Conceptual Understandings of Computers: Challenges to Research, Pedagogy and Curriculum

Paul D. Chandler
Yarra Valley Grammar

Introduction

Over the past 20 years or so, increasing attention has been paid to constructivist understandings of student learning. In the subject area of Science the concept of "children's science" has received considerable attention. Osborne & Freyberg (1985, p. 12) comment that:

- From a young age, and prior to any teaching and learning of formal science, children develop meanings for many words used in science teaching and views of the world which related to ideas taught in science;
- Children's ideas are usually strongly held, even if not well know to teachers, and are often significantly different to the views of scientists; and
- These ideas are sensible and coherent views from the children's point of view, and they often remain uninfluenced or can be influenced in unanticipated ways by science teaching.

This paper is an opinion piece, whose author is a teacher of both science and computing, and its central argument is a need to take seriously "naïve understandings" of information technology in the same fashion as "children's science" influenced areas of science education.

An Historical Reflection

It has always been the case that students would have entered a science classroom having had exposure to the world around them throughout their lives; children's science has taken seriously the idea that the understandings that they may have acquired about the way the world works should be taken seriously as a departure point for future teaching and learning in Science classes.

Twenty years ago, life experience with computers could not be assumed. More or less the only experience students had of a computer was that which they were given in the laboratory. Very few students had a computer at home. Well-meaning teachers (such as the author) would try to make connections between life experiences and teaching computer use. For instance, to introduce concepts such as 'desktop' 'copy' 'paste' 'cut' (hard for the novice user to imagine with pre-GUI word processing software such as Wordstar and Xardax), one could clear a desk space, placed a newspaper on it and created a toolbox of scissors and glue and proceeded to act out what the students were doing with the software.

Compare this with a science teacher who teaches some introductory electricity by using an 'interpretive discussion' rather than the presentation of facts. This teacher might bring a battery connected to a light bulb into class, and ask the class to observe. Perhaps after a few minutes, the light bulb is no longer lit, and the teacher asks "why". Students may explain that electric current flows from battery to bulb and is eventually used up, which is counter the scientific perspective of the conservation of energy. The teaching sequence couldn't stop there, but a simple demonstration, drawing on previous experience, has been the starting point of a discourse which is conceptual in nature, to the point that it makes the previous example the bringing in of paper, scissors and glue seem trite. Despite our best efforts, teaching computing, at least in the introductory levels, was about fundamentally about knowledge acquisition. The teaching of computer programming was a welcome relief, because so much of it rests on pattern recognition, so that the thinking of the students, rather than the knowledge imparted by the teacher, could be the focus of the learning activities.

Nearly twenty years' later, I teach in an environment where we assume that students have ready access to computer resources outside of school time. If we use terms such as 'copy' and 'paste', it can be fairly safely presumed that they can do this. Students of the present era are frequently described as "ICT savvy" (Christopherson, 2006) or "digital natives" (Christopherson, quoting Prensky). To spend a lesson acting out these word-processing tools would be gently tolerated by the students at best and, by-and-large, a completely unnecessary piece of teaching. We are no longer teaching ex nihilo.

By not re-teaching skills which students bring to the classroom, teachers take this reality seriously, at least at one level. However, this paper takes as its departure point the need to take seriously the underlying concepts about computing which students have developed and that they bring to the learning environment.

Epistemological and Ontological Understandings

Constructivitism is an epistemology which informs us of how knowledge might be said to be acquired and integrated with other knowledge. Though dating back at as early as the 16th century (von Glasersfeld, 1988), constructivism has received considerable attention in recent decades and is as central to the works of theorists such as Piaget and Vygotsky (Ridgway & Passey, 1991).

The central tenant of constructivism is that we come to know our world by interacting with it (Brookes, 1987, p. 64). In the constructivist paradigm, knowledge is not a representation of an observer-independent world. Rather, individual learners construct understandings of concepts, situations, people, etc. which are viable in the experiential world of the knower (von Glasersfeld, 1988, p. 1). Knowledge arises neither from the subject nor the object, but rather through their interaction. It can be said that there is no pre-defined body of knowledge, a state of affairs referred to as "subjective realities" (Brookes, 1984, p. 24). Stern (1992) expresses these core ideas by saying that,

> ... the self is made up and determined by inner structures which the self itself (so to speak) is able to design, build and alter … (p. 22)

and Knowles & Holt-Reynolds (1994) (quoted by Carter & Doyle, 1996, p. 122) have explained that,

> people construct ideas as they learn, and they use prior knowledge, experiences, and beliefs, as well as interpretations they generate in the moment, as the stuff out of which to build those ideas. (p. 6)

To adopt a constructivist epistemology is to assert that knowledge is robust (insofar as it proves to be viable), idiosyncratic, sensitive to the particular holder, incomplete, familiar and sufficiently pragmatic to have taken the learner to where he or she is today, and to acknowledge that knowledge is not constrained to the learning of propositions and rules at identifiable moments in time, but includes the rather more ad-hoc accumulation of experiences, beliefs and interpretations.

The concern which I share in this paper is that rather than accepting learners as being "IT savvy" at face value and without enquiry, there should be a more earnest effort to identify the knowledge that students have about computers and computing. That is, to presume that rather than being authoritative, that such knowledge may well be robust, idiosyncratic, familiar and pragmatic, but unique to the individual learner.

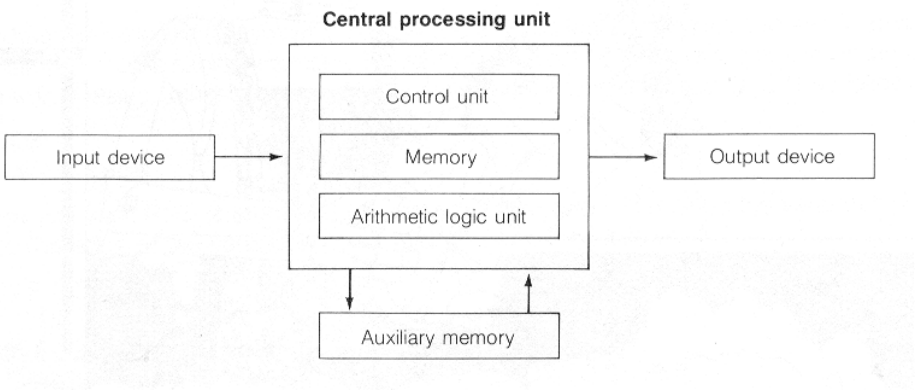
In the following section, I will share a range of anecdotes which highlight for me that learners do indeed have concepts about computing which do not always compare well with authoritative knowledge about the topic. Before doing so it is important to acknowledge an ontological difference between knowledge in computing compared with that in science. Theories in science – whether they be those of Newton, Einstein, the alchemists, or a student – are part of a quest towards ever more productive explanations for how the world works; there are no absolutes in terms of explanation. In contrast, computing - at least at the level of software or macroscopic hardware function – offers absolutes: what a task bar is, how saving a file should work, for instance, are precisely defined.

Anecdotal Observations

In my time as a teacher, there have been a number of examples where students seem to find computing difficult, and which and the origin of the difficulty seems to be with the underlying understandings of what might be called 'generalist computing concepts'.

I can think of a number of instances where **saving files** is a concern, and these are discussed in the following paragraphs. However, it is firstly important to acknowledge what 'saving' means from the point of view computer science. From this authoritative view point, when a file is created, it exists within the computer's working memory, which is volatile and thus will be lost when the computer is turned off. The act of saving a file transfers the file from the

working memory to one of a number of available secondary (or auxiliary) storage devices. Once upon a time, we used to use the students first experience of saving a document to explain something of this model of computer architecture (Tatnall & Davey, 1985, p. 16):

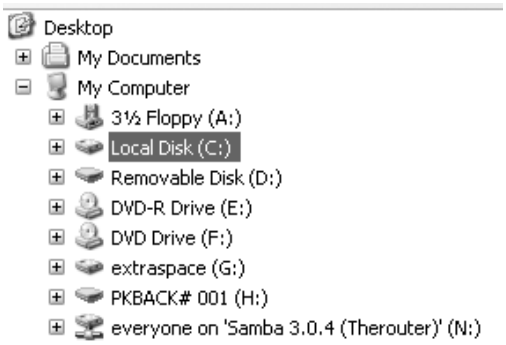


For a semester, I taught an adult learner of non-English speaking background for his first introduction to computers. Despite being shown the process of creating, saving and printing, each lesson he focused on placing his disk in the drive correctly, and then proceeded to create and print his work, only to come in the following lesson to re-type and re-print his work. After about four lessons of this, he realized what his disk was all about, and how wonderful it was to be able to save his work, and how this would improve his efficiency each lesson. Whilst at the very early stages of this student's computer use, this anecdote suggests the possibility that the accepted and definitive model of computer architecture is neither intuitively understood nor easily grasped.

Another anecdote concerns another adult learner had reason to save a document 'to the floppy disk'. This person had been using computers in her teaching, but was very conscious of a need to learn more. A telephone conversation to suggest that she save the file to the floppy disk so that she could work on it elsewhere did not faze her. When, in fact, the work was not to be found on the floppy disk, this was a cause for distress. To diagnose the problem, she was asked how she went about saving it, to which she replied that she had clicked on the 'save to disk' icon on the tool bar. It transpired that, thinking that a picture of a floppy disk meant save it to a floppy disk, she had clicked on  rather than 

The discussion that followed, however, revealed that this learner had no conception that floppy disks are often identified as 'drive A:' or that the other drive letters have any meaning useful to the novice. This simple anecdote not only reinforces the possibility that the accepted and definitive model of computer architecture is neither intuitively understood nor easily grasped, but neither association between drive letters and secondary storage devices as implemented in Microsoft Windows, nor is the 'iconic vocabulary' expected of computer users.

A similar issue can be identified from test responses from junior secondary students to an item on a multiple choice test. Students were shown the following diagram:



The question asked the students to identify the network drive and the local hard drive. This diagram was somewhat different to the network that they are used to (they are familiar with drive H: being the network drive), and they were not given the option of reporting drive C: as the local hard drive. The correct responses – G: for the local hard drive and N: for the network drive – should have been easily identifiable by attending to the icons, but a considerable diversity of responses was received. Clearly, there is something interesting about how students approach such a question, and potentially their understanding of **relationship between secondary storage devices and drive letters**.

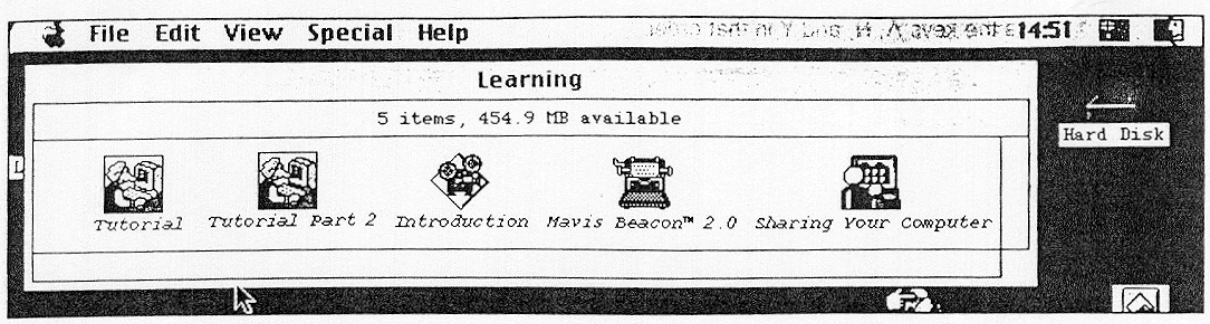
The **setting of tabs/margins** seems to be an area which many learners find difficult. The difficulties which they experience are not ones I relate to easily, because I used a typewriter, and MacWrite on the Macintosh was the first word processor I used. With a typewriter, one had to set the tabs and margins appropriate for each section of the document, and then make a physical change if necessary. Likewise with MacWrite, you inserted a ruler, and settings made on that ruler remained in force until another ruler was inserted. To bring a typewriter into class seems almost as archaic as bringing paper, scissors and glue stick for a demonstration of copy-and-paste. However, when the tab key is used or when tabs or margins are set, it seems important to enquire what understanding learners are basing their work on, as the physical model (the typewriter) is most probably not an accessible one.

Another potentially arcane practice was to bring in a pile of overhead transparencies and illustrate how you could place text on one slide and graphics on another, and they could overlap but coexist in the same document. Overhead projectors are becoming a thing of the past, but **layers** are not: they are crucial to understanding a range of concepts in many software packages, for instance frames in a word processor, master slides in PowerPoint, and layers in Paint Shop Pro, Dreamweaver and Flash. So, in the absence of a tangible analogue, one might also wonder what sense is being made of 'layering' in software, and the differences between how different software products implement this idea, and how experience with one product can be transferred to another.

Computer users often seem to be confused about **when to click and when to double-click**. As I sit by colleagues (not so much students) to assist them, many seem to double-click on hyperlinks on web pages, or on icons in the start

menu. In simple terms, the accepted explanation is that one clicks once to select, and double-clicks to execute. The idea of 'executing' a program presupposes an idea taking the program from secondary memory and placing it in the computer's working memory. So, from the computer architecture point of view, double-clicking on a hyperlink (which implies trying to execute something, rather than selecting it) quite an interesting understanding of what's going on. Computer systems themselves are not very consistent with the mouse action required (Windows varies from Macintosh which varies from X-Windows), so perhaps users are simply making the best of a bad situation. But the responses of junior secondary students to the following question give some food for thought.

The diagram below shows the screen of a computer



Which of the following is the most likely way to start the program "Mavis Beacon"?

(a) Type "RUN" on the keyboard
(b) Move the picture of the typewriter onto the picture of the hard disk
(c) Choose "File" from the menu bar at the top of the screen and click "Open"
(d) Point the mouse at the picture of the typewriter and click the mouse button once

This question was borrowed from a past paper of the 'Australian Computer Studies Competition', and to those in-the-know, represents a Macintosh environment, for which the answer is unambiguously (d) The competition, however, does not set questions specifically for one platform or the other, and in the spirit of this, I borrowed it to probe something of understanding of 'when to click' in students who did not use Macintoshes. The results (from over 100 students) show that some students would select each response. I probed this with a reference group of 8 students, which led to a surprising comment of one girl who chose (c). She explained that while she had never seen anything like that, she thought it would be worth investigating. There was a sense in which her answer displayed an encouraging spirit of experimentation, but that experimentation was important because her experience *had not* led her to develop any patterns that might be useful to apply to this situation. It was with this sentiment that the other students agreed. This highlights the need to more

fully explore the pre-existing understandings that student have when met with such situations and how that effects how they work with computers.

Other ideas which are sometimes presented by students which cause me some consternation are that **Bill Gates invented the internet** and that **anything that goes wrong with my computer is a virus**. In the latter case, the view is often so well entrenched that an attempt an explanation of the difference between viruses, adware and a hardware fault often lands on deaf ears. Likewise in the former case, one can practically see students' eyes glaze over when one begins to explain about the American military, Tim Berners-Lee and the days when altavista was a pre-eminent search engine. It is clear that student understandings (particularly misunderstandings) are a powerful influence on current learning.

This ends the recollection of anecdotes which suggest, to my mind at least, that the explanation of what is going on in the student's mind might be best explained through idiosyncratic and personally held 'generalist computing concepts'. I now turn to a review of the relevant literature.

Literature Review

As this is an opinion piece, it won't be claimed that an exhaustive literature search has been undertaken. However, an ERIC search has been conducted and several useful papers have been identified.

Powers and Powers (2000) have conducted some research in to student preconceptions related to 'Computer Science and Information Systems' (CSIS). They observe that

> little work has focused on identifying the initial ideas that students bring with them to the door of their first computing class. Some related work on student conceptions has been done, but it differs from ours in the following important ways. First, these works primarily consider the conceptions that students construct once in the CSIS classroom, not the conceptions that they bring with them to the door. Also, most of this work is limited in scope to programming per se, as opposed to CSIS generally (p. 1).

Later in their paper, Powers and Powers observe that

> considerable research has focused on the erroneous ideas that beginners develop in the process of learning to program. But to our knowledge, the idea of considering the intellectual framework that existed before the learner engaged the subject has not been explored. To what extent might their erroneous ideas be the result of general knowledge, formed in the general social setting, which is either inaccurate or misapplied? CSIS educators must confront the erroneous preconceptions that students bring to the discipline from general society, and these

> preconceptions cannot be confronted until they are identified (p. 4)

This is exactly the sentiment which I have been expressing throughout this paper and if Powers & Powers' consider that their work in CSIS is broader than the discussion of the intellectual frameworks of beginner programmers, then what I am suggesting in this paper is broader still.

Powers & Powers identify several preconceptions which effect student learning in CSIS, which are interesting though fairly narrowly focused on CSIS. However, their research method is not discussed. It is not clear whether their paper is primarily a reflection on what a teacher has casually observed after much time in the classroom, or whether a data gathering probe analogous to the 'interview about instances' common in children's science research (Osborne & Freyberg, 1985) has been employed.

Scott Brandt has written a number of papers discussing the mental models of learners as applied to the role of the reference librarian (Brandt, 1997; Brandt, 2001; Brandt & Uden, 2003), again drawing on a constructivist understanding of student learning. The most recent of these papers, "Insight into mental models of novice Internet searchers" is interesting for three reasons. Firstly, Brandt uses the 'mental model' terminology of cognitive psychology to distinguish between "conceptual models" (an authoritative, idealized understanding of how a system works) and "mental models" (a user-specific collection of knowledge which builds a foundation of understanding and provides the tools for problem solving in a given domain). In reference to the ontological difference between knowledge in computing and science discussed earlier, this terminology is helpful.

Brandt & Uden have also clearly presented their research methodology and approach to data collection – that of applied cognitive task analysis. Thirdly, the work is interesting because it is directed at applying an understanding of student's mental models to the differences between experts and novices.

Primarily, the literature review conducted this far has revealed that there is no explosion of academic activity in the field of naïve understandings of computing concepts of which the author is unaware, although there is consolidated research in some related fields such as computer programming. There is much scope for work in identification of research methodology and suitable and efficient data gathering techniques.

What might be looked for?

The intention of this paper is to argue for research into what I have loosely described as 'generalized computing concepts'; essentially to make a concerted effort to probe learners about why they do what they do – to strive to document the thinking that accompanies action and user explanation. It is thus highly

presumptuous to suggest which 'generalized computing concepts' might be, or even if this title proves ultimately to be the best description of it.

However, one needs to start somewhere, and the reader also possibly needs some suggestions to more readily understand the sort of thing that I am referring to. So, based on nothing more than anecdotal observation and intelligent guesswork, I am wondering whether there might be important 'general concepts' which are associated with:

- Physical understandings which connect the orientation of them mouse the connection between mouse and pointer, the difference between insertion point and pointer
- A 'windows' interface: what it means to have active window; background tasks etc
- What it means to click; click-drag, double-click, etc
- The filing system, drive letters, folders and the idea of saving and 'save as'; absolute and relative paths
- The idea of an object and the notion of 'selecting' text or objects
- The idea of layers
- The distinction between viruses, adware and hardware/software faults
- The internet, networks and connectivity
- How search engines work

All of these are fairly fine-grained. Consistent with the discussion earlier in the paper, broader concepts, such as that of computer architecture, maybe important. In addition, Powers & Powers (2000) some (mis)understandings which may have broader implications:

- Computers as analogue devices

  Students are accustomed to devices that respond linearly with variances in their input. For instance, the accelerator or brake in a car changes the speed of the car in response to the degree of pressure on the pedal. Based on this type of experience, a novice programming student would expect that a program that is "nearly" correct will produce output that is likewise "nearly" correct. The fact that changing a single line of code, indeed even just changing a single bit in certain cases, can drastically alter the output runs counter to their analogue experiences. Demonstrating the reality of such a possibility can helps students deal with debugging down the road.
- Complexity and levels of abstraction

  Teaching students to deal with the complexity of computing systems through top-down design and step-wise refinement are standard fare in introductory programming courses. But few students ever truly grasp the degree of complexity of computing systems and the absolute necessity for levels of abstraction throughout computing. Indeed, students often equate the complexity of computers with other much simpler digital devices, such as telephones and stereos. It has been asserted that such student conceptions of the complexity of computers are orders of magnitude worse than comparing ... the supersonic jet plane with a crawling baby. When confronted with the differential in

complexity, a computer architecture student remarked "I used to get [angry] when my computer crashed. But, the more I find out about what's going on inside, the more amazed I am that it doesn't just crash all the time."

- The computer as a giant brain

  Because computers are increasingly commonplace, most students have some concept of what a computer is. However, these concepts range from those that are very accurate to those more akin to a box of monkeys (as seen in cartoons) or a giant brain. The simplest of these preconceptions, like the giant brain, oftentimes seem to be a direct reflection of the student's perception of the capabilities of the machine. For instance, the sophistication of today's user interfaces inspires a myriad of preconceptions regarding the capabilities of the machine. Consider screen images which concurrently present information from multiple applications. The reality that this information is distinct and not necessarily shared within the computer is not apparent to students. The result may be a "giant brain" conceptualization of the machine, in which all parts are "aware" of all the information available. "Why do I need to write code in my program to determine the date? The computer already knows that!"

Having more clearly identified the sort of thing that research into conceptual understandings of computers might identify, I now move to discuss some possible motivation for this line of research.

The Motivation into Research into Conceptual Understandings of Computers

At one level, the interest in conceptual understandings of computer is purely theoretical, to raise the possibility that there is more than one way of looking at using a computer than amassing skills. **Can it be shown that certain broad-based conceptual understandings and widely held by learners, and, if so, what are they?** Following the identification of these, **what might then be suggested as teaching methods and curricula focused around the fostering of these conceptual frameworks** – compare the influence of teaching methods such as 'interpretive discussions' and 'predict-observe-explain' in science teaching.

As stated in the historical reflection (above) I have concerns that the curriculum and teaching of computing is based around the acquisition of skill and knowledge rather than the understanding of concepts. Certainly, the Information and Communications Technology Domain of the Victorian Essential Learning Standards (http://vels.vcaa.vic.edu.au/essential/interdisciplinary/ict/index.html) places a good deal of emphasis on the application of computing knowledge and skill, and it would be unfair to suggest that only lower-order thinking skills are being encouraged. However, the computer use is described in terms of skills/knowledge to be learnt rather than broad-based conceptual understandings that need to be fostered. Nor are the VELS unique in this perspective –

curricula such as the International Computer Driving License (http://www.ecdl.com/main/download/syl4feb06.pdf) are also described in terms of skills/knowledge.

Certainly, Fensham & Lui (1999) have identified some questions which as relevant to learning how to use a computer as they are in any other discipline area, and they direct one's thinking away from the teaching of skills to the teaching of broader conceptual foundations. Among their suggestions are:

- What wider or general sense or reality does this content exemplify and open up to the learner?
- What basic phenomenon or fundamental principle, what law, criterion, problem, method, technique, or attitude can be grasped by dealing with 'this' content as an 'example'?
- What situations and tasks are appropriate for helping students grasp the principle of the content by means of the example of an elementary 'case', and to apply and practice it so that it will be of real benefit to them?

It is not impossible to believe, for instance, that knowledge of frames in a word processor, might aid an understanding of master slides in PowerPoint, and layers in Paint Shop Pro or Dreamweaver. My experience of IT teaching and curriculum has not been to cast it in these terms, but to be concerned with the acquisition of application-specific knowledge. (I would not suggest that this particular example is, definitively, an important broad-based concept; that remains to be identified through relevant research.)

There are some relevant practical implications of such an approach. In circumstances where students are thought of a "IT savvy", and where it is reported that students are learning more of their ICT skills outside of school than in the classroom (Meredyth et al, 1999), **it may make more sense for the purpose of curriculum to be directed around developing a good conceptual framework rather than teach more and more skill. It may even be that certain conceptual understandings are best developed by engaging at activities away from the computer.**

Casual observation of students over many years suggests that some students who have had a fair degree of exposure to computers have picked up some 'enabling' skills, knowledge and concepts – skills, knowledge and concepts that in some way enable them to more quickly 'catch on' and adapt to new types of software. This is very much the expert/novice differences as considered by Brandt & Uden (2003). **It is, after all, very much the role of computer education to help students develop understandings that are increasingly productive and viable.**

Conclusion: Suggestions for the Future

It seems to me that the next thing to do would be for a small group of researchers to conduct some exploratory probes into student understanding of computing concepts. There are a small range of sources which could provide inspiration for suitable data gathering instruments. Brandt & Uden (2003) have used applied

cognitive task analysis. Then there is the range of techniques discussed in White & Gunstone's (1992) aptly titled *Probing Understanding*. Fensham & Lui (1999) have developed the technique of 'primed clinical interviews', and the author has tinkered with some interview schedules inspired by this technique for probing understanding of file management and use of internet search engines (see Appendix).

The major difficulty in the work which the author has completed to date is the need for enthusiasm, goal-setting, discourse and motivation would be present in an academic team. It is hoped that a group may be formed to further what has been argued as the important work of probing conceptual understanding of computing concepts.

Appendix: Understandings of Internet Search Engines

To probe understandings of search engines, students were asked to respond to the following questions. Inspired by the 'primed clinical interview' of Fensham & Lui (1999), there are two groups of questions. First of all, there are questions which orientate the student to their ideas about the topic. Secondly, students are asked a second group of questions which do not ask for their knowledge of the topic to be described directly, but in terms of analogies; none of the analogies are a precise representation of the subject, but each of them has a 'ring of truth' about them.

To what extent do you believe that the following are true statements about internet search engines:

- They search for data on every page of the Internet
- They locate key words by searching the world
- They are just like a library
- They search a certain part of the Internet for you
- A search engines searches the pages which are connected to it

Write a few sentences to describe why a search engine might be like:

- a library
- a library catalogue
- a librarian
- a game of Chinese whispers
- the index page to a book
- someone who is a speed reader

Can you thinking of any other analogies for search engines?

Which do you think is the best analogy? Why?

References

Brandt, D. S. (1997). Constructivism: Teaching for Understanding of the Internet *Communications of the ACM, 40*(10), 112-117.

Brandt, D. S. (2001). Reference, Mental Models and Teaching Technology. *Reference Librarian 74*, 37-47.

Brandt, D. S, & Uden, L. (2003) Insight into mental models of novice Internet searchers: Adaptation of Task Knowledge Structures. *Communications of the ACM, 46*(7), 133-136.

Brookes, M. (1984). A constructivist approach to staff development. *Educational Leadership, 42*(3), 23-27.

Brookes, M. (1987). Curriculum development from a constructivist perspective. *Educational Leadership, 44*(4), 63-67.

Carter, K., & Doyle, W. (1996). Personal narrative and life history in learning to teach. In J. Sikula, T. Buttery, & E. Guyton (Eds.), *Second handbook of research on teacher education* (pp. 120-142). New York, NY: Simon & Schuster Macmillan.

Christopherson, P. (2006). From the VCAA Corner. *INFONET (The Journal of the Information Technology Teachers' Association), 16*(2), pp. 2-3

Fensham, P. J., & Lui, J. (1999). *Teachers' Capability to Transpose the Content of Science Topics in the Curriculum*. Paper presented at the Contemporary Approaches to Research in Mathematics, Science, Health and Environmental Education, Deakin University, Melbourne, Australia.

Knowles, J. G., & Holt-Reynolds, D. (1994). Special issue on personal histories as medium, method and milieu for gaining insights into teacher development. *Teacher Education Quarterly, 27*(1).

Meredyth, D., Russell, N., Blackwood, L., Thomas, J., & Wise, P. (1999). *Real time. Computers, change and schooling. National sample study of the information technology skills of Australian school students*. Griffith University, Brisbane: Australian Key Centre for Cultural and Media

Osborne, R., & Freyberg, P. (1985). *Learning in Science: The Implications of Children's Science.* New Zealand: Aukland: Heinemann.

Powers, K. D. and Powers, D. T. (2000, Nov). *Constructivist Implications of Preconceptions in Computing*, Proceedings of ISECON '00 (Annual Conference for Information Systems Educators)

Ridgway, J., & Passey, D. (1991). A constructivist approach for educational computing. *Australian Educational Computing, 6*(2), 4-9.

Stern, N. (1992). The metaphors of constructivism. *Australian Educational Computing, 7*(1), 22-25.

Tatnall, A., & Davey, W. (1985). *Computer Science for Year 11.* Jacaranda Press

von Glasersfeld, E. (1988). *Cognition, construction, knowledge and teaching.* Washington, D.C.: National Science Foundation. (ERIC Document Reproduction Service ED 294 754)

White, R., & Gunstone, R. (1992). *Probing Understanding*. London: The Falmer Press.