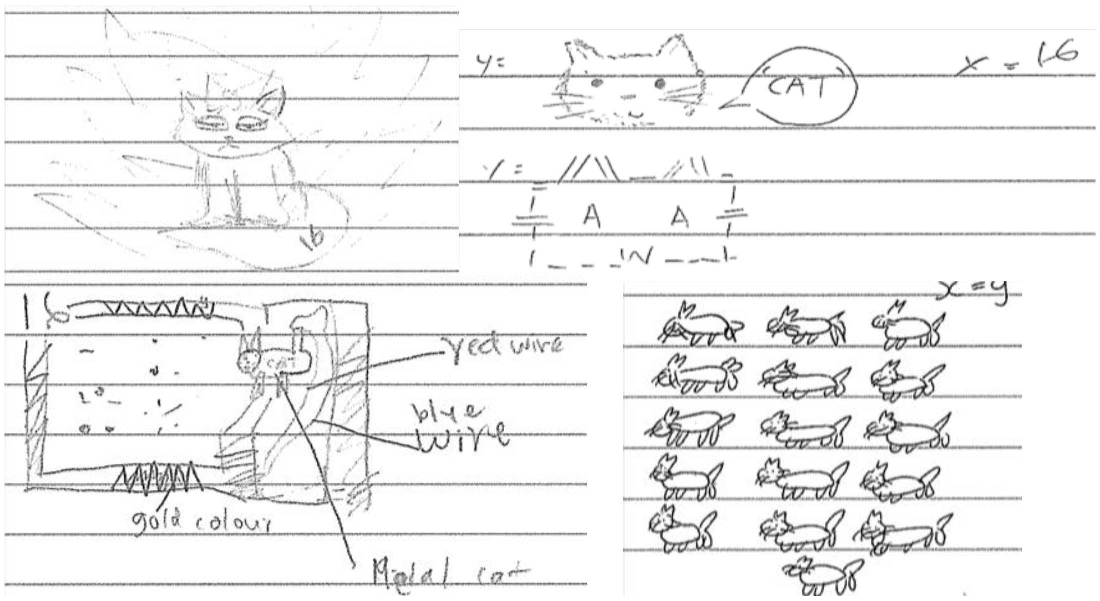


Fantasy

Imagination and fantasy elements are well represented amongst some contributions. The student who has seen Y='Cat' and has envisaged miniature felines certainly has some 'fantasy representation' but moreover has not seen the letters between the quotation marks as simply a sequence of characters but as a real-life object which needs to be modeled in some way.

The fantasy models represented ranged from a smiling young cat with a bow tie and another drawn using ASSCI characters. Another drew a small army of marching cats and two tried to physically draw a small cat, hard-wired into the computer circuitry. This support the conceptual understanding that the cat is both inside the computer and an integral part of the circuitry or perhaps the code.

Figures 10, 11, 12 and 13: Student Responses indicating 'fantasy understanding'



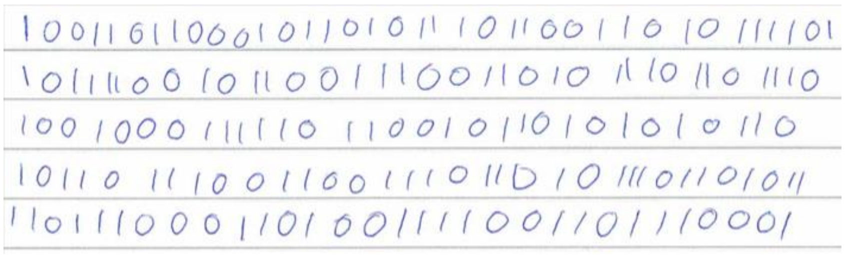
This anthropomorphic representation may draw on a wider culture such as the Internet, books or television shows such as "Nian cat" or "Cyberchase". "Nian cat" is a YouTube animation and internet meme including a mildly irritating music and flying computer cat. Videos and images of this computer cat are popular and shared between students. The Cyberchase animated cartoon series presents what is occurring within a computer as a kind of miniaturized version of the real world.

It would be drawing a long bow to suggest that that students would believe that a computer would contain an actual miniature model of a cat from the real world, but what is clear is that is that they have interpreted Y='Cat' to be actually indicative of a feline rather than a string of characters. Therefore, their parsing of the line of code is more based on 'common reading' of the sentence than a computer-science based one and, whilst 'cats in the computer' is probably not actually sensible to them, they do not really have any idea of a representation which makes any more sense than that. This reading of the program probably embeds some elements of Pea's notions of intentionality and egocentrism.

Unstructured code

Several students tried to represent the data as binary numbers in different ways. One tried to connect the data to each variable, another represented the huge iteration of zeros and ones. A third tried to illustrate that the stored data could be 'visualised' this way if you examined the could hold a magnifying glass to a computer chip. It is interesting to note that there was some confusion about how this information is grouped as discrete variables or a wall of data with no discrete boundaries.

Figure 14: Student Responses indicating 'unstructured code'

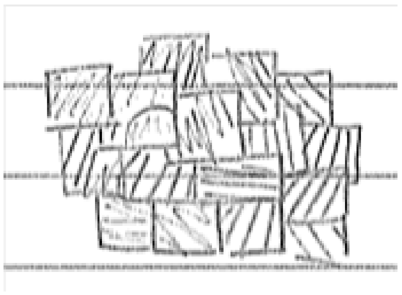
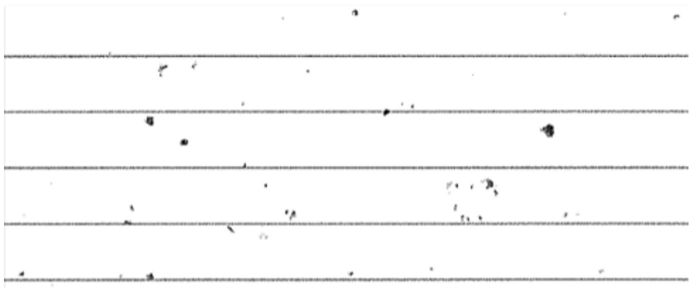


This representation is clearly influenced by prior learning that information in a computer is not stored as native words, decimal numbers or physical objects, rather it is directly converted, codified and stored as a binary number. There was perhaps some doubt with how the computer can tell different ‘boundaries’ to delineate between the different tokens represented in this binary sequence or where this code would be physically located.

A physical reality

Two students provided an interesting and conceptualisation of how data is stored inside a computer. After a considerable period of time struggling with a suitable answer, one settled on a series of deliberate dots on a surface to represent the information inside, perhaps on a chip. Another tried to represent this not as a series of dots but as a set of magnetic field lines along a surface.

Figures 15, 16: Student Responses indicating 'physical understanding'

These representations may have been influenced by some prior reading or learning about how information is stored on a DVD, compact disk or magnetic tape. The students may have attempted to map this visualisation into their model of how a computer works or looks inside. Whilst this is perhaps the most technically interesting answer at a physical level for information storage, it is worth noting that these students did not ground their representation in the more symbolic and abstract ideas.

In summary, there are not many visual representations which seem to be very viable in terms of explaining the lines of code presented. Whilst not wanting to stray into Pea’s misconception of intentionality (the attribution of forsightedness to a program) the majority of representations which are ‘static’ rather than part of a ‘process’. In contrast, to the computer scientist, a computer is a ‘busy place’, shunting data from one place to another at a fantastical speed. As represented in the order of the discussion above, the classification which we are on a surer footing, though, is abstraction compared with physicality, and it is from this basis that conclusions and recommendations proceed.

Conclusion

In the small amount of data collected, we have seen examples which fit in a range of positions that can be located upon a spectrum which extend from a 'macroscopic, physical' understanding on one extreme though to a 'microscopic, physical' understanding on the other, and with various forms of abstract thinking somewhere between. This is presented in Table 1.

Table 1: Spectrum of computer understandings

Macroscopic, physical	Abstract	Microscopic, physical
External hardware model with physical devices	For instance - naïve symbolic - mathematical - data flow - binary number - fantasy	Internal physical model with data storage elements

The computer programmer typically works in an abstract space, dealing with data, data structures and algorithms. At one end of the above spectrum is the physical infrastructure which makes all of this possible, such as the motherboard circuitry and the design of storage devices is broadly the domain of electronics engineering, but is rarely open to view because it is either hidden in a case or microscopic in size. At the other end of the spectrum is the physical reality of a computer which the user 'sees' and interacts with directly, the province of user-interface design. Data structures occupy the 'middle ground' between these two extremes, but unlike them is entirely an abstract conception. Where students are able to think abstractly, some elements of Pea's misconceptions can be inferred. Consistent with our earlier work (Chandler & Gesthuizen, 2010), the challenge seems to be in fostering student thinking at an abstract level at all. An extremely small number of students could be said to be thinking at an adequately abstract level where a deeper understanding of misconceptions could provide direction for productive future teaching and learning.

Given the small extent of this study, there is considerable scope to repeat this activity and compare the results with a larger cohort of students, different ages or amongst adults such as teachers or parents. A larger sample would allow a more careful exploration of the extent to which abstract (compared with physical) conceptions are indeed prevalent and to provide a more careful of the account of the range of mental models.

We came to this investigation as educators immersed in Science education, and that discipline has taught us that it is valuable for teachers to understand the conceptual understandings that students bring to the classroom, viewing these as mental models that are neither right nor wrong. Rather they should probably be viewed as alternative conceptual understandings that students have constructed from prior experiences. As Pea (1984) encouraged us to ask the question "How do inadequate mental models get transformed to better ones?", it is this perspective that we do not find widely represented in current ICT education, which seems to be more informed by minimalism.

What this brief study has suggested is that the first step towards that is to find ways to encourage thinking at an adequately abstract level. What confronts us is that there is, in a sense, a 'right' or 'wrong' to naïve understandings because an explanation based on physical realities (at either end of the spectrum) will always be inadequate. It is one thing to suggest that teachers should be aware of the various perceptions so that they can be better placed to diagnose and design activities that challenge this understanding and stimulate learning. But in order for that dictum to be meaningful, we must not only seek a more conceptual basis for our work but to *firstly* reveal the abstract 'space' that either implements, or is implemented by, the physical reality.

References

Ben-Ari, M. (1999). *Bricolage forever!* In proceedings of the Eleventh Workshop of the Psychology of Computer Programming Interest Group, Leeds, UK, pp. 53-57. http://www.ppig.org/papers/11th-benari.pdf

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in*

Mathematics and Science Teaching, 20(1), 45-73.

Ben-Ari, M. (2002). *From Theory to Experiment to Practice in CS Education*. Paper presented at the 2nd Annual Finnish/Baltic Sea Conference on Computer Science Education, Koli, Finland.

Ben-Ari, M., & Yeshno, T. (2006). Conceptual models of software artifacts. *Interacting with Computers, 18*(6), 1336-1350.

Cardinale, L.A. (1991) Conceptual models and user interaction with computers. *Computers in Human Behavior, 7*(3), 163-169.

Chandler, P. D. & Gesthuizen, R. J. (2010, April). *Learner's conceptions of 'common place' computing activities: a case in word processing*. Paper presented at the Australian Computers in Education Conference, Melbourne, Australia.

Chandler, P. D. (2010, Nov). *Rethinking computer science from a representational approach*. Paper presented Symposium on Contemporary Approaches to Research in Mathematics, Science, Health and Environmental Education, Deakin University, Melbourne, Australia. http://www.deakin.edu.au/arts-ed/efi/conferences/car-2010/papers/Author 1.pdf

Chen, C. (2003). A constructivist approach to teaching: implications in teaching computer networking. *Information Technology, Learning and Performance Journal, 21*(2), 17-27.

Chesñevar, C., Maguitman, A., Gonzáles, M., and Cobo, M. (2004). Teaching fundamentals of computing theory: a constructivist approach. *Journal of Computer Science and Technology, 4*(2), 91-97.

Diethelm, I., & Zumbrägel, S. (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 Calling '12 (pp. 67–73). New York, NY: ACM Press.

Fleury, A. E. (2000). *Programming in Java*. In Proceedings of the thirty-first SIGCSE technical symposium on Computer science education - SIGCSE '00 (pp. 197–201). New York, New York, USA: ACM Press.

Glaser, B., & Strauss A. (1967). *Discovery of Grounded Theory. Strategies for Qualitative Research*. Sociology Press.

Hammond, M., & Rogers, P. (2007). An investigation of children's conceptualisation of computers and how they work. *Education and Information Technologies, 12*(1), 3-15.

Harvey, B. (1997). *Computer Science Logo Style Volume 1: Symbolic Computing*. MIT Press. Available from http://www.cs.berkeley.edu/~bh/v1-toc2.html

Hubber, P., Tytler, R., & Haslam, F. (2010). Teaching and learning about force with a representational focus: pedagogy and teacher change. *Research in Science Education, 40*(1), 5-28.

ImpaCT2 (2002). *Pupils' and teachers' perceptions of ICT in the home, school and community*. BECTA, Coventry, UK. Available from http://www.bnecta.org.uk/research/impact2

Kafai, Y. B. (2008). Understanding virtual epidemics: children's folk conceptions of a computer virus. *Journal of Science and Educational Technology, 17*(6), 523-529.

Levy, S. T., & Mioduser, D. (2008). Does it "want" or "was it programmed to …"? Kindergarten children's explanations of an autonomous robot's adaptive functioning. *International Journal of Technology and Design Education, 18*(7), 337-359.

Mavers, D., Somekh, B., & Restorick, J. (2002). Interpreting the externalised images of pupils' conceptions of ICT: methods for the analysis of concept maps. *Computers and Education, 38*(1-3), 187-207.

Mesaroş, A.-M., & Diethelm, I. (2012). *Ways of planning lessons on the topic of networks and the internet*. In Proceedings of the 7th Workshop in Primary and Secondary Computing Education on - WiPSCE '12 (p. 70). New York, NY: ACM Press.

Pane, J. F., Ratanamahatana, C., & Myers, B. A. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies, 54*(2), 237–264.

Papastergiou, M. (2005). Students' mental models of the internet and their didactical exploitation in informatics education. *Education and Information Technologies, 10*(4), 341-360.

Pea, R. D. (1984). *Language-Independent Conceptual "Bugs" in Novice Programming*. (Tech Report No. 31). New York: Bank Street College of Education, New York, NY: Center for Children and Technology. Retrieved from http://eric.ed.gov/?id=ED319373

Pearson, M., & Somekh, B. (2003). Concept-mapping as a research tool: a study of primary children's representations of information and communications technology (ICT). *Education and Information Technologies, 8*(1), 5-22.

Powers, K. D., & Powers, D. T. (2000, Nov). *Constructivist Implications of Preconceptions in Computing*, Proceedings of ISECON '00 (Annual Conference for Information Systems Educators). Available from http://isedj.org/isecon/2000/408/ISECON.2000.Powers.txt

Schiff, J.L. (2005). *An introduction to Cellular Automata*. Available from http://psoup.math.wisc.edu/pub/Schiff_CAbook.pdf

So, W. W-M. (2002, June). Constructivist Teaching in Primary Science, *Asia-Pacific Forum on Science Learning and Teaching, 3*(1). Available from http://www.ied.edu.hk/apfslt/v3_issue1/sowm/index.htm

Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: are the folk wisdoms correct? *Communications of the ACM, 29*(7), 624–632.

Tobin, K. (1993). Referents for making sense of science teaching. *International Journal of Science Education, 15*(3), 241-254.

Tytler, R., Haslam, F., Prain, V., & Hubber, P. (2009). An explicit representational focus for teaching and learning about animals in the environment. *Teaching Science, 55*(4), 21-27.

van Duuren, M., & Scaife, M. (1996). "Because a robot's brain hasn't got a brain, it just controls itself" – children's attributions of brain related behaviour to intelligent artefacts. *European Journal of Psychology of Education, 11*(4), 365-376.

Yan, Z., & Fischer, K. W. (2004). How children and adults learn to use computers: a developmental approach. *New Directions for Child and Adolescent Development, 105*, 41-61.