First and Fourth Year Design-build Team Projects: a Comparison

David C Levy

School of Electrical and Information Engineering
The University of Sydney
dlevy@ee.usyd.edu.au

ABSTRACT

The new Software Engineering degree program has been designed as a collaborative effort between the School of Electrical and Information Engineering and the School of IT at the University of Sydney. The design of the new curriculum is based on the ACS, ACM and IEEE curriculum recommendations and the Software Engineering Body of Knowledge (SWEBOK), as well as the results of a detailed industry review. Close attention was paid to Computing Curriculum 2001 (CC2001) and to Software Engineering 2004 (SE2004). The CDIO framework was used to ensure the new program incorporates the full set of knowledge, skills and attitudes that students should possess as they graduate from university. As part of this exercise, team projects are being introduced to all years of study, from first through to fourth year. This paper compares team based student design-build projects taken by students in the first and fourth years of study. The objective is to determine what changes there are, if any, in the students' approach to their projects in terms of the CDIO principles. The students' approach is assessed for conception, information gathering and analysis, experimentation (where necessary), design process, implementation, testing and presentation. The results indicate a broad improvement in approach. The paper will present results in each of the above areas and make some recommendations concerning the imparting of CDIO skills.

KEYWORDS

CDIO, curriculum design, design-build experiences

INTRODUCTION

The new Software Engineering degree program implemented in the School of Electrical and Information Engineering at the University of Sydney is based on the Australian Computer Society (ACS), the American Association of Computing Machinery (ACM) and the Institue of Electrical and Electronic Engineers (IEEE) curriculum recommendations along with the Software Engineering Body of Knowledge (SWEBOK), as well as the results of a detailed industry review. Close attention was paid to Computing Curriculum 2001 (CC2001) and to Software Engineering 2004 (SE2004) both published jointly by the IEEE and ACM. The CDIO framework was used to ensure the new program incorporates the full set of knowledge, skills and attitudes that students should possess as they graduate from university.

The philosophy of the degree program is to produce graduates who combine a high level of software skills with the design approach inculcated by an Engineering education and who will be able to apply engineering principles to the problem of managing complexity in the design of large software systems, ranging from global scale distributed enterprise networks to embedded systems, including components of software, systems, telecommunications and engineering management. Typical graduates might lead the development of complex

software applications in specialist areas such as banking and insurance, telecommunications, real-time systems, multimedia and industrial control, bringing to bear skills in disciplined engineering design and a systems engineering approach.

In order to equip graduates with the skills they need for the modern workplace, the new program has introduced project-based learning extensively through all four years of study, following the recommendations of the CDIO framework and modern PBL trends. This paper describes two of the design-build experiences used in the first year introduction to engineering course and one of the final year courses and compares the students' approaches and performance to determine if they are acquiring both the technical skills and team skills we seek to impart.

Alignment with CDIO

Conceive

Conceiving of software systems is dealt with in several ways. In the team projects in years 1, 3 and 4 students are expected to brainstorm and then refine their ideas by applying fundamental mathematical, abstraction and modeling skills. Mathematical thinking prepares students for all stages of system development from design to the correctness of the final implementation [ACM Communications, September 2003].

There is thus a strong focus on mathematics during the first year (12 credit points), along with foundation units in Software, Computer Engineering and the Engineering Profession. In second year a full unit of discrete mathematics and graph theory is offered. Software abstraction and modeling skills are developed in most units, particularly in INFO2110 System Analysis and Modeling. In the electives, students may study domain specific modeling in depth, for example in ELEC5614 Real Time Computing.

Software Design

A core issue in CDIO is that of Design, and this is dealt in the program in INFO1103 Introduction to Programming, INFO1105 Data Structures and INFO3220 Object Oriented Design which build students' skills in software development, covering object-oriented software development with design-by-contract, which is the state-of-the-art in industry. Students are then expected to use these skills in the team project in 3rd year and the research project in 4th year. Part of the design process includes Software Risk Analysis, which is dealt with in depth in INFO2110 System Analysis and Modeling. ELEC5618 Software Quality Engineering addresses risk as an essential issue in any software project.

Software Implementation

Implementation issues are addressed in INFO1103 Introduction to Programming and INFO1105 Data Structures in first year, and then at increasingly advanced levels in the later years to considerations of large scale systems in COMP5348 Enterprise Scale Software Development. Domain specific implementation issues are considered in recommended electives such as Real-time Computing and Multimedia. A key part of the implementation process is Software Testing, Verification, Validation and Quality Assurance

These topics are addressed in INFO2110 System Analysis and Modeling, the Software Design units and ELEC5618 Software Quality Engineering. The concepts are reinforced in all the software development units and students are required to demonstrate their skill in this area in the software project.

Operate

Software operation requires installation, configuration and maintenance. Maintenance of large software projects can be very expensive, consuming more resources than the C, D and I phases combined. These issues are addressed in INFO3402 Management of IT Projects,

ELEC5618 Software Quality Engineering and COMP5348 Enterprise Scale Software Development. INFO3402 is taught by a team of industry professionals.

Team Project Based Learning

CDIO emphasizes the need for students to undertake authentic, team-based activities, termed Design Build experiences onto which more abstract learning can be mapped (CDIO Standard 5, Crawley 2007). These also provide the natural context in which to teach many CDIO syllabus skills such as teamwork, communications, designing and implementing. Although the software engineering process is treated specifically in INFO3402 Management of IT Projects, it is instilled in students from the word go in all relevant units. Problem-based learning is used, in conjunction with increasingly demanding projects, to introduce design-by-contract, refactoring, good abstractions, high-level terms including pre- and post-conditions, ability to produce a design with appropriate information-hiding, evaluation of the quality of a design, verification and validation, etc. Students are expected to display all of these skills in the 3rd year team project and in the 4th year engineering capstone project. Every student must complete a final-year capstone project (worth twice as much as other units), either individually or as a team. In the capstone project, students bring together all the wealth of knowledge gained over the previous 3 years. A wide range of topics are made available to the students, and they are also encouraged to suggest their own topics. Many students choose to undertake original research for this project; others may choose an industry-based project, working with an industry co-supervisor.

The outcomes of the final project comprise a working piece of software or system, along with a project report, typically of around 50 pages, including details of the project management. Students also give a 15 minute presentation on their work. The best 30 projects from all the degree streams in the School, chosen out of about 120, are then also presented at the Faculty's annual open day, to which many local industrial and commercial enterprises are invited.

Starting in first year, in ELEC1601 Foundations of Computer Engineering, students work in small groups, so they experience many of the issues of team interaction that are important in practice. Also, students take responsibility for planning their own learning to meet required objectives, so they will develop skills to learn from resources including reference materials and examples. As the program progresses, students undertake more advanced projects in the advanced elective units of study, leading to a full team-based project in 3rd year and a research-based thesis project in the final year.

In ELEC1601 students working in groups design, program and test a small system involving sensing and control. The project allocated in 2008 required students to assemble a small robot buggy kit, controlled by a PIC microprocessor and equipped with contact sensors and program it to negotiate a maze then at the end of the maze to move in a dance and play a tune.

In ELEC5614 Real Time Computing, the fourth year course, students used model-based engineering to design, program and simulate a parking garage management system for a parking garage on the campus which would sense cars entering and leaving each level and display at the entrance to the garage the total number of free spaces and at the entrance to each level the number of free spaces on that level.

Both projects required software development. The first year project required kit assembly and a visible physical result, while the 4th year project required simulation in Rational's Rose Technical Developer environment, requiring a mix of model-based engineering and coding..

Assessment

To facilitate comparison, both projects were assessed identically. Students were required to form teams, produce an initial individual assessment, proposal and project plan. The next

step was to submit a group design, followed by a group implementation progress report. This was followed by a group demonstration of the working system followed by an individual overall project report assessing the outcomes and including a reflective review of the project.

The first year buggy project demo took the form of a race-off of the buggys, which were required to traverse the maze in less than 6 minutes to qualify as have completed this phase. Only one team's buggy did not meet this requirement with times ranging from 2:10 to 5:43 minutes.

The fourth year project required the simulation to process a file of car arrival data. Students were given a test file for development and then had to process a different file for the demo, showing where the cars parked. The data only gave times of entry to the parking garage and to the section chosen. The software was free to place the cars anywhere in the section.

Both projects, following [14], required students to display innovation, to conduct research and experiments into the problem in order to optimise the solution, to manage the team by allocating work fairly and tracking and integrating its completion, dealing with time and resource budgets and show a connection between theory and practice.

Performance

As we wished to track improvement in the students' performance, we chose as metrics the following criteria shown in the table below. The four reports were submitted online and graded online using the WebCT learning management system. Students were given explicit instructions as to the requirements for each report For the demos, two external industry practitioners were invited to assist with the grading. Students' grading criteria were more stringent in 4th year, where students were required to show greater management skills and a more organized attack on the problem. Actual grades do not differ widely, indicating the criteria were met. The report and demo grades were released to the students, but the innovation through management assessments were kept for internal use only. For the 1st year projects, there were 171 students in 30 groups of 5 ot 6, while for the 4th year project there were 22 students in 7 groups of 3 or 4.

Table 1. Project outcomes

Requirement item/performance (out of 5)	4th yr	1st yr
Innovation – inventive solution	3.82	3.61
Modeling and analysis	4.31	4.05
Planning and project management	4.42	3.91
Theory and analysis	4.11	4.32
Time management	4.19	3.87
Team management	4.35	4.01
Report 1: Initial evaluation	4	3.58
Report 2: Initial design	4.29	4.33
Report 3: Implementation	4.31	4.28
Report 4: Quality of solution and reflection	4.06	4.45
Project demo and presentation	3.94	4.42

The most distinctive outcome from the evaluation is that the 4th year students were better at managing their teams. They also devoted more effort to evaluating the problem and planning their attack before coding. They were not as good at reflective evaluation. The final demo

results were also weaker, perhaps because the performance requirements wer more stringent.

Conclusion

The SE program provides a strong, up-to-date curriculum that is relevant to the needs of Australian Industry and meets the recommendations of EA/ACS, IEEE/ACM SE2004 and SWEBOK. It follows the CDIO framework and provides students ample opportunities to develop core skills, to specialize and to test their abilities in a series of projects that demand increasing capabilities through the four years of the program. Between the 1st and 4th year projects students show a clear improvement in approach, taking more care over the initial phases of analysis and knowledge acquisition and showing better management skills.

Bibliography

[1] ACS 1995: *Core Body of Knowledge for Information Technology Professionals,* Australian Computer Society, accessed at http://www.acs.org.au/ictcareers/index.cfm?action=show&conID=cbok on 2 Oct 2008

[2] IEEE 2005: *Computing Curricula 2005: The Overview Report,* Inst. of Electrical and Electronic Engineers, accessed at http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/CC2005-March06Final.pdf on 2 Oct 2008.

[3] SWEBOK 2004: Guide to the Software Engineering Body of Knowledge, accessed at *http://www.swebok.org/* on 30 July 2008.

[4] CC2001: *Computing Curricula 2001*, accessed at http://www.sigcse.org/cc2001/ on 2 Oct 2008.

[5] SE2004: Software Engineering 2004: *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,* accessed at http://sites.computer.org/ccse/ on 31 July 2008.

[6] Crawley 2007: Edward F. Crawley, Johan Malmqvist, Soren Ostlund, and Doris Brodeur: *Rethinking Engineering Education: The CDIO Approach,* Springer-Verlag.

[7] IEEE Std 610.12-1990: *IEEE Standard Glossary of Software Engineering Terminology,* IEEE, Piscataway, NJ.

[8] Sukkarieh 2004: Salah Sukkarieh, *The University of Sydney, Faculty of Engineering: Contextualised Graduate Attributes,* accessed at http://www.itl.usyd.edu.au/graduateAttributes/facultyGA.cfm?faculty=Engineering on 2 Oct 2008.

[9] EA 2008: *Engineers Australia Program Accreditation,* accessed at http://www.engineersaustralia.org.au/education/program-accreditation/program-accreditation.cfm on 2 Oct 2008.

[10] EA-Comp 2008: *Australian Engineering Competency Standards - Stage 1 Competency Standards for Professional Engineers,* accessed at http://www.engineersaustralia.org.au/shadomx/apps/fms/fmsdownload.cfm?file_uuid=0446A3D5-B812-B0F4-4B66-8AF85D4C337B&siteName=ieaust on 2 Oct 2008.

[11] ACS 1995: *Core Body of Knowledge for Information Technology Professionals,* Australian Computer Society, accessed at http://www.acs.org.au/ictcareers/index.cfm?action=show&conID=cbok on 2 Oct 2008

[12] IEEE 2005: *Computing Curricula 2005: The Overview Report,* Inst. of Electrical and Electronic Engineers, accessed at http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/CC2005-March06Final.pdf on 2 Oct 2008.

[13] SWEBOK 2004: Guide to the Software Engineering Body of Knowledge, accessed at *http://www.swebok.org/* on 30 July 2008.

[14] Malmqvist et al, *Advanced-level Design-build experiences,* accessed at http://www.cdio.org/meetings/feb03material/db_discussion.ppt on 20th March 2009.

Biographical Information

David C. Levy is director of the Software Engineering program at the University of Sydney and leads the School of Electrical and Information Engineering's CDIO Initiative and is Chair of the ANZ CDIO regional group. His current scholarly interests are in real-time distributed systems and in team-project based learning.

Corresponding author
A/Prof David C. Levy
The University of Sydney
NSW 2006
Australia
61-2-9351-6579
dlevy@ee.usyd.edu.au