Business computing education: a radical approach for efficient streamlining of an effective education process and relevant curriculum

Roy I. Morien

Naresuan University Language Centre,
Naresuan University,
99 Moo 9, Tha Po,
Phitsanulok, 65000, Thailand
Email: roym@nu.ac.th

Abstract: College courses in computer system development are divided into discrete, term based subjects, with systems analysis and design, programming and database design as separate and distinct subject areas presented by different academics with different experience and interests. Seemingly appropriate pre-requisite knowledge is often forgotten, or was poorly learned, and theory is taught on the promise that it will be useful in future subjects. Computer system education methods are fundamentally anachronistic and ignore industry practices and activities. A single systems development subject amalgamating these distinct subject areas is suggested, and ultimately the course should advance students through a learning program based on a substantial, multi-year project of sufficient size and complexity to provide students with practical, hands-on, deep learning of the essential characteristics of systems development required in industry. The educational motif is comprised of three significant aspects: project based learning (encompassing problem based learning), student centred learning and continual assessment.

Keywords: integrated development; computing education; problem-based learning; PBL; project-based learning.

Reference to this paper should be made as follows: Morien, R.I. (2017) 'Business computing education: a radical approach for efficient streamlining of an effective education process and relevant curriculum', *Int. J. Advanced Media and Communication*, Vol. 7, No. 1, pp.38–55.

Biographical notes: Roy I. Morien has more than 35 years in the IT industry, including nearly 30 years as a teaching academic in a prominent Australian university, and teaching in Thailand, Singapore, Hong Kong and Indonesia. He has maintained consulting interests in Australia, Singapore and Thailand concurrent with his academic endeavours. His current position is Academic Editor and Language Specialist in the Naresuan University Language Centre, in Thailand, a prominent provincial university. His primary and immediate research concern is the place of agile methods in university and college IS curriculum, and Agile Education; the application of 'organisational agility' to the education process.

This paper is a revised and expanded version of a paper entitled 'Streamlining business computing education' presented at *KSII The 6th International Conference on Internet (ICONI)*, Taipei, Taiwan, 14–16 December, 2014.

1 Introduction

A typical course curriculum plan for courses in business computing, information technology, information systems and the like usually includes systems analysis, design, programming and database design as distinct and separate subject areas. Overall, most curriculum is taught in discrete time limited subjects, by different academics with diverse experience, and often divergent intentions and philosophies. Many subjects are on theory, with a promise that the subject matter will prove to be useful and applicable in future subjects, and in the future generally. Theory subjects particularly are taught using the conventional 'chalk and talk' lecture approach.

The proposition offered in this paper is that there should be first, an amalgamation of most aspects of business computer or information systems education (especially but not only database development, systems analysis, systems design and programming) in a practical, hands-on project-based course, that advances students through a learning program, building on students' deep learning seamlessly over an extended period of time. A 'just-in-time' learning strategy firmly based on a project-based learning (PBL) approach is suggested.

The nature of information systems development, particularly for business applications, is eminently practical, which implies, even demands, a practical, hands-on, project-based teaching and learning approach, with continuous assessment allowing immediate remedial education if a student has not grasped, practiced and become adept at a particular concept or practice.

Events in the information systems industry, as we could generally term it, have overtaken the long-held view of systems development being an orderly progression through a Waterfall model of discrete, linearly sequenced phases. Rather, computer systems development is now an amalgam of analysis, design, code construction and tool use applied in an iterative fashion; it is a tool-enabled integrated activity, encompassing all aspects together and at once. Furthermore, software development is a complex activity, termed a 'wicked problem', and this needs to be reflected in information systems education. As well, it would be rare for a system developer in an organisation to work alone. Usually, the developer is part of a team. This, too, needs to be recognised in information systems education.

Course designers the world over have a problem in ensuring that their courses are both comprehensive and coherent, while at the same time including all essential curriculum as well as meeting the dual requirements of academic acceptability and industry acceptability and relevance. There needs to be an efficient but effective streamlining of curriculum to ensure this. Trying to break Information Systems curriculum down into a number of discrete, bounded, simplified sections fails to acknowledge this 'wicked problem' notion, and the integrated nature of the software development activity, and students are given an education that does not reflect the industrial world, nor does it prepare the student for their forthcoming career. Theoretical learning based on the traditional 'chalk and talk' lecture approach is not appropriate. The development of small, simplistic, easily understood, semester-based projects, usually considered to be the 'practical' side of the learning activity, does not allow students to gain the experience essential to proceed to the next 'level', which is supposedly the next subject probably in the next semester. Trivial projects bounded by the limited time scope of a semester or term fail to provide students with appropriate and useful experience, and the deep learning necessary in the practical systems development environment.

Changing project requirements during the semester is seen as being unfair and wrong. Exam-based assessment encourages, indeed almost requires, shallow learning for immediate application in the imminent exam. The usual practices of giving 'exam hints' and having the 'swot vac' (a study period prior to the exam period) almost formalise this shallow learning.

Addressing the situation from an educational psychology or educational best practice viewpoint, the further proposition in this paper is that an effective, perhaps the most effective, pedagogical approach is the project-based approach (Patton, 2014; Blumenfeld et al., 2014; Thomas, 2000; Thomas et al., 1999). Combined with an amalgam of student-centred learning (Felder, 2014; Froyd and Simpson, 2013), perhaps even student-driven learning (Nielsen, 2014), and continuous assessment (Alausa, 2014; National University of Singapore, 2014), this holistic approach overcomes many of the educational problems usually experienced by students. Students also must accept a significant degree of self-responsibility for their learning achievements, and should be provided with an appropriate, relevant and interesting learning environment to support this.

These observations and opinions are based primarily on this author's personal experience and convictions developed over more than 30 years, regarding software development methodologies and the associated software project management implications arising from the adoption of a particular development approach. These opinions and viewpoint have also arisen from the obvious changes over time in the type of systems being developed in the 'real' world, into which the university or college graduate will enter, and particularly the extraordinary advances in the type and availability of development tools now available to support all aspects of system development implementation, testing and operations, as well as user-based development, automatic code generation and the impact of Cloud Computing. To top off this list of almost revolutionary changes is the style of the 'new' development approaches now being adopted, albeit rather hesitantly in many countries.

The development approach favoured by the author is Agile Development. This author has clearly demonstrated the success of the suggested style of IS education, albeit on a smaller scale than envisaged in this paper (Morien, 2004a, 2004b, 2005b, 2006b). These previous papers illustrate how the concepts and practices of Agile Development have been both taught in the curriculum, and immediately applied to both the Teaching and Learning processes. Two highly desirable outcomes were achieved; first, the students maintained a high level of interest, and almost excitement with what they were doing and achieving, and secondly, almost all achieved a high standard of outcome; useful to the clients, useable by the clients and valuable in the various businesses. An 'agile' pedagogical approach that mirrors many of the characteristics of agile software development methods, which are, in turn to a great degree drawn from concepts of lean thinking, and prior development methodologies and practices such as software prototyping, rapid application development, joint application development and others, have proved very successful in all aspects of the teaching and learning process.

2 Cases in point

To illustrate the problems encountered, following are a number of cases in point drawn from the experience of the author. Each case in point, when encountered, drew the author closer to the realisation that there needs to be a significant change in information systems

and business computing education. A number of these 'Cases in point' are discussed here.

Case in point #1: Early in the rapid application development/software prototyping period (Jenkins, 1985; McConnell, 1996; Naumann and Jenkins, 1982), attempts to introduce software prototyping into the curriculum were obstructed because the faculty staff were organised into four general academic areas; systems analysis, system design, programming and networking. Software prototyping was an orphan! It could not be neatly fitted into the classic Waterfall model of development, as the academic groups were. Software prototyping had too much 'development' for the systems analysis people, and had too much 'analysis' for the programming people. The use of enabling tools, such as code generators and report writers created a situation too liberal for the technology teachers. Placing development capability into the hands of users was almost anathema.

It was soon personally realised at that time (about 1988) that while Software Prototyping represented a whole new way of thinking; seeing systems development as a much more integrated activity demanding an integrated academic approach, it was nonetheless eschewed by most IS and business computing academics, as well as, it must be said, many practicing IS professionals. Regardless of the extensive published research at that time in support of software prototyping, it was not included in university curriculum, not in curriculum standards, such as published by IEEE. This unfortunate situation was commented upon in Morien and Schmidenberg (1994), 10 or more years after a significant bibliography of software prototyping articles and papers had been published and Software Prototyping conference proceedings published (Budde et al., 1984; Jenkins, 1985). This author published relevant papers at that time which initiated his thinking on these matters (Morien, 1992, 2005a; Morien and Schmidenberg, 1994).

Case in point #2: In a systems analysis subject, a small project was undertaken as the 'homework' component of the subject. The students were requested to download some software products from the internet and learn to use them. One such product could be installed as a virtual printer which the students were told would be useful in their project. Unfortunately, most students ignored this, on the grounds that this was not part of the systems analysis curriculum, as published, and what more they could not see the relevance of it because they were not specifically asked to develop printed reports, so did not do so (and possibly because they felt that it would not be a question in the end of term exam, so learning it would be a waste of time). The next subject was the final 'capstone' software project. When a student was asked how she was going to develop, and more particularly to test, reports in the project, the student was puzzled as to the point of the questions. She was asked, quite specifically "Well, how are you going to check that the reports are being formatted correctly?". When the question had been explained to her; basically, the whole topic of reports, writing reports, testing reports, report writing software, need for reports, was just outside her understanding, she replied that she would print them out to review them. However, she did not have a printer but this problem would be resolved by borrowing a friend's printer. When she was asked about using a virtual printer, thereby both not requiring her to borrow a physical printer, and also potentially saving large quantities of paper, it was obvious that she had completely failed to comprehend the lesson about downloading that virtual printer software and the reasons why it would be a useful thing to do.

Two further software products recommended to the students were a remote connection product, and an installation manager product. Again, the students failed to do

what was recommended, it not being examinable, not specified as part of the required learning, not aware of the relevance of these products. Unfortunately, again the practical considerations were not an essential part of the Teaching and Learning activities of the systems analysis subject and were not specifically relevant to this 'theory-based subject' and so the usefulness and purpose of this software could not be demonstrated *in situ* in the subject. In the following subject, the final 'capstone' software project, some students had to develop a small system for a business in a distant town. They developed the system in the traditional way; find out a bit about the requirements, spend 10 weeks developing it, and then send a disk to the remote user to be installed by them. There was always the hope that the system was useful, although this had not been tested because the user was in a distant town and therefore the students could not keep in close touch with them.

In each of these situations; virtual printing, remote connection and support, and managed system installation, there was no follow-up or use of these products in the subsequent project unit showing that there was an almost total disconnect between the theory and the practical application. The subjects were conceptually and intellectually remote from each other. The relevance of the software was not demonstrated, and whatever lessons had been learned were already forgotten. The teacher in charge of the subsequent project subject made no further mention of such software. Clearly, there was no 'team teaching' concept; there was no communication between subjects or subject teachers, and no thought of curriculum continuity from one subject to the next. In a teaching and learning situation as is being suggested in this paper, this situation would never develop.

Case in point #3: In the previous 'Case in point' part of the problem was that even though the first subject required a small 'hands on' development project as part of the assessable requirements, the teacher was not familiar with the programming language and environment that the students had been taught, and had to rely on what the students remembered of what they had been taught in a previous programming subject, which unfortunately turned out to be very little. The lecturer was there to teach systems analysis methods, and his lack of knowledge of a particular programming language was not a relevant matter in that subject. However, in a 'team-based' teaching approach, where all and any of the teaching faculty are on the teaching team, as would be the case in the 'super' project-oriented approach, this problem would never have arisen. The point must also be made that the author had previously written a textbook on programming, using a particular language for teaching purposes (that is, the lecturer was in fact a highly experienced and very competent programmer), but other lecturers did not like that language, and taught their own preferred language.

Case in point #4: A lecturer from a University in Australia was presenting a paper at a conference about the problems experienced in the Teaching and Learning of database development. The problem can be stated simply; One subject was dedicated to the development of an entity model, and a second subject was dedicated to the design and creation of a database and database processing system based on the entity model developed in the previous subject, and presumably programmed using a programming language taught elsewhere in another subject. However, it was usually found that the ER model had to be significantly modified to be useful in the second subject, even though the students had received good marks for their ER model. The perplexed lecturer was seeking a solution to this predicament. He did not see that by combining the subjects into a single

development of the database in an integrated way students could be taught the whole process to completion without barriers or 'Chinese Walls' between the subjects that teach each aspect of this development. That is, the integrated, project-based approach!

As well, the fact is that the ER model, the data model and the physical database can be developed in an iterative, incremental manner, not requiring a fully developed (apparently) ER model requiring a full 16-week semester to develop. By having this integrated and practical development approach many good ideas and practices of database processing could be both demonstrated and implemented in the programming, which would also extend the students' programming knowledge and skill.

Case in point #5: At the commencement of a subject entitled Advanced Database Design for final year students, it was immediately obvious that the students had virtually no understanding of database design, notwithstanding that they had previously successfully completed an introduction to database development subject. They quite simply stated that it was two years since they had done the Introduction to Database Development subject, and they had forgotten most if not all of what they had learned. This demonstrated two major problems. First, their prior learning had clearly been very 'shallow' and whatever they had learned had basically been in a 'chalk and talk' manner, with a final exam that they had crammed for, and secondly, that was two years ago and they had just forgotten in that time. It can obviously be considered that an entire semester had been wasted teaching the Introduction subject, because it all had to be taught again. Clearly also the time remoteness of the first learning from the advanced learning was a problem with consequences.

Case in point #6: It was observed that none of the system development methods that were taught in the subject Systems Analysis Methodologies were applied in the subsequent 'capstone' project unit that the students were required to do. Neither were any of the project management methods that were taught in the subject software project management applied. Therefore, the subjects where systems analysis methods, software project management methods and documentation standards were taught were, for all practical purposes, a waste of time. Perhaps this also indicated that what was taught in these subjects was irrelevant and not useful, so spending a semester teaching them had also been a waste of time. Perhaps it also indicated that the Project Management teacher, and the 'capstone project' supervisor were not 'on the same page' as far as the development approach taught and to be applied.

Final Case in point: When teaching three separate subjects in a semester, the author was severely criticised for "teaching the same thing in every subject". The response, by way of explanation, was that obviously in the systems analysis subject, systems analysis methods must be taught, and in the database development subject these same systems analysis methods could be applied, so needed to be explained, or revised, and subsequently, in the software project management subject it was necessary to understand these systems analysis methods to be able to manage the developers using that method. Not only did this situation demonstrate further the essential need for integration of these subjects, but also demonstrated that other teaching academics did not understand this matter nor were they knowledgeable of the development methods being taught, even if they knew what was being taught. The overall curriculum was not coherent. The usefulness, correctness or applicability of what had previously been taught is not the issue here. What is at issue are the lack of continuity and the apparent waste of time and

effort in teaching the practices that were then 'forgotten' in the subsequent subjects. Clearly, there was no concept of team teaching or of a teaching team, nor was the curriculum 'stream' seamless and effective.

All of these problems would be overcome in a PBL activity with deep learning, demonstrated relevance in situ, an integrated team teaching approach, and based on a coherent and substantial continuing project developed by the students over the whole period of their course.

The fact is, there are a myriad of 'Cases in point' that could be included here (see Morien (2005c, 2006a) for a more substantial discussion on the major matter of database education). However, the point has been made; so much time is wasted, so much teaching is rendered useless, so much is forgotten by students because of exam-oriented shallow learning, so little understanding of the relevance of one part of the curriculum to another part because of the policy of teaching those aspects of software development in discrete, bounded, time remote and unconnected subjects.

3 Thirty years ago things were different

Another significant change in the development environment that should have impacted the style of teaching and learning, is the obvious fact that the technical side of database development has been basically commoditised, in that there is a substantial marketplace for database design tools; table creation wizards, SQL creators, database administration 'front-ends' and similar development tools simplified the 'technical' side of database construction. What had once upon a time been a subject of major interest, that of the actual creation of the physical database itself, is now almost a 'click the button' activity. The often highly technical and complex activities of building tables, creating indexes and more problematic, modifying tables, and so on is now a simple, straightforward task. Thirty years ago the available database management software included IBM's IMS which was the main DBMS in use at that time, and its use required well-trained technical personnel. Every IBM mainframe shop had dedicated IMS and CICS experts. Creating and maintaining IMS databases demanded a high level of technical skill, as did subsequent DBMS's such as ADABAS and IDMS. However, given today's database landscape, still primarily 'relational', just to teach table design and table construction is insufficient and inadequate as the 'database design' subject because this is no longer a highly technical matter. Also, it has been understood for decades that "the database is the roadmap to the business", as was the title of the paper by Chikofsky (1990), so database development requires close collaboration between users and developers, so is substantially 'analysis', 'design' and 'construction' in combination. Database system development should now be what is taught, and be taught in an integrated teaching and learning approach. Now, 'database design', or better, 'database system development', has a major element of systems analysis in close collaboration with users, a major element of system design enabled by design tools, and then clearly implies programming and algorithm design, now inevitably using GUI-based IDE's. This could be stated as a corollary, which is when teaching systems analysis and design, an online database processing system is an entirely reasonable system to 'analyse and design'. When learning to program, students needed to understand the very important topic of data definition; what is a data field, what types of data field, how to define data properly. In addition, developing processing code to access the database, manipulate the data, extract

and report upon the data is substantially a programming task, supported by appropriate tools. In other words, database design could be combined with systems analysis and design; a database processing system is a particular type of system that can be, should be analysed, designed and constructed in an integrated manner. It is not a topic apart!

A term which has relatively recently arrived in the lexicon of system developers is 'DevOps'. According to Wikipedia, the term 'DevOps' was popularised through a series of 'DevOps Days' starting in 2009 in Belgium. Since then, there have been DevOps Days conferences held in India, the USA, Brazil, Australia, Germany, and Sweden. The term 'DevOps' started appearing online in the Spring of 2010. 'DevOps' is a portmanteau of 'development' and 'operations', and is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) professionals. DevOps is a response to the interdependence of software development and IT operations. It aims to help an organisation rapidly produce *software* products and services'. Again, 30 years ago Operations was conducted behind closed and secured access doors by a group of skilled operators, separate in almost all aspects from the developers. It is doubtful that any university or college course at that time included studies of computer operations, yet today it is an integral and integrated part of the system development activity, albeit in an entirely new and different form. With the advent of the 'cloud' (again a term almost unheard of even five years ago) and the availability of many tools and services to manage and 'operate' virtual systems, an understanding and level of skill in using these tools is essential in the integrated 'DevOps' environment. One commentator states "To achieve the cutting-edge speed and agility promised by DevOps, you need to choose the right tools to enable automation across all aspects of development, production, and operations". This clearly indicates the importance of both this new concept of DevOps, and of a well-trained ability to use the tools. If the significant hands-on use of a variety of such tools is not embedded in business computing education nowadays, that education must be seen as sadly lacking. Of course, this puts even further pressure on the course designer to find the time and a place for this in the course. The necessity to streamline and efficiently proceed through the course, in an integrated manner, is even more paramount now.

The system development landscape of 2016 is so far removed from, and in advance of, the environment of 1996, and from 1976 if we measure the period of change from when SDLC development methods were first starting to be published. Can it be said that the pedagogical structures and methods in 2016 have matched this extraordinary change? This author's experience is that they have not.

4 Points to ponder

Further to what has been expressed above, and to bound the theses in this paper, a list of important educational and software development factors can be stated:

- Teamwork is an essential part of the working environment, so the education experience should include practical experience in teamwork and team participation. Student experience in working in a team, as a team, is essential, and the creation of teaching teams of teachers is almost vital to overcome the knowledge gap between academia and industry, and to provide a seamless teaching and learning environment.

- Trivial development projects that can be specified and taken to completion in just 14 or so weeks do not provide the necessary experience, especially as such projects are part of a subject which is probably one subject in four or five being studied at the same time. Significant projects over an extended period of time should be part of the deep learning experience.
- Realistic understanding of the 'wicked problem' that is software development, where full knowledge of requirements is not possible, and requirements do not remain static and unchanged over time. Reducing hands-on projects to being simple and unchanging is not realistic or very helpful to the learning process.
- Software development, according to contemporary thought leaders, is an eminently practical and hands-on activity, which both demands and lends itself to a pedagogical approach that is practical and hands on, such as a project-based approach. Theory should be taught *in situ* with the practical application of the theory to ensure its relevance, and an understanding of its relevance, to the system development task.
- The availability and use of development tools has increased so enormously over the last 30 years that extensive, and intensive, practical experience in their use must be an essential part of the development education experience.
- The style of software being developed today, which has a significant element of creative design rather than just technical knowledge and ability, demands a different curriculum and learning approach.
- Contemporary (but now not so new) development approaches generally describable as 'lightweight' are much more appropriate to contemporary software development methodological needs, and are very well suited to a hands-on, project-based approach to teaching and learning. These same development approaches are also applicable to teaching and subject administration.
- These same lightweight approaches are much more reflective of the integrated, holistic development style wherein the previously separate and bounded 'phases' of development; analysis, design, programming, testing, are now seen as being done together as part of the normal way of approaching a development task.
- In the educational environment, assessment is obviously necessary to judge the student's knowledge gain for the purposes of mentoring and revising subject matter to assist the student, as well as demonstrating the student's knowledge gain and ability. Thus, a program of continuous assessment within the project-based approach seems necessary and preferable.
- The relevance and usefulness of the subject matter being taught needs to be demonstrated '*in situ*' and reinforced by immediate application. The problem of students 'forgetting' previously taught theory could be overcome by the deep learning characteristics of a practical, hands-on approach where the theory is immediately applied and the student must be able to demonstrate a successful, practical, outcome. An enormous amount of 'learning' is wasted when supposedly pre-requisite knowledge must be taught again because the students' 'crammed' it for the purpose of an exam, and it is consequently almost immediately forgotten. Similarly, when theory taught in one subject by a specific teacher is left to fade away

by not being practically applied by other teachers in other subjects, this implies either the theory is irrelevant, so it was a waste of time to teach it, or the theory is not being applied resulting in the students not comprehending the usefulness of it or just quietly forgetting about it. Either way, it implies a waste of time; time which is necessarily limited in any course. Perhaps 'just in time' learning is a principle to be followed.

- Overcoming the growing knowledge gap that often develops over the course of a semester, where the knowledge elaborated in a lesson is not fully understood or learned before the next lesson. This results in an ever-growing knowledge gap over the duration of the semester. This sequence of 'non-learning' continues over the course of the semester with an ever increasing knowledge gap which drives students to cheat and copy; not appropriate learning strategies, with a result of either failing hopelessly in final exams, or intensive pre-exam study which may be just enough to pass the exam, but is not deep learning by any measure, and is then forgotten almost immediately after the final exam. What teacher has never seen the situation where students have gained excellent assessment results for 'homework' done during semester, yet the student fails the end of term exam miserably?
- Problems of student boredom with passive learning in 'chalk and talk' lectures, and the failure to connect theory with practice. Too often students are taught theory, or just taught on somewhat obscure matters, and are basically told 'trust me, you will understand why this is relevant at some future time in your course'. Of course, as discussed elsewhere, this subject matter may not in fact ever be visited again anyway. There must be an immediacy of application of theory for the theory to be understood as relevant. Equally, there must be an immediacy of application to demonstrate that the theory is indeed relevant. If that cannot be demonstrated, then perhaps the theory is irrelevant and should be removed from the curriculum. Equally, practical hands-on experience can feed into the understanding of the theory where the perhaps problematic experience of the student can be seen as an exemplar of the need for the theory, where students see the relevance of the theory *post-hoc*.
- Especially in software system development, problem-solving skills are essential. It has often been suggested that to be a good computer programmer you need to be good at maths. However, whatever the truth or otherwise of this proposition, the more important skills needed are logical thinking and problem solving skills. Anyone who has ever approached the development of an algorithm and subsequent coding where this is needed to solve a particular problem knows this. Passive learning of, say, Systems Analysis methods does not teach these skills and abilities. The best that such methods can do is provide a checklist of things to do, but do not provide much guidance beyond that. Therefore, requiring students to approach a problem, analyse it and provide a coded solution does provide an experiential approach, calling on the students' abilities, or emphasising to the student the necessity for these abilities.
- Professional responsibility of the students for their own learning outcomes, and a future of life-long learning must be inculcated in the students as part of their preparation for a career in this field. This implies concepts of self-directed learning. Also, students learn on an 'as needed' basis where they must search out a solution to each problem encountered. Enough problems to be solved lead to enough learning to be done in a focused, self-directed manner.

5 The proposed pedagogical approach

First, some definitions: project-based learning has been variously defined, but the most straightforward definition is in Patton and Robin (2012);

> "Project-based learning refers to students designing, planning, and carrying out an extended project that produces a publicly exhibited output such as a product, publication or presentation."

Elsewhere (Jones et al., 1997),

> "Project-based Learning (PBL) is a model that organises learning around projects. According to the definitions found in PBL handbooks for teachers, projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; give students the opportunity to work relatively autonomously over extended periods of time; and culminate in realistic products or presentations."

It is suggested that the proposed 'super' project concept encapsulates all of these aspects of PBL.

The proposed pedagogical approach is to amalgamate a number of subjects in a Business Computing or Information Systems course, where teaching systems development is the main intention. Separate subjects, that is semester-based subjects, which teach systems analysis, design, database design, programming and software project management would be integrated into a single, multi-semester 'super' subject; perhaps incorporating two thirds of the total teaching time of the course. In this 'super' subject a course-wide project of significant complexity and variety would be the major teaching and learning vehicle. The teaching and assessment strategies would be a project-based (including problem-based) approach, with continuous assessment and obvious application of student-centred learning, and student-driven learning strategies. Students would undertake the development of the system in groups; perhaps four or six students per group. Teamwork is an important lesson to be learned; it is not an intuitive ability in most people.

A contemporary 'agile' development approach would be used, meaning an on-going iterative approach, allowing changes and updates to requirements as a matter of course.

6 Example project

There are many possible projects which could be developed, limited only by the imagination, and knowledge, of the teachers in charge. Such as system would obviously incorporate a database with appropriate data maintenance and secure data access and reporting. All aspects of data design, SQL and information extraction and presentation, with interfaces to a word processing package and a spreadsheet would be incorporated. Interfaces with email for wide-band transmission of emails as well as specifically addressed emails would be developed. At all times the program code to achieve each processing activity would be taught and the code developed to a useable and tested state.

All aspects of data storage, locally, on a network, and in the 'cloud' would be developed, and data archiving and backup would be achieved, with good knowledge and understanding of data security.

Aspects of 'mobile data' and 'data anywhere' would be part of the system. Data security and system access security would obviously be a significant topic. This is probably one area that deserves an independent subject in its own right; 'system and network security', including counter-hacking, defending against worms, trojans, DOS attacks, viruses and so on.

As the project proceeds, new and additional requirements can be stated. For example, if the context of the project is a large university, developing a University Administration System, student timetables would be part of the system, showing class times and locations, hyperlink interfacing with a mapping app to show class locations on a campus map. Also, a smartphone app to show available accommodation on and near the campus could be included in the system, and perhaps a smartphone app to show the whereabouts of the campus shuttle buses, and the probable arrival time at a particular bus stop. There are plenty of possible useful system ideas that could be incorporated at appropriate times during the project.

At all times appropriate software development tools would be used. The software marketplace is rich with development tools that enhance productivity or allow simple management of resources.

For those who may feel that such a project is too large for undergraduate students to complete, it is suggested that completing the project is neither especially important, nor is it required to reflect 'real' world situations where many projects have a required finish date. Completing the project has no especial educational value. The educational value has been inherent in the project right from the start, and at every step of the process, by way of the hands on, practical, highly iterative approach and continuous assessment. As for complexity, it must be understood that the development will achieve many small learning goals over the duration of the project, and each small goal would not be overwhelmingly complex.

7 Student centred learning

The term 'teaching and learning' should always be used in decision making about curriculum, pedagogical style and course structures for the simple reason that both teachers and students must be considered. Student centred learning is about perceiving the educational situation from the student side, and about how the teachers should assess their teaching styles and knowledge acquisition outcomes. Following is discussion from the students' point of view.

7.1 Student learning outcomes

The author has previously supervised student capstone projects in the final year of their courses. These projects were industry based with the intention of producing a successful system to meet the requirements of the industry client. Over the period of two years, with the involvement of more than 150 students working in groups of four, the students' learning outcomes were evaluated and assessed (Morien, 2004a, 2004b, 2005b, 2006b). The students were encouraged to use the development method previously discussed, that is, an agile, iterative approach. The students' learning outcomes were a good understanding of systems analysis, and deep learning of database schema and data structure, interface design, and programming skill. Of particular importance was that the

students' interest was held with a sense of achievement being felt throughout the project, and all the aspects of systems development were learned; team work, tool use, project management, software development economics, professional responsibility to the users, and so on.

It can be confidently stated that the breadth and depth of learning experienced by the students was significant, and certainly much more substantial than if standard classroom 'chalk and talk' teaching was used, or even when small, mini-projects based in individual subjects were undertaken.

It is suggested that this method, while being clearly database system development oriented, has both theoretical and practical relevance in most other types of system, including internet-based systems. It should not be interpreted too narrowly in its applicability.

It is also suggested that the significant success observed in the 2-semester capstone projects referred to previously would be achieved and more in a well-managed 'super' project, as is being suggested in this paper.

7.2 Student assessment

In a continuous assessment situation, combined with an iterative development process, students were required to prepare documentation including a weekly personal activity plan and a personal diary of activity. The planned activity could be compared against the actual activity diarised, and the actual activity could be compared against the demonstrable output; code, screens, reports, documentation, tool usage ability. A cycle of "this is what I planned to do leads to this is what I actually did supported by this evidence" was established and continually applied. In this manner, each student could be assessed individually rather than being able to hide behind other members of the group. As well, there is a significant implication of continuous encouragement in the continuous assessment situation, provided such assessment is not obviously made for punitive purposes. If the assessment activity is perceived by students to be positive feedback on their efforts, allowing appropriate mentoring as well as recognising good work done, then this adds to the students' positive perception and desire to learn. A significant aspect of PBL is that the students are building or creating something that will be displayed or demonstrated to interested others, and this gives incentive to succeed and produce something that can be shown with pride.

7.3 Student interest and enthusiasm

The experience of the author in managing substantial student projects in the past clearly indicates that students demonstrate significantly more interest in their own learning and learning activities, and are much more enthusiastic in their application to the learning task when they are practically involved in a hands-on manner. It has been clearly demonstrated that students learn more deeply, are more willing to seek out solutions to their problems by themselves, demonstrate a wider scope of information seeking and are willing to extend their learning environment well beyond the classroom, especially now having the internet as an information source of huge richness and variation.

This was observed to the point that many students continued to work over the long Summer break, from November to March, even though there were no classes, and usually

no academic staff available, indicating the interest that was engendered by participating in such a project.

8 Barriers to implementation

Like any good idea, there will be barriers to implementation. In the education process there are four matters to consider;

- curriculum content
- teaching process and teachability
- learning process and learnability
- assessment process.

Suggestions from the literature on PBL under the heading of 'teachers enactment problems', termed teaching process and teachability here, include:

Time: Projects often take longer than anticipated, and there are often time constraints imposed by official rules, course structures and curriculum content. This may lead teachers to make the projects sufficiently trivial to fit into the available timeframe. In the 'super' project concept, Time is not an issue inasmuch as, while there is clearly an end time, there is no requirement for the students to 'finish' their project. Their ongoing, continuously assessed learning is the relevant matter. In any case, given that this is a learning experience, it is difficult to see what the finished system would look like, so achieving that is not the issue.

Classroom management: teachers may have difficulty balancing the need for classroom discipline and good order against the usual and indeed necessary 'confusion' and 'chaos' of the creative environment necessary for students to work productively and creatively. As such, many teachers may find it difficult to work in such an environment, and may well oppose it. However, it has been seen that the level of interest and enthusiasm maintained by the students doing the project diminished the problem of classroom discipline substantially. It is the teachers who may have this problem, not the students.

Supporting student learning: teachers may not have the knowledge, experience or mindset to provide appropriate scaffolding of student activities, and erring either on the side of too much independence or alternatively restricting independence of students' activities.

Technology: even in a study area such as business computing and other computer related courses, teachers may not have the knowledge of current technology availability nor expertise in use of the technology.

Assessment: teachers may have difficulty in designing appropriate assessment methods that allow students to demonstrate their skills. It is suggested, however, that the primary assessment artefact envisaged in a computer system development project is the delivered project itself; does it satisfy the requirements? Does it have bugs and incorrect structures? Is the visual interface developed according to well understood Human–Computer Interface standards? And so on.

An important problem of assessment arises when teams or groups of students are being assessed. Should each student be individually assessed, or the team as a whole? How does each student have a grade applied? Is the assessment fair to all members of the team? For example, one member of the team may have done practically nothing, but the rest of the team have achieved an excellent project outcome. Should the inactive student be given the same high mark as the achievers, or should the achievers be penalised by receiving a lower team mark due to the poor work of one member?

One assessment method that has been used was to give the team a total mark which could be divided up amongst the team members, by the team members themselves. This is peer assessment at its most liberal. To illustrate, a team of 4 may be given a 'project mark' of, say, 280, out of a total of 400, which means the project is assessed as earning 70%. The students may decide to distribute that 280 as 80, 75, 65 and 60, rewarding the best student and acknowledging the lower participation of the poorest student.

However, requiring the students to maintain a personal learning diary, and the project group as a whole maintaining a project planning and activity diary, with regular evaluations of progress, overcomes the assessment problem substantially.

Homogeneity of learning: teachers are often concerned that in a PBL approach, where students are not following a strictly stated and adhered to curriculum, not all students will learn the same things. Whether or not this is a problem depends very much on formal assessment and examination regimes where all the students are examined and assessed on their knowledge of a specific set of learning outcomes. Apart from that, it is suggested that this would be more apparent than real, if a reasonable regime of teamwork is applied.

Teaching teams and teamwork: One major aspect of academic life, certainly in this author's experience, is the curriculum/teaching dichotomy. That is, to put it in colloquial terms, to have all participating academics 'on the same page' or 'singing from the same song sheet' agreeing to the curriculum content, and agreeing to participate in the project. The fact is most academics have their own often treasured expertise and ideas on curriculum content. There are those, for example, who strongly favour a traditional SDLC analysis and design methodology, while others are more agile oriented. There are those for whom object-oriented practices are what their reputation and academic status rely upon, whereas others are indifferent to this, and strongly prefer different styles of analysis, design and programming. As discussed in Morien (2005c, 2006a), database development approaches and curriculum content vary substantially and confusingly when different teachers approach the topic, and in different textbooks. There are a myriad of matters that could be taught, and some are of great importance to some teachers, while other teachers are either indifferent to them, or are actually hostile to teaching them, or just see other things as being more important. It has even been suggested that there may be significant conflict between those teachers who support the 'new' instructional methods, and the opposition of those with deep-seated beliefs from their prior experience. This has, in other contexts, been termed 'the baggage of experience' where prior experience overwhelms and refuses new thinking.

Clearly, a coherent and cohesive team approach to teaching is demanded for the course structure and practice proposed in this paper to be workable, viable and successful. One wonders if this is actually possible.

Like many aspects of 'requirements determination' in a system development activity, every teacher will have slightly different, even substantially different, curriculum

requirements. The problem will be to accommodate sufficient of those requirements in a coherent fashion to both satisfy the various teachers and ensure appropriate and sufficiently comprehensive curriculum. It is suggested, however, that in many respects curriculum will be self-determining, inasmuch as the development of a complex computer system demands certain skills and abilities. Learning requirements reveal themselves as new requirements are confronted, and new development problems are encountered. It may be reasonably suggested that if a particular matter is never encountered during a development project of a substantial but typical system, of significant complexity, over an extended period of time, then understanding about that matter is irrelevant and addressing it in the curriculum would be a waste of time.

Enabling student learning responsibility: It is neither fair nor reasonable to expect freshman students to be intuitively able to take responsibility for their own learning, and to readily take up the challenge. In all probability their prior education experience in middle and upper school has been very teacher-driven, based on rote learning in a classroom setting, with a fixed curriculum stated and controlled by the teacher. In this case, considerable effort would be needed to introduce students to the new pedagogy, giving them permission to think, giving them licence to act independently, and helping them become self-sufficient and self-directed, and be able to assess their own efforts appropriately.

From the student perspective, being confronted with 'a problem' that may be, or in fact should be, ill defined, fuzzy in its content and boundaries, will apply themselves to the problem in accordance with their learning characteristics. Not all students will exhibit the necessary learning styles for PBL. One study categorised students in being 'challenge seekers' who have a high tolerance for failure, are learning goal oriented. 'Challenge avoiders', however, were more amenable to the traditional instructional style, now termed, somewhat pejoratively by some, as the 'chalk and talk' teaching and learning. However, closely observed student teams working on a computer system development project demonstrated that they were both willing and able to rise to the task, and showed significantly greater enthusiasm and interest than they usually showed in small, assessable projects in particular individual subjects. That is, most students tend towards being 'challenge seekers' (Alausa, 2014; National University of Singapore, 2014). A caveat to this statement is the effect of the prior teaching and learning environment experienced by the students. If they have never been required to act in such an independent and self-motivated way, they will probably be more 'challenge avoiders' than 'challenge seekers'.

It is suggested, as well, that students would have a greater 'knowledge base' available to them if the team teaching approach was extant. The undeniable fact is that no academic can be expert or even proficient in all and every aspect of the information systems universe today; there is just too much to know and too much continuous change. Having a teaching team where all teaching academics are essentially contributing their particular expertise would provide students with a significantly greater 'knowledge base' than if they were reliant on the single 'lecturer-in-charge' of the particular subject. Also, the concept of the teaching team can involve teaching academics in a learning role as well, thus giving them in-house knowledge gain by learning from their fellow academics in the teaching team situation.

9 Conclusion

Software systems development is an activity requiring an integrated, holistic approach, and system development education needs to reflect this by providing students with hands-on development experience, deep learning at every level, with 'real-world' experience as an integral concern. Separation of a number of aspects of development, analysis, design, database, programming, into bounded, dissociated subjects is inefficient, ineffective and wastes time, effort and, worst of all, wastes opportunities for learning. Current education practices in Business Computing and Information Systems fail in these aspects.

The proposition in this paper is, therefore, that Business Computing and Information Systems education should be based on a substantial project of an industrially realistic size and complexity that is used as the primary learning vehicle over a number of years of the course, adopting an educational strategy based on an amalgam of PBL, problem-based learning, continuous assessment, student centred learning and student driven learning. It is acknowledged, however, that implementing this is not without its problems.

References

Alausa, Y.A. (2014) *Continuous Assessment in our Schools: Advantages and Problems*, http://www.nied.edu.na/publications/journals/journal9/Journal%209%20Article%202.pdf (Accessed 1 October, 2014).

Blumenfeld, P.C., Soloway, E., Marx, R.W., Krajcik, J.S., Guzdial, M. and Palincsar, A. (2014) 'Motivating project-based learning: sustaining the doing, supporting the learning', *Educational Psychologist*, Vol. 26, Nos. 3–4, pp.369–398 (Accessed 1 October, 2014).

Budde, R., Kuhlenkamp, K., Mathiassen, L. and Zullighoven, H. (Eds.) (1984) *Approaches to Prototyping*, Springer-Verlag, Berlin, Heidelberg, New York.

Chikofsky, E.J. (1990) 'The database as a business road map', *Database Programming & Design*, Vol. 3, No. 5, May.

Felder, R. (2014) *Resources in Student Centered Teaching and Learning, Resources in Science and Education*, blog, http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Student-Centered.html (Accessed 1 October, 2014).

Froyd, J. and Simpson, N. (2013) *Student-Centered Learning Addressing Faculty Questions about Student-centered Learning*, Transforming Undergraduate Education in Science, Technology, Engineering and Matehmatics, Washington DC.

Jenkins, A.M. (1985) 'Prototyping: a method for the design and development of applications systems', *Spectrum*, Vol. 2, No. 2, April.

Jones, B.F., Rasmussen, C.M. and Moffitt, M.C. (1997) *Real-life Problem Solving: A Collaborative approach to Interdisciplinary Learning*, American Psychological Association, Washington DC.

McConnell, S. (1996) *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, Redmond, WA.

Morien, R.I. (1992) 'Prototyping large on-line systems: using a concept of a focal entity for task identification', *Proceedings of the Third Australian Conference on Information Systems*, 5–8 October, Wollongong, Australia.

Morien, R.I. (2004a) 'Insights into using agile development methods in student final year projects', *Proceedings InSITE2004*, 26–28 June, Central Queensland University, Rockhampton, Australia.

Morien, R.I. (2004b) 'Insights into using agile development methods in student final year projects', *International Journal of Issues in Informing Science and Information Technology*, Vol. 1, pp.605–624, Informing Science Institute, www.informingscience.org

Morien, R.I. (2005a) 'Agile development of the database: a focal entity prototyping approach', *Agile2005*, Denver, Colorado, pp.103–110.

Morien, R.I. (2005b) 'Student experience of using agile development methods in industrial experience projects', *Proceedings ISECON 2005*, Vol. 22, Columbus, Ohio.

Morien, R.I. (2005c) 'A critical evaluation of database textbooks, curriculum and educational outcomes', *Proceedings ISECON 2005*, Vol. 22, Columbus, Ohio.

Morien, R.I. (2006a) 'A critical evaluation of database textbooks, curriculum and educational outcomes', *Information Systems Education Journal*, Vol. 4, No. 44, 24 October, 2006, http://isedj.org/4/44/

Morien R.I. (2006b) 'Student experience of using agile development methods in industrial experience projects', *Information Systems Education Journal*, Vol. 4, No. 103, 24 October, 2006, http://isedj.org/4/103/

Morien, R.I. and Schmidenberg, O. (1994) 'Educating information systems professionals: the tertiary educational challenge', *Proceedings of Apitite'94 (Asia Pacific Information Technology in Training and Education)*, June, Brisbane, Australia.

National University of Singapore (2014) *Learning to Teach, Teaching to Learn, A Handbook for NUS Teachers*, http://www.cdtl.nus.edu.sg/handbook/home/foreword.htm (Accessed 1 October, 2014).

Naumann, J.D. and Jenkins, A.M. (1982) 'Prototyping: the new paradigm for systems development', *MIS Quarterly*, Vol. 6, No. 3, September.

Nielsen, L. (2014) *Student Driven Learning = Passion-Based Classrooms*, blog, http://theinnovativeeducator.blogspot.com/2011/04/student-driven-learning-passion-based.html (Accessed 1 October, 2014).

Patton, A. (2014) *Work that Matters: The Teachers Guide to Project Based Learning*, http://www.innovationunit.org/sites/default/files/Teacher'sGuidetoProject-basedLearning.pdf (Accessed 1 October, 2014).

Patton, A. and Robin, J. (2012) *Work that Matters: The Teacher's Guide to Project-based Learning*, http://www.phf.org.uk/wp-content/uploads/2014/10/Teachers-Guide-to-Project-based-Learning.pdf, Published by the Paul Hamlyn Foundation, http://www.phf.org.uk/, (Accessed 23 August, 2016).

Thomas, J.W. (2000) *A Review of Research on Project-Based Learning*, March, 2000, http://www.bie.org/index.php/site/RE/pbl_research/29 (Accessed 10 October, 2014).

Thomas, J.W., Mergendoller, J.R. and Michaelson, A. (1999) *Project-based Learning: A Handbook for Middle and High School Teachers*, The Buck Institute for Education, Novato, CA.