Tertiary education for software engineers
N.V. Balasubramanian

City Polytechnic of Hong Kong, Hong Kong

Abstract

The paper outlines the author’s experience over eight years starting from the time he led a team of academics to design the curriculum of a bachelors degree with a business oriented software engineering thrust, all the way through its validation, implementation, subsequent amendments, revalidation and professional accreditation. The importance of inter-personal skills is brought out as well as the manner in which curriculum design and assessment techniques promote such skills. In particular, attention is focused on industrial placement in the sandwich degree with its tremendous potential for nurturing students’ teamwork and presentation skills.

1. Introduction

Hongkong, as with industrialised countries, has traditionally relied on vendor oriented training for its programming community; these ‘professionals’ had little formal education in computing; in time these programmers moved on to become systems analysts as they gained sufficient experience in application development. Given the calibre of the people involved and the centralised batch-processing nature of computer applications, lack of an appropriate academic qualification and a properly structured training programme thereafter did not hamper their work severely. However, with increasing application complexity, more integration between applications, more and more applications going on line, and progress in data base technology, open systems and software development methodologies, a fresh approach is called for in order to produce computing professionals for the 21st century. Also, the engineering discipline as a whole has been coming under close scrutiny. This is well articulated by Dr the Honourable Samuel Wong, Legislative Council
member in Hongkong (reference[1]). He argues that well qualified engineers armed with advanced technological skills are unable to get their points of view across to decision makers; also, they are ineffective in team situations. The example he quotes is a HK$127 billion project now being undertaken in Hongkong in building a new airport, sea port facilities, highways, bridges, railway system and other infra-structure. Engineers are bypassed while making key decisions, when basically it is nothing but a mammoth engineering project. Dr Wong argues that this is a consequence of low self-esteem on the part of engineers and poor perception of engineers on the part of the public at large. He compares engineers with architects, lawyers, doctors and accountants. Teamwork, communication skills and the ability to learn (and take the initiative to do so) are of paramount importance, he concludes. It should therefore not come as a surprise that the image of software engineers will also depend more heavily on such factors than on pure technological skills. Frequently, hardware and software engineering disciplines have been compared and contrasted. It is argued that the hardware engineer has a strong foundation based on physics and mathematics, carries out his/her work in a systematic manner and applies quality control procedures throughout the design and manufacturing life cycle. Why cannot the software engineer do likewise? If it were that simple, we could narrowly teach computing students set theory and predicate logic and use them as the basis for teaching ‘Z’ or ‘VDM’. Many would argue that all our current software engineering problems will disappear if we did the above at undergraduate level. Well, we have done this but the problems still persist. In my view, it is unfair to compare hardware and software engineering. In the former, you are dealing with voltages, logic circuits and timer clocks; in the latter you are dealing with deterministic logic on one side and human beings on the other side. The hardware engineer could care less whether the processor chip is going to be used for an accounting application or an executive information system. Recognition of this fundamental difference between the two engineering disciplines is the crux of the matter in designing the curriculum and planning teaching strategies and assessment techniques.

2. BSc (Honours) in Computer Studies

The original design of our curriculum dates back to 1986. The first major decision was to choose between a sandwich and a non-sandwich degree structure. Given our business oriented software engineering thrust, it became obvious to us that certain skills could be better developed in an industrial setting. Such placements are popular in journalism, nursing, hotel & catering management, physical therapy, marketing, international business, engineering and computing in the U.K. In a recent visit to the Northeastern University (Boston, USA) the author found that some American universities too have a tradition of well structured ‘co-operative’ education programmes (reference [2]). As employers in Hongkong prefer a long sustained placement period to
cover a broader scope of software development life cycle, we settled for a ‘thick’ sandwich degree programme. The first two years had no electives and the concentration was on providing the foundation for later studies as well as preparation for the placement period. The final year provided a choice of electives for specialisation as well as the honours project. During the revalidation in 1992-93 heavy use was made of the joint ACM/IEEE-CS recommendation while preparing the syllabuses (reference [3]).

3. Software Engineering

The coverage of software engineering is best illustrated using the “V” model of software development life cycle proposed by C C Tonies of Hughes Aircraft company (reference [4]). A simplified version of this model is reproduced in figure 1. In the 2nd year software engineering module, the emphasis is on synthesis and testing which are represented by the lower half of the “V” model; the final year software engineering module focuses on analysis and evaluation represented by the upper half of the “V” model. Students learn to semantically model the software specifications in the form of “Z” schemas. They learn to relate these to pre-conditions, post-conditions and invariants in algorithms. Students are also exposed to metrics --- Halstead, function point and cyclomatic complexity. Three design approaches are emphasised --- data structure, data flow and object oriented. Students use CASE tools for individual aspects of the software development life cycle.

4. Industrial Placement

Many aspects of software engineering need exposure to real life situations not possible in the campus. Passing of the third year placement is mandatory for graduation. The polytechnic takes the responsibility for the placing of students while the salary is negotiated between the individual student and his/her employer. In addition to industrial supervisors each student is provided an academic supervisor. This promotes a healthy interaction between students, academia and the industry. Students work in real computer-oriented environment, relate the concepts and principles learnt in years 1 and 2 to this real environment, put into practice some of the methods, tools and techniques of the computer related disciplines and sharpen their inter-personal skills. They gain insights into group and organisational behaviour in a work environment. Students become better sensitised to the business world. (There are 2 modules in the first year of the course, one focusing on business functions and the other on accounting systems.) There is a considerable amount of effort in soliciting placements from companies, defining and articulating the scope of projects to be undertaken, matching these requirements with students’ strengths and interests, placing the students, supervising them and assessing their performance. These are all clearly documented in the form of a handbook. Students have in the past clearly
demonstrated significant gains in self confidence, maturity and self discipline. They enhance their competence in the skills oriented aspects of the course. Through these experiences, they are better able to cope with the more analytical and evaluative aspects of the final year. Their documentation capability is enhanced through:

- half-monthly logs of activities / progress,
- initial report, after 3 months, on company organisation / department employed,
- mid-year report on achievements and application system description, and
- final report similar to the above plus an evaluation on how the first two years of study contributed to their placement and how the placement filled the gaps in their knowledge.

Students also have the opportunity to browse through documentation prepared by their senior colleagues. Frequently, students are required to make oral presentations during the placement period. The value of placement has been most effectively articulated by President Mary Robinson of Ireland when she inaugurated the 8th World Conference on Co-operative Education in August 1993: “It appears that in a world where lifelong learning is the imperative of the future and real work becomes more undefinable, the dividing line between education and work is becoming more undistinguishable.”

Placement is also a valuable feedback mechanism for course monitoring. For example, we replaced the 2nd year psychology module with one on Human Computer Interface which the students found more applied and therefore more relevant. Another change made was to spread the two English modules over two years (rather than over two semesters in the first year). This enabled the lecturers to recycle year-1 material in the following year.

5. Final Year

The final year is also referred to as the ‘honours’ year in which analytical and evaluative aspects of the course are given prominence while planning teaching strategies and assessment techniques. The placement year would have prepared the students well for this transition. Two main areas form the focus in the final year --- software engineering and distributed systems. Reference has already been made regarding the shift to the upper half of the “V” model of software development life cycle. In the other area, the syllabus includes locational, functional, data and control oriented distribution, transparency requirements, design goals, loosely coupled architecture, workstation / server model, process pool model, connection patterns, naming, component name passing, interprocess communication, process synchronisation, open systems technology, object request brokers, standards, distributed transaction processing, update synchronisation and case studies. The key stone module is the final year ‘honours’ project. In this, students individually explore an area of computing of their own choice, develop their skill and knowledge in the chosen area, exercise good project management methods to the planning,
development and monitoring of progress, and apply good technical writing and oral presentation skills. Typical reports are 20,000 words long plus appendixes. Students undertake a potentially useful piece of work which could be of practical use to a real person either in the polytechnic or in some external agency. This will promote problem solving, critical analysis and informed judgement. Project makes a significant contribution towards award classification. Assessment of project is under five headings:

- management of the project,
- individual development of the student,
- technical merit,
- intermediate and final documentation, and
- oral presentation / demonstration.

The viva-voce at the end of the presentation is mainly to confirm that the work was indeed undertaken by the student himself/herself. The final year also includes elective modules for specialisation; placement enables students to make a more informed choice of these. The English communication module in the final year focuses on user training, writing user manuals and project report writing.

6. Social Analysis

In parallel with the industrial placement, there is a module on Science, Technology and Society. Students gather on alternate Saturdays to attend 14 two-hour sessions. This module elucidates contemporary science and technology as social institutions; and investigates the role of science and technology in social development and social change. The academic staff from the department of Social Studies discuss with students the impact of social, political and economic interests on the development of science and technology, and explore the legal, societal, ethical and professional issues which arise in the work place of a computer professional. Frequently, two lecturers are present and do role playing; arguing pros and cons. Guest lecturers are also brought in from the concerned professions to provide a rich teaching environment. Teamwork is encouraged in many modules through coursework assignments. While the campus limits team members to fellow classmates, the variety of team members in the industrial setting during placement exposes students to a plethora of views from users, managers, systems analysts and programmers. Students thus have an opportunity to listen to conflicting views, see how these are resolved and observe how their ‘seniors’ articulate their thoughts. During the accreditation visit of the British Computer Society panel, much attention was paid to inclusion of social analysis and the development of social skills. As stated in the objectives of the course, students should be able to recognise and take on the responsibilities of a professional, maintain currency with technological and theoretical developments and, thereby, uphold professional standards upon graduation. Learning to learn is a vital attribute of any professional. The
honours project is singularly successful in promoting this attitude as well as in taking personal responsibility for own achievements.

7. Learning Strategies / Assessment

In some of the modules, students are given a list of journal articles. After they have read these in the library, the students come to small group tutorials and carry out debates. In some cases, they are divided into two groups. Each group does a critique of the other. Alternatively, one group presents the pros while the other discusses the cons. In such tutorials, the lecturer takes on the role of the moderator. He/she also does a summary at the end. A Study Process Questionnaire (SPQ) was designed by Professor Biggs of the Hongkong University. When we administered this SPQ in our course, the second year students were found to be taking a ‘shallow’ approach to learning. This was explained by us as the students’ way of optimising their time while achieving the highest grades possible. The placement period was successful in promoting a ‘deep’ learning approach. Ms Tang, a doctoral student under Professor Biggs, has evolved an Assessment Preparation Strategies Questionnaire (APSQ). We have received a small scale research grant to investigate whether the form of assessment influences the learning approach of computing students; this research will make heavy use of SPQ and APSQ. In 1994-95, the eight lecturers involved in the 2nd year of the course will be informed of the research project but will not be required to subject their coursework assignments / exam papers for vetting. (However, their awareness of the study in itself may result in better assessment strategies.) In 1995-96, the research team together with the eight lecturers will amend the assessment items and redesign them to promote ‘deep’ learning (e.g., more open book exams). At the conclusion, we will prepare guidelines on appropriate ways to structure assessment items. We believe that this will better prepare the students for industrial placement.

8. Conclusion

The paper describes the rationale in the development of curriculum, teaching strategies and assessment techniques of a bachelors degree in computing at Hongkong. Nurturing of inter-personal skills and the role of industrial placement in a sandwich course in promoting such skills are also described. Proper administration of placement is critical in tapping the full potential of the placement period. Good assessment techniques promote ‘deep’ learning. This is most noticeable in case of the final year honours projects which are individually carried out by students, have no structured syllabus and make significant contribution to honours classification. Real life projects increase students’ motivation. Quality assurance of the degree programme should encompass the building of inter-personal skills and not just dwell on technical
knowledge gained. It has been argued elsewhere (reference [5]) that several other initiatives such as a structured training programme upon graduation and a modular masters degree in software engineering must all be undertaken to gear the software engineers for the 21st century. Close collaboration with the industry is a must for these initiatives to take deep roots. In fact a joint R & D Centre on Asian Language Computing funded by UNISYS Ltd. has inspired several projects undertaken by our students.

References:


![Figure 1: “V” Model of Software Development](image-url)