Teaching Software Engineering as an Undergraduate Elective

Musti Sastry
The University of West Indies, Trinidad and Tobago, mks_sastry@ieee.org

Kim Mallalieu
The University of West Indies, Trinidad and Tobago, kim.mallalieu@sta.uwi.edu

Azim Abdool
The University of West Indies, Trinidad and Tobago, azim.abdool@sta.uwi.edu

ABSTRACT

Teaching Software Engineering is quite a different undertaking from that of teaching programming languages. At its best, it facilitates a real world experience for students while at the same time satisfying the academic requirements of a university programme. This paper presents an innovative strategy that has been adopted to teach a software engineering course as an elective to undergraduate students at The University of West Indies (UWI) on the St Augustine campus in Trinidad and Tobago. The strategy incorporates many aspects of a software industry experience within typical classroom and semester constraints. It does so on a modest budget. Course composition, content delivery, hands-on training and assessment criteria are described and feedback on the teaching and student experiences is provided.

Keywords: Engineering Education, Software Engineering, Computer Programming

1. INTRODUCTION

Undergraduate students generally think that Software Engineering (SE) is all about writing computer programs. However in reality, writing programs is only one of the phases of the software production process, which in turn is one of the phases in the SE cycle. Traditional courses which emphasize the theoretical aspects of SE, typically do not capture key elements of the full cycle and often exclude exposure to large software projects, software quality indicators and relevant non technical issues. Yet, time limits of the semester system; experience, skills and professional background of teachers; and the need for extensive laboratory support often present significant challenges to the planning and delivery of more effective approaches. The development and delivery of software engineering courses have attracted a great deal of attention on account of the many pressing challenges in a university environment. Reported models span a range of delivery modes, course content and assessment criteria. Many (Jacob, 2003; Todd, 2001; Hal, 2001) have split the entire class into groups of 3 to 5 to undertake a single large project. Others (Garry, 2005) have used industry as customers to give students a real-life experience. With the benefit of our past experience and that featured in the literature, a novel and effective teaching strategy has been planned and implemented in the delivery of ECNG-3023 *Introduction to Software Engineering*.

2. LEARNING OUTCOMES AND CONTENT DELIVERY

This course is offered as an elective to the students of 3-year B.Sc Program in Electrical and Computer Engineering at The University of West Indies, St Augustine Campus. The course content has been planned to enable both theoretical and real-world exposure to representative SE processes through a coordinated teaching strategy.

The major learning outcomes are centred on

1. Engineering the overall process and life cycle of a large software project
2. Analysis of the software modules to measure complexity and efficiency
3. Design and development of software applications against pre-defined quality standards
4. Maintenance planning and reliability estimation of a developed application.

These learning outcomes are consistent with the overall BSc program's learning outcomes which are based on international standards. Eight different theoretical modules with relevant hands-on tutorials have been administered over a 13 week academic semester. At present, there is no pre-requisite for this taking up this course, due to the fact that all students of Electrical and Computer Engineering would have completed one or two mandatory computer programming courses offered by the department. These mandatory courses are offered at level 1 and level 2 and impart skills of program development, implementation and the related problem solving. However, it is reTable 1 shows the topics for lectures and tutorials.

Table 1: Topics for Lectures and Tutorials

Major topics for Course Lectures	Tutorial Topics
• Introduction to Software Engineering • Modeling the Process and Life Cycle • Capturing the Requirements • Designing the System • Software Metrics • Program Development • Quality Assurance and Testing the Programs • Testing the Systems	• Introduction to Microsoft Visual C++ .Net development environment • Using the Microsoft VC++ .Net to develop console applications • Practical Exercise on Requirement analysis and System design/ Group Work • Introduction to MS Windows Forms • Programming with MS Windows Forms

The classroom lectures are delivered over 3 hours every week, which are supplemented with a 1 hour tutorial. Faculty of Engineering, UWI has an academic alliance with Microsoft Corporation, USA; through which students can access the latest SW developmental technologies for application development. MS Visual Studio .NET 2008 environment is used for tutorials and course work. This is a very recent and state-of-art C++ development platform from Microsoft, based on .NET framework technologies. Students are trained on using the environment and instructed to develop the applications given as part of the coursework. The core SE fundamentals with examples have been taught over 12 lectures. Initial lectures focused on importance of requirement analysis, overall design and adopting a viable process life cycle. More emphasis is kept on SW metrics and testing processes to give students a feel of the application complexity and importance of quality. The tutorials are focused on hands on experience with analysis, design and programming with real life examples (Sastry, 2007a; Sastry, 2007b) like design and development of large information systems.

3. ASSESSMENT PROCESS

To test the theoretical and practical knowledge acquired, a typical assessment procedure is adopted. Three course work examinations and a final examination are used to test the student's learning. Table 2 provides the composition of the assessment strategy used for evaluating the students.

Table 2: Assessment Strategy

ASSESSMENT COMPONENT	DETAILS
Course Work 1 (10% weighting)	Take Home type, Time to Solve: One week, Problem solving using C++ Structures
Course Work 2 (10% weighting)	Take Home type, Time to Solve: One week, Development of C++ Classes/libraries for specific applications
Course Work 3 (20% weighting)	Take Home type, Time to Solve: Two weeks, Design and Development of User Interface using Windows Forms
Final examination	3 Hour Examination, Closed book type, 60% weighting

4. EXPERIENCING SE PROCESSES THROUGH COURSE WORK

In this offering, 20 students have registered for the course. These students have been divided into 10 groups with 2 students per group. Ten different problems of more or less same level of difficulty have been prepared for the students to solve as part of the coursework. Table 3 lists these ten problems, with the salient features expected to be met by the students. These non-engineering type problems are mostly specific to Trinidad and Tobago and so identified to avoid plagiarism. It is a normal practice among the students to look for help over the internet for regular text book problems or the problems with standard algorithms. Several websites provide problem statement, solution design and even the complete code with comments for the most common and popular problems. Students will try hard to find the solutions, related information from the internet using popular search engines, if the course work is based on well known problems, algorithms or so called stereo type problems.

It can be observed these problems are not very common or popular and there are no ready made solutions available on the internet. Every two students get the same problem and each of them should work independently on the problem in all phases. In first course work, students develop requirement analysis and design. It is easy to identify if the students actually colluded with each other, as there are only two students in a given group. On the other hand, students are openly encouraged to consult each other on application development, documentation standards, code implementation and using the development environment. This strategy of assigning individual problems is opposed to the well known, classical method of assigning a large, single problem to the entire class. Students had to work on their problems individually to make the documentation and then develop the applications based on their own design. Only short description on each course work problem is provided and this is deliberate. This approach provides students to think without limits and come up with their original ideas, specifications and features.

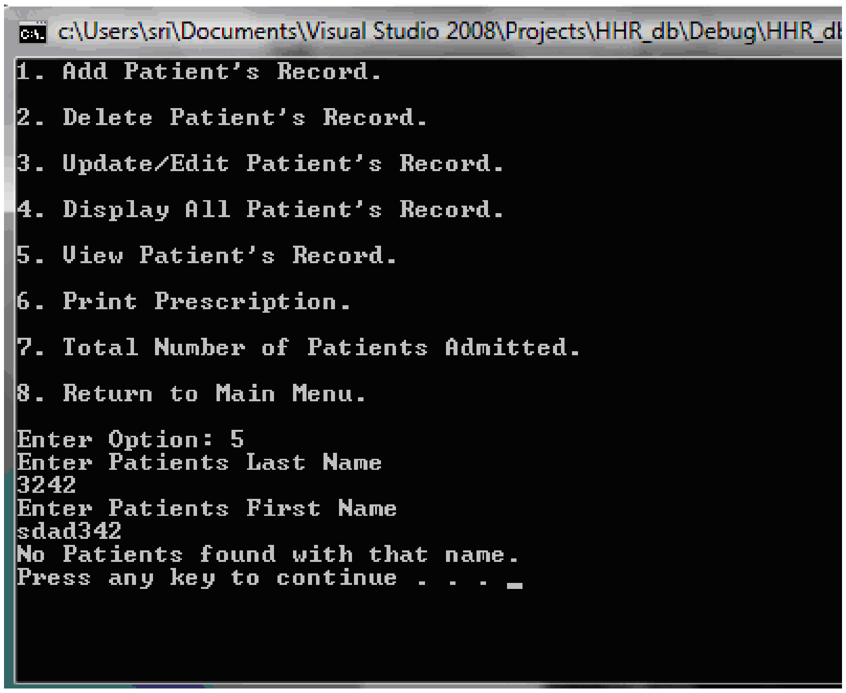
5. STUDENT PARTICIPATION

The attention and participation of students registered for this course is very good as evident from their response in the classroom. There is indeed a strong resistance for individual course work problems, as they were habituated to get away with a single problem for entire class. However, after explaining the benefits of having individual problems such as freedom to individuals, no scope for resorting to plagiarism and working knowledge on state-of-art Microsoft technologies; students agreed to work on independent problems. Coursework submission guidelines are clearly defined (IEEE Std 1074-1997; IEEE Std 1233, 1998) and handed to students well before the deadlines. For instance, first course work submission should have five major components viz., a) Problem Statement and definition b)Goals and objectives c) SW Requirement Specification d) SW life cycle model e) SSD

Quality of course work submission is also found to be good. Figure 1 shows a console application developed as part of second course work and Figure 3 shows a windows based graphical user interface as part of third course work.

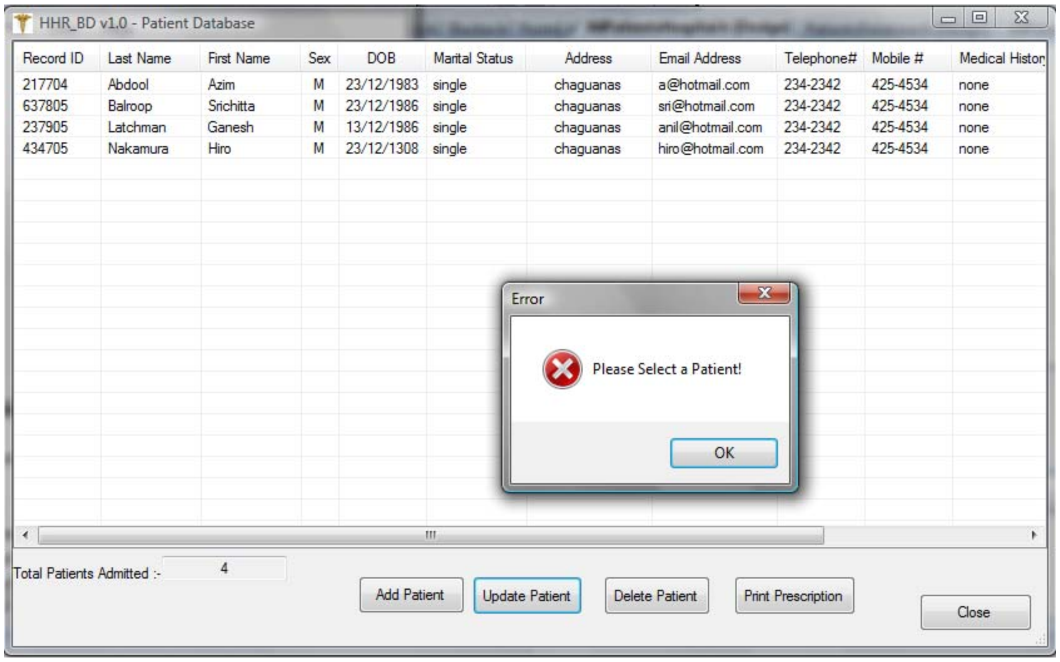
Table 3: Course Work Problems with short description

1. Port-of-Spain to Tobago Sea bridge Features - Schedule Management - Ticket enquiry/ sales/ printing for travelers - Ticket enquiry/ sales/ printing for vehicles - Luggage management/ check-in/ tracking - User / Operator Management - Data archival and information retrieval	6. Customer Management for a Telephone Company Features - Customer management - Call history - Billing management - Faults and Complaints - Outlets for bill payments and Top-ops - Data entry screens for Operators - Data archival and information retrieval
2. Port-of-Spain to Tobago Air bridge - Schedule Management - Ticket enquiry/ sales/ printing for travelers - Seat display and selection - Luggage management/ check-in/ tracking - User / Operator Management - Data archival and information retrieval	7. Automation of a Laptop selling shop - Laptop information system - Stock inventory - Sales management - Accessories information and management - Customer management - Data archival and information retrieval
3. Software for Cricket Matches - Data entry screen for operator – ball wise event input - Score card preparation - Batsmen profiles and summary - Bowler profiles and summary - Umpire profiles - Data archival and information retrieval	8. Automation of a CD/DVD rental shop - CD/ DVD Library information system - CD/DVD player renting - TV / large screen renting - Lending management - Customer management - Data archival and information retrieval
4. Public Health Facility - Organization management [Director/ doctors/ nurses and lab technicians] - Patient Management - Data entry screens for operator - Prescription printing - Data archival and information retrieval	9. Automation of Driving Licensing and Vehicle Registration - License management system - Vehicle Information system - Owner and buyer information - Fee management system - Officer management system - Data archival and information retrieval
5. Drug Store Management - Distributor management - Drug details and management - Drug inventory with expiry dates of drugs - Sales management/Receipt printing - Data entry screens for operator - Information on subsidized and free drugs	10. Automation of Cell phone selling shop - Cell phone information system - Stock inventory - Accessories information and management - Sales management - Customer management - Data archival and information retrieval



```
c:\Users\sri\Documents\Visual Studio 2008\Projects\HHR_db\Debug\HHR_dl
1. Add Patient's Record.

2. Delete Patient's Record.

3. Update/Edit Patient's Record.

4. Display All Patient's Record.

5. View Patient's Record.

6. Print Prescription.

7. Total Number of Patients Admitted.

8. Return to Main Menu.

Enter Option: 5
Enter Patients Last Name
3242
Enter Patients First Name
sdad342
No Patients found with that name.
Press any key to continue . . .
```

Figure 1: Console application for Hospital Information management System



HHR_BD v1.0 - Patient Database

Record ID	Last Name	First Name	Sex	DOB	Marital Status	Address	Email Address	Telephone#	Mobile #	Medical Histor
217704	Abdool	Azim	M	23/12/1983	single	chaguanas	a@hotmail.com	234-2342	425-4534	none
637805	Balroop	Srichitta	M	23/12/1986	single	chaguanas	sri@hotmail.com	234-2342	425-4534	none
237905	Latchman	Ganesh	M	13/12/1986	single	chaguanas	anil@hotmail.com	234-2342	425-4534	none
434705	Nakamura	Hiro	M	23/12/1308	single	chaguanas	hiro@hotmail.com	234-2342	425-4534	none

Error

Please Select a Patient!

Total Patients Admitted :- 4

Figure 2: GUI for Hospital Information management System

This new approach actually provided a new avenue for bringing students to the books, thoughts so that they will create their own solutions. At the end, students have realized that SE processes indeed are complex and require continuous practice and experience to perfect them.

6. OVERALL STUDENT PERFORMANCE AND FEEDBACK

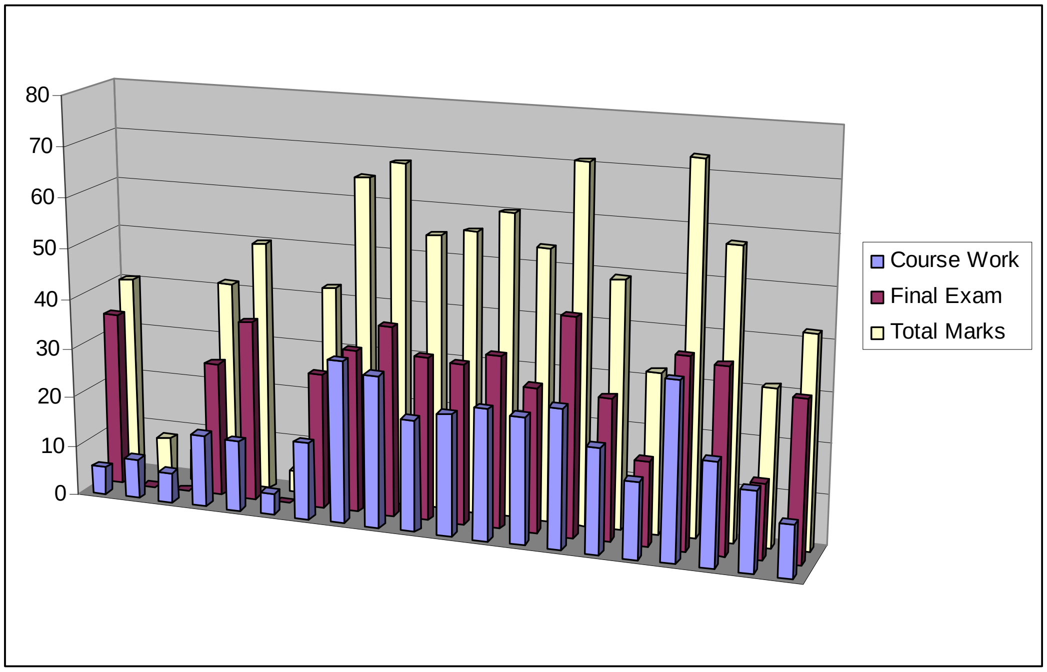


Figure 3: Student Performance in coursework and final examination

Students reacted positively over the course contents, lecture material and tutorials. The strategy of three stage course work leading to a windows based graphical interface, which is unique to each student paid rich dividends at the end. When students completed the final coursework, they felt the experience of completing one complete project cycle involving various SE processes from specifications to the end product. Though feelings of the students varied on the nature of the coursework, most felt that they should have worked more to gain programming experience in latest visual technologies. The overall student performance with coursework and final examination components is shown in figure 3. Out of the twenty students, three students failed. Two out of the three failed students did not do well in the coursework and also were absent for the final examination. The overall minimum mark is 4.3 and the maximum mark is 73 out of 100. The average mark is 45 and the standard deviation of the class distribution is 21.

7. CONCLUSION

A new teaching strategy for SE course has been presented. This combines the classical classroom teaching with a three stage coursework on latest programming technologies. From the student reaction and feedback it is evident that this strategy has yielded good results. Details of planning, course work description, content delivery, programming platforms and feedback are illustrated. It is hoped that this can help others as guide for replication elsewhere.

REFERENCES

Gary Pollice (2005), "Teaching software development vs. software engineering",

15 Dec 2005, www.ibm.com/developerworks/rational/library/dec05/pollice/index.html, last accessed December 2008

Hal Abelson, Philip Greenspun (2001), “Teaching Software Engineering-lessons from MIT”, *Tenth International World Wide Web Conference (Hong Kong)*, May 1-5, 2001, http://philip.greenspun.com/teaching/teaching-software-engineering, last accessed December 2008

Jacob Sukhodolsky, (2003), “Teaching Software Engineering To Undergraduates”, *International Conference on Information Systems and Engineering (ISE 2003)*, Quebec, Canada, pp 165-173

Keren. G (2001), “Why do Universities Fail Teaching Software Engineering?” http://users.actcom.co.il/~choo/lupg/essays/software-engineering-and-uni.html, last accessed December 2008

Pfleeger, Shari Lawrence (1998), “Software Engineering: Theory and Practice, Prentice Hall, Upper Saddle River, NJ.

Sastry MKS, “Development of Academic Information System for Effective Education Management”, International Journal of Management in Education, Int. J. Management in Education, Vol. 1, No. 3, 2007, pp.276–286. (Web link: http://www.inderscience.com/IJMIE)

Sastry, M.K.S., “Integrated Outage Management System: an Effective Solution for Power Utilities to Address Customer Grievances,” International Journal of Electronic Customer Management, Vol. 1, No.1, 2007, pp. 30 – 40, (Web link: http://www.inderscience.com/IJECRM)

Todd Stevens (2001), “Experiences Teaching Software Engineering for the First Time”, *6th annual conference on Innovation and technology in computer science education*, Canterbury, United Kingdom, pp.77 - 80

IEEE Standards for Software Engineering

IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements

IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions.

Authorization and Disclaimer

Index Terms â€“ Undergraduate engineering education, context, content, pedagogy, paradigm shift. INTRODUCTION. This paper advocates an approach to undergraduate engineering education based on a new paradigm in which engineering is practiced with social as well as technical expertise.Â As long as the world is perceived merely as a machine and the engineer as a mechanic, then one can successfully interact with the world in a linear way. However, when we begin to recognize that the world is alive and that we are part of it, linearity is no longer appropriate.Â The traditional approach to teaching engineering systems and network theory relies on an understanding of linearity and reductionism.