Using Experiential Learning to Improve Understanding of the Software Development Process

Dr. D. B. Lowe, Mr. J. R. Leaney, Mr. A. J. Boswell

School of Electrical Engineering,
University of Technology, Sydney

Electrical engineering students have traditionally had difficulty in obtaining a deep understanding of the principles of software development. Although they understand and can use the required techniques, they do not have the experience required to develop an acceptance of the need for these techniques. This paper details the redevelopment of the introductory computing subject, and in particular focuses on the introduction of teaching innovations aimed at assisting the students in obtaining a deeper understanding of the issues and ramifications of software engineering. The techniques used include case studies of expert solutions, peer review, and software walkthroughs. The techniques are outlined, and the results of preliminary trials described. Finally, the application of these techniques to other aspects of engineering education are considered.

Introduction

This paper describes the redevelopment of the UTS Electrical Engineering subject "Software Development 1". This subject is the first computing subject for Electrical Engineering (EE), Computer Systems Engineering (CSE), and Telecommunications Engineering (TCE) students. The teaching innovation described aims to engender in students a deeper understanding of the software development process through the use of small group-based interactive workshop sessions. This will be evidenced by students going beyond the ability to use techniques in a structured assignment to an understanding of why these techniques are necessary, and where they fit into the overall context of software development.

Background

History

The authors belong to the Computer Systems Engineering (CSE) group within the School of Electrical Engineering at the University of Technology, Sydney. In 1982 the school switched its initial teaching language to Pascal. Since then the degree course has undergone a number of restructurings and has spawned two additional degree courses (Computer Systems Engineering, and Telecommunications Engineering). Throughout these changes the initial language remained Pascal and the method of teaching it remained essentially the same, although the material being taught evolved significantly.

When it was first introduced (in a subject called "Computer Programming") it was a 3-hours per week, 16 teaching weeks subject. Along with the language syntax, only minimal material on structured design was included (such as the use of top-down design, structure charts, software life cycles etc). Progressively more material on software engineering was added; data flow diagrams and data dictionaries, pre and post conditions, coupling and cohesion, abstract data types. Additionally the use of the language was expanded to cover data structures; queues, linked list, stacks etc. By the end of 1992, the material had expanded to cover two subjects: "Fundamentals of Computing" and "Software Engineering 1" (each being 3 hours per week, 14 teaching weeks). The details of these subjects can be found in (UTS, EE, 1993).

Review

During 1993, the Software Thread Group within the School undertook a review of the software subjects within the undergraduate degree courses offered by the School. This review had two primary objectives. The first was to investigate the relevance of the material which was being taught. The software development field has

...
advanced significantly over the last decade, and although the software subjects had been incrementally updated, there were a number of changes which had not been incorporated. The most significant of these was the rapid rise of Object-Oriented (OO) programming languages and development methodologies.

The second objective was to investigate the educational effectiveness of the subject structure and teaching methods. During the review a number of significant problems were identified with the software development abilities of the undergraduates. Firstly, although the students emerged with the ability to develop software, this software often lacked "quality" and exposed a shallow understanding of the concept of quality when applied to software. Secondly, the students often indicated that they could not understand the need for the formal development methodologies. Finally, the work of students towards the end of their degree often lacked evidence that they were using these methodologies (Lowe and Leaney, 1993). These three observations all pointed to a weakness in the learning practices. The students tended to produce solutions which were indicative of a level of 2 to 3 (unistructural and multistructural) on the SOLO taxonomy developed by Biggs and Collis (1982). Van Rossum and Schenk (1984) showed that students at level 3 or lower almost universally used a surface approach to learning.

<table>
<thead>
<tr>
<th>Type of student</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>5</td>
<td>27</td>
<td>19</td>
<td>9</td>
<td>5</td>
<td>2.72</td>
</tr>
<tr>
<td>No programming experience</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>2.63</td>
</tr>
<tr>
<td>Programming experience</td>
<td>1</td>
<td>18</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td>2.79</td>
</tr>
<tr>
<td>Students who failed the subject</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2.00</td>
</tr>
<tr>
<td>Students who passed the subject</td>
<td>1</td>
<td>22</td>
<td>17</td>
<td>8</td>
<td>5</td>
<td>2.89</td>
</tr>
<tr>
<td>Students who obtained at least a credit</td>
<td>0</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td>3.25</td>
</tr>
</tbody>
</table>

Table 1: Rating of solutions to essay style questions on aspects of the software engineering process (using the SOLO taxonomy) before the implementation of teaching innovations. (N.B. 1=Use of irrelevant information, or no meaningful response, 2=Answer focusses on one relevant aspect only, 3=Answer focusses on several relevant features, but they are not co-ordinated together, 4=The several parts are integrated into a coherent whole, 5=Answer generalises the structure beyond the answer given.)

Although the students learnt the basic techniques and skills required, this learning was not developed to the stage where the students obtained a deep understanding of the material and it's context. This problem is common in many areas of engineering education, especially those involving the teaching of a process. In the SOLO taxonomy, the higher level responses (4-5) differ from the lower level responses in that they imply a bringing together of isolated components (such as techniques and skills) into an overall framework. We wish to encourage student learning to lead to these higher level responses.

PROPOSED CHANGES

Redevelopment of the Subjects

In order to address the problems outlined above, the software thread subjects (i.e. all software subjects in the course) were extensively redesigned. The initial phase of this process was to analyse the requirements of the software thread. Both thread objectives and teaching objectives were developed and analysed, along with boundary conditions and resourcing issues. The document resulting from this process (Software Thread Group, 1993a) was then used as the starting point for a design of the subjects in the thread. The first two of these subjects to be redesigned were the introductory computing subjects (Software Development 1 - SD1 - and Software Development 2 - SD2) These subjects provide an appropriate introduction to computing, programming languages, and the software development process. During the design of these two subjects two
seperate areas were addressed relating to the problems identified in the initial review; the subject content, and
the teaching methods.

Initially, the material to be covered in SD1 and SD2 was initially broken into 8 separate submodules

- Introduction to Computing
- Object-Oriented Software Development
- The Eiffel Programming Language
- Software Crafting 1
- Introduction to Software Engineering
- Structured Software Development
- The C Programming Language
- Software Crafting 2

These 8 modules were then partitioned into the the two subjects. The first six of these modules simply include
relevant academic content for computing subjects. The primary innovation in these modules is the order in
which we teach the material. This will be discussed first, and then the last two modules listed above (Software
Crafting) will be considered:

Subject Content

Traditionally, in an an initial software subject a programming language is taught. *After* the language has been
mastered, a development methodology is often taught. It is the Authors belief that this is the wrong way to
approach teaching this type of material. Developing software is inherently a process. By teaching the mechanics
of programming and not the rest of the process (problem analysis, design, etc.) the students tend to develop a
misunderstanding of what the process requires.

In the redevelopment, the subject SD1 teaches an Object-Oriented language (Eiffel) and an OO development
methodology simultaneously, and integrates these together. Similarly, the subject SD2 teaches a structured
language and appropriate development methodology together.

It is worthwhile mentioning one other important aspects of the order of material presentation. It has
traditionally been significantly more common to teach a structured approach to software development (SASD),
and then an object-oriented approach (OO). We have reversed this order. As discussed previously, the students
were previously learning the skills required to use the methodologies taught, but were not developing an
understanding of why the formal methodology is required. This problem is partially due to a poor
understanding of how the process results in quality. This would in turn appear to result from an overly complex
process model. The structured methodology is generally accepted as having a poor continuity. The relationship
between the software and the real-world system being modelled is, at best, convoluted. During the development
process, two major changes occur in representation, notation, and structure of the system. As a result of this,
the students are usually unable to develop a deep understanding of the process model.

During the current redevelopment of SD1 and SD2, we decided, on the basis of the above issues, to change the
order in which we teach the development methodologies. We shifted from teaching SASD first to teaching OO
first. The OO model is significantly more straightforward. Throughout the entire development process
(analysis, design, implementation, testing, and maintenance) the same representation, notation, terminology,
and structure is used. Each section of the software has a direct relationship with a part of the real-world system.
As a result of this we would expect that the students would find it significantly simpler to obtain a deep
understanding of the process. The process is much simpler to model. The students should be able to spend less
effort on understanding the model, and more effort on understanding its context, relevance, and application.

Subject Structure

At the same time as redeveloping the subject content, we looked at the teaching strategies used in the subject.
The major innovation in this respect was the introduction of a series of small group-based workshops into the
subjects. These workshops include peer reviews, critical analysis of concepts, software walkthroughs, and case
studies. These are aimed at getting the students to focus on the need for, and the context of, appropriate
software development methods (Biggs 1990). This will be achieved by encouraging the students to critically
analyse the subject material, deconstruct case studies, and to analyse what both they and their peers are doing.
The resultant subjects moved from being a lecture-tutorial structure to lecture-tutorial-workshops.
The workshops sessions are integrated into the subject. Some of the workshops (such as a critical investigation discussing software quality) replace lectures and are used to introduce new material. Other sessions (such as most of the case studies) have replaced tutorials and are aimed at reinforcing material covered elsewhere. The overall structure of the subject is as follows:

<table>
<thead>
<tr>
<th>Wk</th>
<th>Session 1a</th>
<th>Session 1b</th>
<th>Session 2a</th>
<th>Session 2b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Admin</td>
<td>Tutorial</td>
<td>Lecture</td>
<td>Investigation</td>
</tr>
<tr>
<td>2</td>
<td>Lecture</td>
<td>Laboratory</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>3</td>
<td>Lecture</td>
<td>Tutorial</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>4</td>
<td>Lecture</td>
<td>Laboratory</td>
<td>Lecture</td>
<td>Investigation</td>
</tr>
<tr>
<td>5</td>
<td>Lecture</td>
<td>Case Study 1</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>6</td>
<td>Lecture</td>
<td>Quiz 1</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>7</td>
<td>Lecture</td>
<td>Peer Review</td>
<td>Lecture</td>
<td>Case Study 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk</td>
<td>Session 1a</td>
<td>Session 1b</td>
<td>Session 2a</td>
<td>Session 2b</td>
</tr>
<tr>
<td>8</td>
<td>Lecture</td>
<td>Tutorial</td>
<td>Investigation</td>
<td>Walkthrough</td>
</tr>
<tr>
<td>9</td>
<td>Case Study 3</td>
<td>Case study 3</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>10</td>
<td>Lecture</td>
<td>Quiz 2</td>
<td>Lecture</td>
<td>Peer Review</td>
</tr>
<tr>
<td>11</td>
<td>Lecture</td>
<td>Tutorial</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>12</td>
<td>Lecture</td>
<td>Investigation</td>
<td>Lecture</td>
<td>Tutorial</td>
</tr>
<tr>
<td>13</td>
<td>Lecture</td>
<td>Walkthrough</td>
<td>Case study 4</td>
<td>Case study 4</td>
</tr>
<tr>
<td>14</td>
<td>Lecture</td>
<td>Revision</td>
<td>Revision</td>
<td>Revision</td>
</tr>
</tbody>
</table>

The Workshop Sessions

Crafting workshops are small group-based sessions aimed at promoting experiential learning. They are called workshops because they are intended to be interactive and group-based. They are called crafting because the ultimate goal is to help the students develop the ability to craft quality software. The focus has shifted from the product (the software itself) to the process (the crafting) which is used to produce the software. The workshops aim to show how the process contributes to obtaining an appropriate product. The three main types of workshops which are used are critical investigation sessions, peer reviews, and case studies.

The critical investigation sessions are aimed at getting the students to investigate and discover for themselves ideas about the development process. A given topic is considered in open discussion by the students (with the tutor acting only to facilitate the discussion). The topic is dissected in detail and active participation by all students is required. These sessions focus on helping students to understand the need for software engineering.

During the peer review sessions the students are expected to analyze each others work. Each student explains their own work to others in the group, who are expected to critique it. The students look at the techniques, used by others, in the solution of problems. The producer of the solution is expected to explain the approach that was taken. These sessions help the students understand the rationale behind adopting different approaches to obtaining a solution.

The case studies are similar to the peer reviews except that the solution being considered is that of an expert. Each case study is dissected by the group, and it is intended not just to look at how the solution works, but why the solution was chosen. One way of viewing the difference between peer review and case studies is that peer reviews broaden the students awareness of the many possible solutions, and the case studies then narrow the focus onto a specific optimal solution.
RESULTS

The initial implementation of the techniques outlined have provided promising results. The use of workshops and changes to the subject content has led to a noticeable improvement in the students learning. This has been exhibited in a number of ways; higher motivation, keener interest in the rationale behind the techniques being taught, and a deeper understanding of the overall context.

In order to further analyse the effects of the subject redevelopment, student essays and descriptive style exam questions were analysed to determine their ranking on the SOLO taxonomy. Table 1 (given earlier) showed the rating of students' work prior to the introduction of the crafting workshops (the problems discussed previously are quite evident). At the time of writing this paper, the subject containing the teaching innovations was being run for the second time, so results are quite limited. These are summarised in Table 2.

<table>
<thead>
<tr>
<th>Type of student</th>
<th>Number of students at given level of SOLO taxonomy</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>2 4 9 11 4</td>
<td>3.30</td>
</tr>
<tr>
<td>No programming experience</td>
<td>2 3 6 7 3</td>
<td>3.29</td>
</tr>
<tr>
<td>Programming experience</td>
<td>0 1 3 4 1</td>
<td>3.56</td>
</tr>
<tr>
<td>Students who failed the subject</td>
<td>1 1 2 0 0</td>
<td>2.25</td>
</tr>
<tr>
<td>Students who passed the subject</td>
<td>1 3 7 11 4</td>
<td>3.54</td>
</tr>
<tr>
<td>Students who obtained at least a credit</td>
<td>0 1 4 6 3</td>
<td>3.79</td>
</tr>
</tbody>
</table>

Table 2: Rating of solutions to essay style questions on aspects of the software engineering process (using the SOLO taxonomy) after the implementation of teaching innovations. (N.B. 1=Use of irrelevant information, or no meaningful response, 2=Answer focusses on one relevant aspect only, 3=Answer focusses on several relevant features, but they are not co-ordinated together, 4=The several parts are integrated into a coherent whole, 5=Answer generalises the structure beyond the answer given.)

These results indicate a significant improvement in the learning practice of the students. Whereas previously 49 % of students were rated 1 or 2 on the SOLO scale, and only 22 % of the students were rated 4 or 5, the figures after the subject redevelopment are 20 % of students rating 1 or 2, and 50 % of the students rating 4 or 5. It is important to remember however that this based on relatively small samples. Evaluation of the changes is ongoing and further data will need to be collected in order to validate the above results.

In order to further validate these results, student surveys will be adapted to specifically look at the students reactions to the new teaching methods, and the depth of their understanding of the material covered (again using the SOLO taxonomy as a basis). Finally, in the longer term the success of the project can be evaluated by looking at the degree to which the students continue to use the techniques they have learnt after they have completed the subject. We intend to monitor students software development style in later stage subjects.
EXTENSIONS TO OTHER AREAS

The work outlined in this paper covers innovative teaching methods applicable to a broad range of process-based subjects. Although the workshops developed as part of the SD1 redevelopment are specific to the content of the subject (Lowe 1994), the techniques and skills developed are readily transferrable to other subjects. This is particularly true of subjects involving the teaching of a process.

It was, in part, the lack of understanding of the context of the process which drove the SD1 redevelopment. The workshops are subsequently especially suited to subjects where students learn skills and techniques, but have difficulty developing a deep understanding of these skills. The workshops have been structured in such a way that allows them to easily replace existing subject tutorials with minimal difficulty. Thus these techniques can be readily integrated into almost any traditional lecture/tutorial based subject.

CONCLUSIONS

This paper has discussed the recent introduction into the Electrical Engineering computing subject "Software Development 1" a series of crafting workshops. These crafting workshops were intended to address a series of problems revolving around the students poor understanding of the context of, and need for, a software engineering process. The workshops were developed and formed part of the subject for the first time during the Autumn 1994 semester. A comparison of appropriate components of the students assessments (using Biggs' and Collis' SOLO taxonomy) has indicated a significant improvement in the quality of the students learning.

This innovation will continue to be refined and evaluated. It is also being introduced into the second computing subject ("Software Development 2": 45133) as from the Spring 1994 semester. An investigation of its appropriateness for other subjects involving the teaching of a process will also occur.

BIBLIOGRAPHY

Biggs, J.B. (1989) 'Approaches to the enhancement of tertiary teaching', Higher Education Research and Teaching 8:7-25


Lowe, D.B. (1994) Software Development 1: Course Information and Notes, School of Electrical Engineering, University of Technology, Sydney


University of Technology, Sydney, School of Electrical Engineering, "School Handbook", 1993