IDENTIFYING A NEED FOR WEB-BASED COURSE SUPPORT

Affleck, G. and Smith, T.

CIS,
University of Paisley
Paisley , Scotland, U.K.

Email: {affl-ci0, smit-ci2}@paisley.ac.uk

Abstract

The focus of this paper is web-based course support for novice computer programmers. The findings from interviews and a test are considered and the problems found are subsequently related to the learning paradigms of constructivism and objectivism. Finally, recommendations are made for the incorporation of web based course support.

Keywords

Education, world wide web, novice computer programmers, objectivism, constructivism, active learning.

1. Introduction

Education has seen several technologies that at advent promised much for education such as:

- Radio
- Television
- Computer aided learning

Each time we have seen an initial explosion of research heralding a new way in which education will be delivered and students will learn. However, the traditional means of higher education (the lecture driven approach) remains dominant. These technologies have been assigned a relatively minor role, most notably in distance education.

The current technology being heavily researched as an educational platform is the World Wide Web. The mass of information and the ease of access and communication would seem to some to be a natural revolution for education. However, the reality for education may be no more than a simple transfer of one medium for another, the learning experience remains unchanged and the faults within that learning experience may become all the more apparent without social support if attempts are made to replace the existing structure.

It is proposed in this paper that the web can have role within the existing educational structure rather than being seen as a replacement (as previous technologies). This role is to provide the course support for existing courses to solve problems that arise within courses that are not easily solvable within the existing structure.

The subject chosen for this study is computer programming. This subject has proved a difficult one to teach at novice level within the educational system. This study focuses on an existing course that has an element of computer programming. The successful provision of web based course support requires the identification and satisfaction of the needs of students and tutors. These will be needs and problems that are not easily rectified within the existing course provision for particular courses.

Research on teaching and learning computer programming indicate that novice programmers have several difficulties with computer programming concepts such as looping, variables, iteration, and manipulating values in computer memory (Putnam et al. (1986), Segal et al. (1992), Du Boulay (1986), Joni & Soloway (1986))

This study consisted of a series of qualitative interviews which were conducted in an attempt to discover the problems that students found in a normal learning environment. Initial interviews with 9 students consisted of one to one at the computer as the student worked through the course lab sheets with the interviewer taking notes on misconceptions and feelings. Following this initial phase of problem finding a small test was given to approximately half the students on the course in order to investigate how many of the problems identified generalised to the rest of the course. The majority chosen for the test were from those that had chosen to carry on to a subsequent course involving programming and all of whom had passed the course assessment. Subsequently a total 43 people were interviewed with 26 interviewed on more than one occasion. The majority of the students were enrolled on a postgraduate conversion course. All interviews were conducted on a one to one semi-structured basis. Notes were taken, transcribed and analysed for common occurrences of misconceptions and phenomena.

2. The test

The test consisted of 4 questions and had no time constraints. The format was closed book with the questions being as follows:

In the first question the students were asked to describe isolated statements of code.

In the second question the students were asked, "What is a method?"

The third question tested understanding of and required a description the outcome of a selection statement.

In the fourth question the students were asked to "Write program code to ask a user for a name and print it to the screen 5 times."

The questions were graded as:

1. Correct
2. Partially correct answer
3. Incorrect
4. No (serious) attempt made

The results

Table 1: Results

	Correct	Partially correct	Vague	No attempt
Q1 "isolated lines of code"	**8**	**14**	**10**	**15**
Q2 "What is a method?"	**7**	**1**	**13**	**26**
Q3 "selection statement"	**10**	**2**	**14**	**21**
Q4 "Write program code"	**10**	**9**	**9**	**19**

These results show that more than half of the students on the course did not gain a basic knowledge of programming and were unable to perform simple programming tasks or read control flow statements correctly. For a large section of the students on the course basic concepts were not understood and could not be applied.

5. Comparison of the results with related research

Several other studies have shown similar findings of novice programmers and research on teaching and learning computer programming indicate similar problems to this study and that novice programmers have several difficulties understanding computer programming concepts:

Putnam et al. (1986) found that high school students in beginning BASIC programming courses had eight areas of misconception:

Assignment statements
Print statements
Read statements
Variables
Loop construction
If statements
Other control flow
Tracing and debugging

Segal et al. (1992) indicated that students had major difficulties using the semi-colon which is the sequencing operator of the programming language.

Du Boulay (1986) identified five general areas of problems for novice programmers:

1. Basic orientation (finding out what programming is.for)
2. Understanding the general properties of the machine
3. Formal language notation
4. Standard structures
5. Mastering the pragmatics of programming

Joni and Soloway (1986) indicated that a poorly constructed working code is often produced by novice programmers. That is novice programmers tend not to focus on the principle of program readability.

6. Interpretation of the interviews and the test

From the results of the test it can be seen that more than half of the students struggled from the start and made no real progression and from the interviews this is interpreted as being generally attributed to two prime factors:

- prior knowledge
- approach to study

Prior knowledge is cited in many learning theories as being of the utmost importance to the learner most notably with the constructivists Jonassen, D. H. (1992) who maintains all learning is built from an interaction with prior experience and current environment and therefore all learning is being built on structures that already exist. Therefore, for optimal learning the learning environment should access the student's prior knowledge and start to build upon it. Prior knowledge was an important factor in this study, with many complaints from students that vocabulary and terminology was used but not explained. Students also complained of being expected to know things already and of concepts out of context and being too abstract. Levels of knowledge and transfers of knowledge that may have seemed obvious to the lecturer were not noticed by the student and caused problems for understanding. The structuring of knowledge became difficult as only isolated pieces of information were being remembered with no real knowledge framework emerging.

Approach to study was an important factor. Two main approaches emerged, the first of which was an approach to read and understand through memorization but very little active involvement in actually writing programs. Those interviewed who didn't learn very well all adopted this approach.

The combination of the two variables (approach to study and prior knowledge) seemed to lead to other problems some students encountered:

- A perceived need to keep up with the lab sheets (and peers) , moving on without gaining an understanding.
- Ambiguities in the course material meant labs were difficult to complete alone.
- A feeling that the lab exercises had simply become typing exercises.
- Feelings of stress and annoyance due to growing disillusionment with the course.
- Group work caused problems such as working and achieving at different rates, having different goals. Coping strategies increased as confidence deteriorated.

The second approach was a more active involvement of trying things at the computer such as writing small programs and doing exercises. All the students interviewed who had learned how to program well both had adopted this approach and talked of hard self-study and struggling to understand (learning programming had not come easy for any).

7. The need for web based support

The way people view the world and the acquisition of knowledge will not only affect the approach of the learner to learning but also the tutor to teaching and the designer to course development. Therefore a major consideration for the practical application of web based course support is the influence of Learning Paradigms on the learner and the learning environment.

The Objectivist paradigm became the dominant paradigm in educational disciplines through the success of behaviourism with such as Skinner and programmed learning. The theoretical goal of behavioural psychology is the prediction and control of behaviour as it attempts to discover lawful causal relationships between input stimuli and output response, which it assumes, will be the same across all organisms. Instructional design based on bahaviourism aims to provide methods that break down complexity of a knowledge domain into components that can be analysed, understood and transmitted to another person. This knowledge is repeatable and can therefore be accepted by everyone. The goal of instruction within such an environment is therefore for the student to gain the correct propositional structure. All students start at the same level and are presented with the same information and are therefore expected to learn the same knowledge. Students who fail to learn are seen as being less capable. The effectiveness of teaching is measured in how well students can replicate instructions. The designer of instruction is able to make abstraction of knowledge from context, a generalisation that people can learn and then transfer to new situations. Testing stands apart from instruction and probes knowledge acquired (depth and amount of processing stimulus events) in an objective way. This paradigm leads both tutor and student to a view that knowledge is to be "told" and memorised and therefore leads to the approach to learning adopted by those who only try to read and memorise for understanding. This is reflected in several research studies that showed it is difficult to fully develop students' programming knowledge through traditional computer programming instruction (Cope and Walsh, 1990)

Recently, constructivism has been growing in popularity within education. Constructivism is a paradigm that strongly promotes experiential, collaborative, student centred, active learning. Constructivist roots are the re-discovered roots from Socrates, Kant, Dewey, Vygotsky and Kuhn (Kang I, 1995)

Constructivists believe there is a real world (environment) that we experience. However, an understanding of the meaning of reality is imposed by the individual rather a mirror image copy of a reality that exists independently. This individual understanding is based upon his/her perceptions and actively constructing knowledge by interpreting perceptual experience in terms of prior knowledge, current mental structures and existing beliefs (Jonassen D. H., 1992)

Therefore with meaning being an interaction between an individual's experience and the individuals interpretation of the environment there are many ways to structure an understanding with many individual perspectives for events and concepts, which help build (construct) an understanding. Meaning is therefore imposed by the individual in

collaboration with the social environment rather than existing independently. Experience must be examined to understand learning. Learning environments should be arranged to encourage a process that promotes knowledge construction based on relevant experience building on prior experience.

8. The need for a constructivist environment

What is the need of the constructivist environment? The traditional environment has been around for a long time and obviously works as many people have learned how to program from within it. So is there a need for a constructivist environment? The goal of an educational environment whether objectivist or constructivist is to aid a student to gain an understanding of a subject. Which way is best? The objectivist aim for economy and simplification of the instructional process, and why shouldn't we simply tell students what they need to know? On a taught course the "experts" have to an extent already negotiated the understanding a student is trying to achieve. This pre-negotiated meaning may therefore be perceived by the student as the "correct" objective view, of which they will be told, expected to memorise and tested on. It is this pre-negotiated meaning that objectivist educators are trying to transmit to students. Students therefore are not entirely free to gain their own understanding. On a taught course they will have to gain an understanding that is compatible with the person who will mark their papers or risk losing marks.

If knowledge cannot be transmitted, information can. Prior constructions that are reconstructed may make what appears to be direct transmission of information possible through the use of previous constructions. Transmittable information may be understandable if prior constructions support it. If I told you my name was Glenn you could use a prior construction (of the concept name) to understand what this information meant. This information would only be meaningful to you if you had constructed the concept of name and how to use it. If no prior construction had existed it would require a more extensive constructive process. Therefore, knowledge can be told to a certain extent and transmission is part of the learning environment from which people gain the information from which to construct knowledge. New concepts and structures can and are taught explicitly, for example through the use of metaphors and analogies new domains can be structured from old using prior constructions. Therefore as the objectivists would point out students can and do sit in lectures and gain the intended knowledge (even if it requires the listener to reflect and construct rather than record). Requirements can also be made explicit and therefore the student will know what is to be constructed.

However, conceptual understanding cannot be gained from transmitted information if the concepts or the processes connecting the concepts involved cannot be made explicit and brought onto the social plane, or when there is no prior constructions to structure the new knowledge. A distinction was made between explicit and implicit knowledge by Tulving (Eysenck M. W., Keane M, T, 1990). Implicit knowledge unlike explicit knowledge cannot be easily "told" and implicit procedural knowledge is built by doing. This implicit knowledge cannot be transmitted. With a subject such as computer programming there is

a great deal of implicit, intuitive knowledge that experts find difficult or impossible to make explicit and therefore cannot be directly transmitted as information for the student to construct knowledge from. The experts simply cannot make verbal this type of experiential knowledge nor could it be understood through only verbal form. For the constructivist, each field has a unique way of knowing which will involve large amounts of intuitive knowledge. In gaining knowledge and becoming competent in a domain it is therefore necessary to go beyond the explicit information given and to construct experiential implicit knowledge.

Constructivism therefore offers a new challenge for instruction and for the application of course support on the web. The challenge is to develop ways of organising learning that allows the contextual practice that is necessary to construct intuitive skills and knowledge that cannot be gained through explicit means. However it is obvious that the constructions have to be made by the learner themselves in an active, motivated manner. The learning environment must be one which promotes constructive learner, a student motivated to actively construct knowledge rather than simply passively receive and memorise.

9. Recommendations

The main problems found in the study that students have with learning to program is accessing prior knowledge and adopting an approach to study that will go beyond memorising explicit knowledge to constructing implicit knowledge necessary to apply and transfer the domain concepts to new situation. It is these problems that course support will need to help solve. Recommendations (based on research to date) for what the implementation of course support should concentrate on are:

Recommendation 1: Stimulate prior knowledge

Stimulating prior knowledge can and should be made explicit, and may be achieved through the use analogies and metaphors, thus using an old environment to structure the new and building on what is already known. This is a constructivist principle but the traditional objectivist environment can continue to be used to teach knowledge that can be made explicit. Visualisations of analogies can be achieved by using computer simulations that can show the concepts in everyday use and compare them to the use in the new domain.

Recommendation 2: Structure the learning

Learning should be structured within the student's developmental range. When new concepts cannot be based on explicit prior knowledge (as with implicit procedural knowledge) it will require the student to gain first hand experience of the environmental domain which is needed to interpret and to construct new concepts.

Simulations and worked examples can be used to provide visualisations and make abstract concepts more concrete.

Simulation can be based on:

- Line by line descriptions of program flow.
- Visualisations of the interactions between program code, input, output and memory.
- Exercises to develop debugging skills, program reading skills and program writing skills.

Recommendation 3: Apply concepts to new situations/problems

The concepts learned should be applied using a task/problem solving approach. The Problems/task presented should again be within a students developmental range.

Using the concepts of a domain and applying them to different situation can aid transfer and build up flexible knowledge (Spiro et al. (1992)) of the type of implicit procedural knowledge that expert programmers possess. Initially the concepts should be applied to problems that the student already understands well so the initial concentration is not on the problem itself but the application of the domain concepts to the problem.

Recommendation 4: Promote an active constructive learner

Learning is dependent on the learner interacting with the environment. At all stages of learning the learner needs to be actively constructing knowledge instead of passively memorising. The environment does not construct the knowledge, the student does. We can however ensure that the optimal amount of stimulus information is available for the student to construct the knowledge and the encouragement to active construction for the information.

Recommendation 5: Improve communication between student and tutor

Communications between tutors and students needs to be improved. Often the tutors on large courses may have little idea of individual student ability. The students themselves may be embarrassed of their lack of knowledge and ability. The web should be used to overcome these problems by providing the monitoring of student progress and communication between student and tutor as needed.

Recommendation 6: Maintain a database of misconceptions

A database of misconceptions and solutions should be maintained. If a large amount of students are having the same misconceptions it could lead to course rewrites or area to concentrate on. Students can query the database to find a solution to problems they are having as needed. The web and database connectivity will allow students access to

different depths of instruction this should allow students to gain as much or as little help as they need.

10. Current and future research

Current research is being carried out on an initial implementation of the course support application which uses an "Add-In" to extend the programming environment and has been designed to provide (from within the programming environment) specific course help to implement the recommendations made. The students have access to web page help, tutorial explanations, exercises and contact to the tutor, with a database of misconceptions being maintained. The course help can be accessed by the students from either the university intranet or via the internet. This course help is currently being evaluated. Future research will include the incorporation of an "intelligent pedagogical agent"

(a subset of software agents) to help provide the user interface, monitor student behaviour, monitor student progress contacting the human tutor as needed, provide emotional support, provide communication with tutors and peers as well as acting as an assistant tutor and advisor to help scaffold the individualized learning environment.

References

Cope, P. and Walsh, T. (1990) 'Programming in schools 10 years on'. *Journal of Computer Assisted Learning*, 6, 119–127.

Du Boulay, B. (1986) 'Some difficulties of learning to program'. *Journal of Educational Computing Research*, 2(1), 57–73.

Eysenck, M.W. and Keane, M.T. (1990) *Cognitive psychology: A Student's Handbook.* Erlbaum, pp. 250–251.

Jonassen, D.H. (1992) 'Evaluating constructivist learning', in Duffy, T.M. and Jonassen, D.H. (eds) *Constructivism and the Technology of Instruction: A Conversation*, pp. 137–148.

Joni, S. and Soloway, E. (1986) 'But my program runs. Discourse rules for novice'. *Programmers Journal of Educational Computing Research*, 2(1),95–125.

Kang, I. (1995) *The Constructivist Principles and the design of Instruction*, PhD Dissertation Indiana University, p. 22.

Putnam, R.T., Sleeman, D. Baxter, J.A. and Kuspa, L.K. (1986) 'A summary of misconceptions of high school basic programmers'. *Journal of Educational Computing Research*, 2(1), 459–472.

Segal, J., Ahmed, K. and Rogers, M. (1992) 'The role of systematic errors in developmental studies of programming language learners'. *Journal of Educational Computing Research*, 8(2), 129–153.

Spiro, R.J., Feltovich P.J., Jackobson, M.J. and Coulson, RL. (1992) 'Cognitive flexibility, constructivism and hypertext random access instruction for advanced knowledge acquisition in ill-structured domains', in Duffy, T.M. and Jonassen, D.H. (ed.) *Constructivism and Technology of Instruction a Conversation*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 57–75.