# A Software Engineering Program Using the CDIO Framework

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 being implemented in 2008 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. This paper gives an overview of the curriculum and follows by describing two of the design-build experiences used in the first year introduction to engineering course and the final year capstone project

***Keywords***
CDIO, curriculum design, design-build experiences

## Introduction

The new Software Engineering degree program being implemented in 2008 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 informed by 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. The aim is to achieve a program that is consonant with:

> "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software." (IEEE)

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.

This paper discusses how the University of Sydney's Software Engineering program addresses the accreditation requirements of Engineers Australia and equips graduates with the knowledge base necessary to achieve the above, as well as the generic attributes required by industry and academia. This paper gives an overview of the curriculum and follows by describing two of the design-build experiences used in the first year introduction to engineering course and the final year capstone project

## The Design of the Curriculum

In designing the curriculum, we had to maintain alignment with the Engineers Australia (EA) accreditation criteria. These required that the program include personal, interpersonal and professional skills very close to those of CDIO. For the technical content, EA follows the SE2004 program guidelines and the major areas set out in the SWEBOK. There are 14 of these, covering Computer Organization and Architecture, through Data Communications and Networks to Software Project Management and Quality Assurance.

### *Core outcomes of the degree*

The units of study that make up the core are chosen to provide graduates with the fundamentals of:

- sciences, technologies and engineering,
- software development and programming language skills,
- software systems life-cycle, product requirements, software project management,
- an introduction to hardware platforms and devices: processor architecture, interfacing, digital systems,
- an introduction to computer networks and data communications technologies,
- engineering design and management,
- a set of generic graduate attributes common to all engineering graduates and
- the opportunity to specialize in a software application domain.

### *Graduate attributes*

The Software Engineering program endeavors to inculcate its graduates with the following generic graduate attributes:

- Research and Inquiry - Graduates of the program will be able to create new knowledge and understanding through the process of research and inquiry.
- Information Literacy - Graduates of the program will be able to use information effectively in a range of contexts.
- Personal and Intellectual Autonomy - Graduates of the program will be able to work independently and sustainably, in a way that is informed by openness, curiosity and a desire to meet new challenges.
- Ethical, Social and Professional Understanding - Graduates of the program will hold personal values and beliefs consistent with their role as responsible members of local, national, international and professional communities.
- Communication - Graduates of the program will recognize and value communication as a tool for negotiating and creating new understanding, interacting with others, and furthering their own learning.

Further details of the Faculty of Engineering's view of graduate attributes are available on the website http://www.itl.usyd.edu.au/graduateAttributes/facultyGA.cfm?faculty=Engineering

***Allocation of credits.***

In the University of Sydney structure, each unit of study is worth 6 credit points, and the normal load for one semester is 24 credits, i.e. 4 units. Thus an 8-semester program is worth 192 credits. In our program, we keep the core load below 144 credits to allow for several options of specialization and combined degrees.

| CORE | 1st Year | 2nd Year | 3rd Year | 4th Year | Total |
|---|---|---|---|---|---|
| MATH | 12 | 6 | - | - | 18 |
| IT | 12 | 30 | 24 | 6 | 72 |
| ELECTRICAL ENG | 12 | - | 12 | 18 | 42 |
| TOTAL CORE | 36 | 36 | 30 | 24 | 132 |
| RECOMMENDED ELECTIVES | 12 | 12 | 6 | 12 | 36 |
| FREE ELECTIVES | - | - | 6 | 12 | 18 |

***Electives and streams***

The recommended electives consist of 42 credit points taken from a table of engineering and computing topics, allowing students to tailor their degree, for example to include a full stream of e-business or of networking or embedded systems. There are several specialization streams and units available to students. These include, among others, multimedia, real-time and embedded systems, information systems, enterprise systems, e-business technology and security.

***Combined Degree Courses***

The BE SE program has been designed to permit students to undertake combined degree courses of Bachelor of Engineering in Software Engineering with the Bachelor of Science, Bachelor of Arts, Bachelor of Medical Science, Bachelor of Commerce or Bachelor of Laws over a period of 5 years, leading to the award of two degrees.

**Alignment with IEEE-CS/ACM SE2004**

The new curriculum is based on the Software Engineering curriculum developed by a joint task force sponsored by the IEEE Computer Society and the Association for Computing Machinery. Their work is described in the document "Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" available at: http://sites.computer.org/ccse/SE2004Volume.pdf

The SE2004 task force developed a Software Engineering Education Knowledge base or SEEK made up of ten knowledge areas: Computing Essentials, Mathematical & Engineering Fundamentals, Professional Practice, Software Modeling & Analysis, Software Design, Software Verification & Validation, Software Evolution, Software Process, Software Quality, and Software Management. Each of these knowledge areas is further refined to individual topics with their relevance (essential/desirable/optional). These topics may easily be groups under the CDIO headings.

The new University of Sydney Software Engineering curriculum includes all the essential topics from SE2004. A complete listing of the SEEK codes and their topic, a list of Units of Study in the new curriculum and a mapping between the units and the essential SEEK codes is available from the author

### *Our approach*

In view of the above, our approach was to list the SE2004 core program components and then group them into a set of core units of study. This gave rise to a set of 15 core units, all worth 6 credit points except for the thesis project:

INFO1103 Introduction to Programming
INFO1105 Data Structures 1
ELEC1601 Foundations of Computer Systems
ENGG1805 Professional Engineering and IT
COMP2007 Algorithms and Complexity
INFO2120 Database Systems 1
INFO2315 Intro to IT Security
COMP2129 Operating Systems and Machine Principles
INFO2110 Systems Analysis and Modeling
INFO3315 Human Computer Interface Design
INFO3402 Management of IT Projects
INFO3220 Object Oriented Design
ELEC3609 Internet Software Platforms
COMP3615 3rd year SE Team Project
ELEC4707 4th year SE Research Project
ELEC5618 Software Quality Engineering
COMP5348 Enterprise Scale Software Development

## 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 3$^{rd}$ year and the research project in 4$^{th}$ 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 Projects and Problem-based Learning**

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 project.

Starting in first year, in ENGG1805 and ELEC1601, 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 and build a small system using a generative programming environment. A typical example might be a system to perform a remote data sensing and testing task using LabVIEW, wireless networking and a remote sensor. The application will emulate an industrial remote sensing application, e.g. to monitor the health of an oyster farm.

In COMP3615, the 3rd year software development team project, students will apply the knowledge and practice the skills acquired in the prerequisite and qualifying units, in the context of designing and building a substantial software development system in diverse application domains including life sciences. Working in groups they will need to carry out the full range of activities including requirements capture, analysis and design, coding, testing and documentation.

The program is designed so that in every stream, every student will undertake a capstone project in 4th year, which may be associated with a specialized area such as Real Time Computing, in which teams may tackle a project like a traffic tool plaza with booths, auto-debit transponders and photo capture, or a controller for some type of industrial machine such as a carpet cleaning system.

Every student must also tackle a final research project, either individually or as a team. In the research 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.

The outcomes of the final thesis comprise a working piece of software or system, along with a project report, typically of around 50 pages, including details of the project management and a 15 minute presentation. The best 30 are then also presented at the School's annual open day, to which many local industrial and commercial enterprises are invited.

**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.