

*A good balance of theory and practice in any curriculum not only serves to re-enforce the students' understanding of abstract concepts, but also provides them with insight and appreciation of the practical solutions at hand. In this issue of CDTL Brief on **Balancing Theory and Practice**, the authors discuss how practice can be integrated into the respective courses to enhance students' understanding of the principles and concepts students learn in the classroom.*

Combining Theory and Practice in the Right Proportions

Associate Professor Seah Kar Heng
Department of Mechanical Engineering

Introduction

For many years, the local public has had the notion that university engineering graduates are strong in theory because that is what we teach here. Polytechnic engineering students, however, are practice oriented because they are given more opportunities to use their hands in the classroom. When the NTU engineering schools were set up in 1984 the press emphasised that their graduates are better at hands-on engineering whereas NUS engineering graduates shun practical work because they are trained to do research. The question then is: Can an engineer function properly as an armchair expert? Hercule Poirot, the famous Belgian detective in Agatha Christie's murder mysteries, is a classic example of an armchair detective who solves mysteries by simply sitting and thinking in his armchair instead of combing for physical evidence. This surely is an extreme case of employing theory to the exclusion of all else. It might work in fiction, but certainly not in real life as I shall attempt to explain in this paper.

The role of theory

Theory is essential for a thorough understanding of the principles of any subject of study, be it medicine, law or engineering. There has to be a set of rules upon which premises and assumptions are made, failing which any project one undertakes will merely be 'a shot in the dark' and no one can be sure whether it will work. For example, when one sets out to build a bridge, certain theoretical assumptions which have been proven reliable through the generations, need to be made. Theory is hence the building blocks on which practice is based.

The role of practice

Theory without practice does not get one anywhere. It is easy to predict how a part of a machine will work by calculation, but it may not work as expected because of factors beyond one's control such as frictional force, changes in ambient temperature or inhomogeneous material properties. With some practical experience, an engineer would be able to assess the situation more accurately and get closer to the truth.

Combining theory and practice

The ideal method of teaching balances theory and practice in correct proportions. However, the 'correct proportion' varies from subject to subject. Most courses in NUS are four-year courses. With rapid advancements in science and technology, more specialised modules are being spawned by new technology in the industry. This means more essential theory to cover within the four years of classroom education, making it increasingly difficult for tertiary education curriculum planners to tailor the engineering curriculum to equip students for industry. Furthermore, heeding Prime Minister Lee Hsien Loong's advice to "teach less...so that our children can learn more"¹ we are now left with even less time to cover all the topics students ought to know before entering the working world.

Still, we need to allocate some time to practical training. In NUS, medical students go for clinical postings

1. Prime Minister Lee Hsien Loong's National Day Rally 2004 Speech, "Our Future of Opportunity and Promise". Sunday 22 August 2004, at the University Cultural Centre, NUS. <http://app.sprinter.gov.sg/data/pr/2004083101.htm> (Last accessed: 4 July 2005).

starting from the third year and they spend one year doing housemanship before becoming full-fledged doctors. In the Faculty of Engineering, we used to make it compulsory for all our students to do a 6-month industrial attachment. This has since been made optional because many students complained that they were wasting time doing menial tasks in the companies. However, the 6-month industrial attachment scheme is still compulsory in NTU. On top of that, NTU students need to do an in-house practical training stint during the long vacation at the end of their second year. Although we do not have such compulsory vacation training in NUS, we make up for it with a full year of rigorous training in mechanical engineering design during the third year of the course. In the first semester, ME3101 “Mechanical Engineering Design Part 1” requires students to do a paper design of their projects. In the second semester, students have to fabricate their paper design for the module ME3102 “Mechanical Engineering Design Part 2”. The final year project, which is compulsory for all engineering courses in universities across the world, will be the climax in which every engineering student proves his engineering skills in a project that employs engineering theory in a practical context.

There is something to be said for the rule that one should teach as much theory as possible in a university, since it is probably the last dose of theory the student is ever going to get. Once the student enters the working world, he has the rest of his life to gain practical experience. Unfortunately, many companies do not see the importance of having a good grounding in engineering theory; they prefer to hire someone who

can be put immediately on the front line because he/she has had plenty of practical experience during his university days. While hiring a ready-made worker may be convenient in the short term, the worker may run out of steam in the long run due to a lack of grounding in the fundamental principles.

If a student asks me whether his engineering design will work, I could either explain the fundamentals of engineering to him and show him how it will work, or I could draw from my experience in research and consultancy work and tell him that I have tried it and found it unfeasible. Both methods will satisfy the student, but it is the explanation of fundamentals that will ‘teach’ him how to ‘fish’ for the rest of his life. That is why although practice is essential for being a good engineer, theory should not be compromised at university. The latter should be taught first before putting a student through practical training. The new concept of problem-based learning, if not handled wisely, could end up putting the cart before the horse (i.e. practice before theory).

Conclusion

Balancing practice and theory is easier said than done. It has been a bone of contention for many curriculum planners. The more time a student spends on learning theory, the less time he has for practical work, and vice versa. It is therefore necessary for each faculty and department to work out an optimal solution to produce students that are well balanced in theory and practice, so that they are in a position to continue improving themselves for the rest of their lives. ■

Theory and Practice: Finding the Balance in Media Studies Modules

Assistant Professor Irina Aristarkhova

Communications and New Media Programme & University Scholars Programme

Introduction

There are two main issues I sought to address in teaching IF2210 “Aesthetics of New Media” and UAR2201 “Cyberart”. First, media studies as a discipline is closely related to the practice of media industries, particularly creative media. Second, aesthetics as a discipline is related to the practice of making art, which is a broad term as far as contemporary and new media arts are concerned. Therefore, in both cases, a careful consideration of proportion and interaction between theory and practice is essential to the form and content of teaching.

The paper will present further discussion on:

1. The question of disciplinary balance between theory and practice, or why one cannot avoid the issue of experience and practice in teaching media aesthetics.
2. Practical solutions which I have found useful in empowering students with practical skills to help them understand the complex theoretical concepts taught in my classes, and in turn, re-apply the concepts to their creative projects.

Relevance of practice and media experience to media studies and aesthetics

Media and cyberculture studies rely heavily on developments in information, communication and bio-technologies. Most of the operative concepts like 'Information and Network Society', 'Knowledge-based Economy', 'Interactivity', 'Mediation', 'Human-Machine Interaction', 'E-Learning', 'New Media', 'Post-Human', 'Cyborg' and so on, are founded on technological innovations. These concepts would not have entered the academic curricula without social and cultural values that provide the necessary political, educational and business frameworks. Thus, the link between theoretical frameworks and practical developments is crucial in media studies.

However, traditionally, many academics in humanities and social sciences have theorised on technology without first hand experience. The European situation is relevant here, since it is there where traditional separation between 'hard' and 'soft' sciences has had a long history. Wolfgang Shirmacher, the dean of the European Graduate School, describes how the lack of experience and practice in media and technology among European media scholars and lecturers leads to a situation of widespread disdain for media industries, such as television, games and the Internet. As a result, the young students of media studies do not consider practice and effort in learning technology as a fundamental necessity of theorising media:

Media aesthetics in Europe is preaching in the desert. Teachers and students are used to a talking head, the professor, and an audience taking notes. A discussion after the master's long talk gives future masters the opportunity to deliver their own talk, mimicking the professor's attitude. There is a disregard for the form of the delivery—as long as the sentences pour out of a significant mouth—because the pre-media academics believe strongly in the traditional dichotomy of form and content and the dominance of content over form. In terms of criticism only, the academic philosopher is interested in media and shares his or her biases freely with a like-minded audience. What a judge the person makes who has never enjoyed a video clip, who at best accepts the Beatles, who was never raised as a television baby and never watches TV commercials for the fun of it! (Shirmacher, 1991)

A scholar of new media needs to be in tune with new developments in the media industries and know how to incorporate them into his/her pedagogy. Practice has to be reflected upon, while theory needs to be grounded in actual experience. Among other strategies to help students understand the importance of collaboration between theorists and practitioners, I have made

guest lectures by creative media practitioners such as artists, filmmakers, game developers and experimental musicians, an important supplement to my courses. These guest lectures allow students to learn from and ask questions of a person who is working 'out there' in the real world. In addition, such lectures substantiate theoretical concepts with a practitioner's perspective, making students refer to and reflect on the realm of practice. It is also important that students observe their lecturer to be open to questioning by the exigencies of practical issues and being challenged by the issues of 'making' versus 'theorising'. In my classes, I teach students to value knowledge that is tested and negotiated by the realities of practice by emphasising the importance of relating theory to practical success in innovative approaches to creative media.

Learning about the practical aspects of abstract concepts through critique sessions

Teaching theory is challenging, especially theories of new media aesthetics. The concepts are often abstract and may seem speculative to students who do not have experience with either art or creative media. Using relevant examples can help alleviate the problem of comprehension, but this method has limitations. Being a theorist myself, I tend to include complex theoretical concepts and difficult texts in the curriculum. For example, in teaching the concept of "virtual reality", I ask undergraduate students to read Rene Descartes' *First Meditation* that speculates on the reality of the external world and pose these issues in relation to the aesthetics and ideas of David Cronenberg's film, *eXistenZ*. Then, students are required to make the connection between Descartes' ideas on reality and Cronenberg's film by posting their views on the IVLE forum, conducting independent research on both the text and the film, and presenting their findings in class. The presentation is followed by an in-class discussion that draws out the salient issues.

However, I think students learn mainly from their practical creative projects that constitute the largest proportion of the final grade. "Cyberarts" is a studio-based module where 70% of the students' grade is based on their video-art project (40%) and net-art project (30%). Even in the more traditionally structured lecture/tutorial module "Aesthetics of New Media", the practical component accounts for 50% of the final grade and the examination accounts for only 30% of the final grade.

This signals to students that a practical deliberation of and 'struggling with' the concepts discussed in class to translate the concepts into their self-defined creative projects is crucial. For the "Cyberarts" class, an important part of such 'translation' has been the critique sessions where students showcase their creative 'work-in-progress' while their classmates and

lecturers ‘interrogate’ the project. The questions mainly revolve around how the concept of the project has been researched and applied to the making of the work itself: media specificity, research process, target audience, exhibition style, relevance to the local context, and critical approach to the concept. During the session, suggestions on these aspects are offered and possible further research directions are highlighted.

Informal student feedback has consistently indicated that such critique sessions though painful, are extremely useful not only for developing the projects and testing aesthetic theories, but also for building students’ confidence in what constitutes an effective critique. These learning experiences are transferable to a variety of professional situations including the students’ professional life.

At the end of each semester, I usually ask students if they have been given enough time for their projects, or whether they should be given more theoretical work

instead. Interestingly, students in the “Cyberarts” class from the last academic year (AY 2003/2004) demanded for a larger net-art project in addition to their video-art project. They claimed that doing the project helped them make the abstract interactive media theories ‘their own’ and helped them appreciate conceptual and technological difficulties of new media art making.

The final class in the course is usually an ‘Open Class’—an exhibition that showcases the students’ creative projects to a wider NUS audience and the public. It closes the full circle of theory/practice connection when students learn to communicate their concepts during the question and answer session after each project presentation, and ‘defend’ their ideas and work like in most professional settings.

Reference

Shirmacher, W. (1991/2000). “Media Aesthetics in Europe”. <http://www.egs.edu/faculty/schirmacher/aesth.html> (Last accessed: 28 June 2005). ■

Teaching an Advanced Design, Team-oriented Software Project Course*

Associate Professor Stan Jarzabek & Mr Pin-Kwang Eng

Department of Computer Science

Introduction

Project courses in the Department of Computer Science teach students how to apply principles and concepts learned in the classroom to large-scale team-based projects and fill the gap between theoretical and experiential software engineering knowledge. CS3215 “Software Engineering Project” is such a course focusing on the design and implementation phases of the software development lifecycle. Design principles and teamwork are taught using the problem-based method through architectural concepts and the iterative development process. The course is conducted in such a way that it is impossible for students to achieve the project goals unless they follow the path of ‘best practices’ we recommend to them. In this paper, we will describe the teaching method of the project course, project infrastructure and lessons learned over three years of teaching the course.

Background

Students learn about design principles and ‘best practices’ in many courses. However, small scale assignments do not give students enough opportunities to appreciate the value of software design principles or learn how to apply principles in practice. Further, there

may not be enough opportunities to develop students’ communication and problem solving skills in the frame of small assignments. Not surprisingly, when exposed to real world pressures of industrial projects, students often find it difficult to use the skills and principles learned at the university to their advantage. Instead, students tend to perceive principles as obstacles rather than tools that can help them complete the project. Though industrial attachments offer an invaluable experience, not all companies expose students to best practices or let student teams experience the whole development cycle.

Based on our experiences of experimenting with various approaches to teaching project courses over the years, we introduced a project course CS3215 “Software Engineering Project” that focuses on advanced design by following a rigorous Software Development Life Cycle (SDLC). The ultimate goal of the course is to help fresh graduates transfer good practices to the industry and contribute to industrial projects. The application domain of the course guarantees challenging design problems, emphasises the role of software architecture and component interfaces, and involves complex data structures and algorithms. The problem is selected and scoped in such a way that students cannot meet the

project goals without applying software engineering principles and ‘best practices’ we recommend to them. The course also aims at enhancing students’ communication and problem solving skills.

An overview of CS3215

The project course starts with ten lectures where a chief instructor would motivate students, clarify course objectives as well as explain the programming problem, project methodology and development process. Students do the project in teams of six students that are further divided into two groups of three students. Each team is assigned an one-hour slot per week for consultation with a supervisor. Supervisors (a chief instructor and teaching assistants) are familiar with all technical aspects of the project and share a common vision of the project course objectives.

Students spend two weeks on problem analysis and another two weeks on architectural specifications. At the same time, they develop a throw-away prototype. A programming problem—a software tool called Static Program Analyzer (SPA)—has been carefully selected to allow the two groups within each team to work on the two subsystems in a fairly independent way. As it is virtually impossible to integrate the two subsystems without a proper definition of the interfaces, we explain subsystem-level SPA decomposition to the students, and their task is to follow up with component-level decomposition and specifications of major component interfaces. At the same time, the interface is complex enough so some changes and refinements of interfaces are inevitable during development iterations. To cope with that, students must work together during architecture design and meet regularly during project development. Students also develop communication skills as they learn how to schedule meetings and how to write documentation in a way that other team members can understand.

Students develop the project in iterations. This process helps students tackle the difficulties one by one, applying principles of separation of concerns, abstraction and refinement. Program reliability (close to industry standards and achieved by reviews and comprehensive testing) is emphasised throughout the project. Students are advised to allocate enough time for planning and testing, to make unit testing an integral part of development and to do integration testing and system testing often, at least at the end of each development iteration.

We provide students with a project handbook that includes a problem description, compendium of recommended software engineering practices for the project, sample solutions illustrating how we expect them to approach design problems and technical tips. At the end of the course, students write a report, present their solutions and we test their programs for errors with

an auto-tester. We use automated clone detection tool to find (very rare) cases of students copying solutions from other projects.

Evaluation of students’ projects

Students’ projects are evaluated based on the scope of the programs’ functionality implemented, programs’ quality attributes (e.g. reliability, reusability, extensibility and the efficiency of a query evaluation strategy), and the quality of project documentation.

At the end of the project course, each team is given one hour to present their work and to complete a final system testing. One test run consisting of 200 to 300 test cases covering a large set of functionalities is executed for each team. Important information such as failed cases, exceptions and timeouts (when the time taken to evaluate a query does not meet the time limit we set) are captured. Any test cases that failed are then re-run and the results are verified manually. Students are allowed to give explanations of what went wrong if they know the reason. Hence, the final testing will give us an objective score on the reliability of the students’ programs.

To facilitate the final system testing, we developed a tool called AutoTester (a server) to automate testing of students’ programs. The AutoTester works with students’ programs according to a client/server architecture concept. The AutoTester reads a set of test cases (program queries in our case) and then repeatedly sends queries to a student’s SPA program (the client). The SPA evaluates the query and returns the results back to the AutoTester for verification. To minimise problems that might occur during the final testing, a trial run is conducted for each team prior to the final testing. This not only helps to familiarise students with the testing procedures but also highlights any problems the team might have when using the AutoTester.

From the statistics of the project course, most teams manage to keep their total number of errors below 10. We believe that our relentless emphasis on reliability manage to drive home to students that testing is important in any large scale software development. We also make the following observations:

1. In most teams, at least one unique failed case is attributed to the team’s misunderstanding of the specification. Some teams purposely imposed their own restrictions on the project to reduce its complexity, and thus failed to meet our project requirements.
2. Programs implemented in Java have fewer total errors because Java is a programming language used frequently for programming assignments in most courses in NUS and hence, students are more proficient in using Java than C++.

3. C++ programs tend to have many timeouts but there are a few exceptions. We believe that the lack of error checking mechanisms in C++ forced our students to be more conscious about error/exception handling and recovery when implementing in C++, resulting in fewer exceptions. However, as most of our students are exposed to C++ for the first time, they tend to code in an inefficient way, resulting in more timeouts.

Results so far

We have been offering CS3215 since 2001, to 80–130 Computer Science students each semester. At the end of the project course, students grow to appreciate the role of software architecture, learn how to communicate in terms of interfaces, learn how to split the project work and how to document the products in a clear way that other team mates can understand. Most of the teams became very much involved in the project and were motivated to work hard in order to deliver a quality product. Though we make little effort to emulate the real world in this course, we believe students have learned some essential skills that will help them deal with real world project challenges in a systematic rather than chaotic way.

Improving students' communication skills is a major concern and focal point in our project course. The ability to translate solutions from the concept level to a proper document design, to code, test plans and other program artefacts, to describe interfaces, to use assertions, as well as to prepare and conduct team meetings in an effective way, are all communication-related skills. Communication skills integrate the essential human and technical aspects of software development. As students do not have enough opportunity to develop communication skills in assignment-based courses, it is a major issue for the project course to address. We believe the course has given students ample opportunity to experiment with a wide range of techniques to develop and master communication skills.

The project infrastructure also plays a critical role in achieving teaching goals. It helps students learn about software tools in a short time and helps instructors evaluate program solutions. A common problem in many universities where project courses are offered to a large population of students is finding enough qualified faculty members and teaching assistants to supervise them. Our teaching approach and the project infrastructure have alleviated these problems. The project handbook and project infrastructure communicate to students the 'what and how' of the project. There is a repository of project information shared among instructors, thus shortening the learning time before new instructors can advise student teams effectively. The project handbook and project infrastructure can be used as it is or customised to account for specific goals, student audiences and specific qualities deemed important in a given offering of a project course.

In summary, we find our approach to teaching the project course enhances students' teamwork and communication skills, helps them apply principles and 'best practices', and deals with practical problems related to teaching a project course for a large population of students. ■

* *This article is an abridged version of a paper presented at the 18th Conference on Software Engineering Education and Training in Ottawa, Canada, 18–20 April 2005.*



The Centre for Development of Teaching and Learning (CDTL) engages in a wide range of activities to promote good teaching and learning at the National University of Singapore, including professional development, teaching and learning support, research on educational issues, and instructional design and development.

contributors

Seah Kar Heng
Irina Aristarkhova
Stan Jarzabek
Pin-Kwang Eng

advisor

Daphne Pan

editors

Teo Siok Tuan
Sharon Koh

layout

Ma Lin Lin

© 2005 *CDTL Brief*
is published by the

Centre for Development of Teaching and Learning.
Reproduction in whole or in part of any material in this publication without the written permission of CDTL is expressly prohibited.

The views expressed or implied in *CDTL Brief* do not necessarily reflect the views of CDTL.

An online version is available at our web site.

Comments, suggestions and contributions should be addressed to:

The Editor, *CDTL Brief*
Centre for Development of Teaching and Learning
Central Library Annexe, Level 6
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260

Tel: (65) 6874-3052

Fax: (65) 6777-0342

Email: cdtpost@nus.edu.sg

<http://www.cdtl.nus.edu.sg>

Printed in Singapore by First Printers Pte Ltd.