# An In-forming Web-based Environment for a Bachelor of Software Engineering Degree – DoIT

***Sita Ramakrishnan and Ashley Cambrell***
***Monash University, Melbourne, Australia***

Sita.Ramakrishnan@csse.monash.edu.au Ashley.Cambrell@csse.monash.edu.au

## Abstract

This paper presents our template-based approach in building a web-based system titled "Dynamic curriculum Organisation by Innovation through Technology (DoIT)". We have considered the meta-environment of any course development process and found that we can produce two kinds of knowledge assets from this environment. A delivery (asset) environment forms the basis of our traditional course delivery mechanisms. An in-forming (asset) environment can be created to engage the students to learn what they have learnt from the delivery environment. Normally, curriculum developers and curriculum implementers (lecturers and TAs) are involved mainly with only one aspect of this asset: the delivery environment. Our Bachelor of Software Engineering students also learn about what they have learnt in their undergraduate degree course by engaging with the in-forming environment of DoIT. We present a meta-environment for creating knowledge assets and show how our DoIT system fits within this educational knowledge framework.

**Keywords**: SWEBOK, Informing science, in-forming

## Introduction

DoIT is a faculty infrastructure funded project for our BSE students to interact with the Monash BSE (Course) curriculum on the web. In Australia, a Degree program is referred to as a Course and the Units taken within that Course are referred to as Subjects within the Course or Degree.

The aim of this project is for the students to be able to interact with a content map of our Bachelor of Software Engineering (BSE) Course and receive feedback on two levels of learning. This approach has been taken so that the students can understand about what they are supposed to learn in a subject in our Bachelor of Software Engineering Course in terms of guidelines set out in the Software Engineering Body of Knowledge (SWEBOK) by the ACM/IEEE Computer Society committee (Huitt, 1996; Dupuis et al. 1998; Bagert, 1999; Bourque et al., 2000). They can also interact with the system to receive a visualized interpretation of the *knowledge acquired* in terms of SWEBOK in a subject. This means that the students are able to see a static view of the content (SWEBOK) and curriculum (course/subjects map) of the BSE and are able to receive a graphical representation of what they have learnt (in terms of SWEBOK) as they move through the course. Hence they see and appreciate the bigger picture. Hopefully that helps motivate them, which in turn will improve their learning. The aim of the system is to provide a visual interpretation for the students and the academics of what the students have learnt in our BSE in terms of SWEBOK for the four years of study. The system has been designed in such a way that the web system can evolve to incorporate any changes that happen to our BSE course subjects or changes to

Material published as part of these proceedings, either on-line or in print, is copyrighted by Informing Science. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission from the publisher at Publisher@InformingScience.org

SWEBOK. DoIT website will evolve systematically with changes to the perceived and core knowledge taught over time. The web-based system, DoIT can be viewed at http://www.csse.monash.edu.au/~doit/

Cohen (1999) states "the fields that comprise the discipline of Informing Science provide their clientele with information in a form, format and schedule that maximizes its effectiveness". In the next section, we discuss the meta-environment of knowledge assets and the educational knowledge framework (see Figures 1, 2). We show that DoIT is an in-forming web-based environment for our Bachelor of Software Engineering Degree and fits within this knowledge framework. Then, we describe the software engineering design and process details of building DoIT. We conclude with some of the innovative outcomes of DoIT.

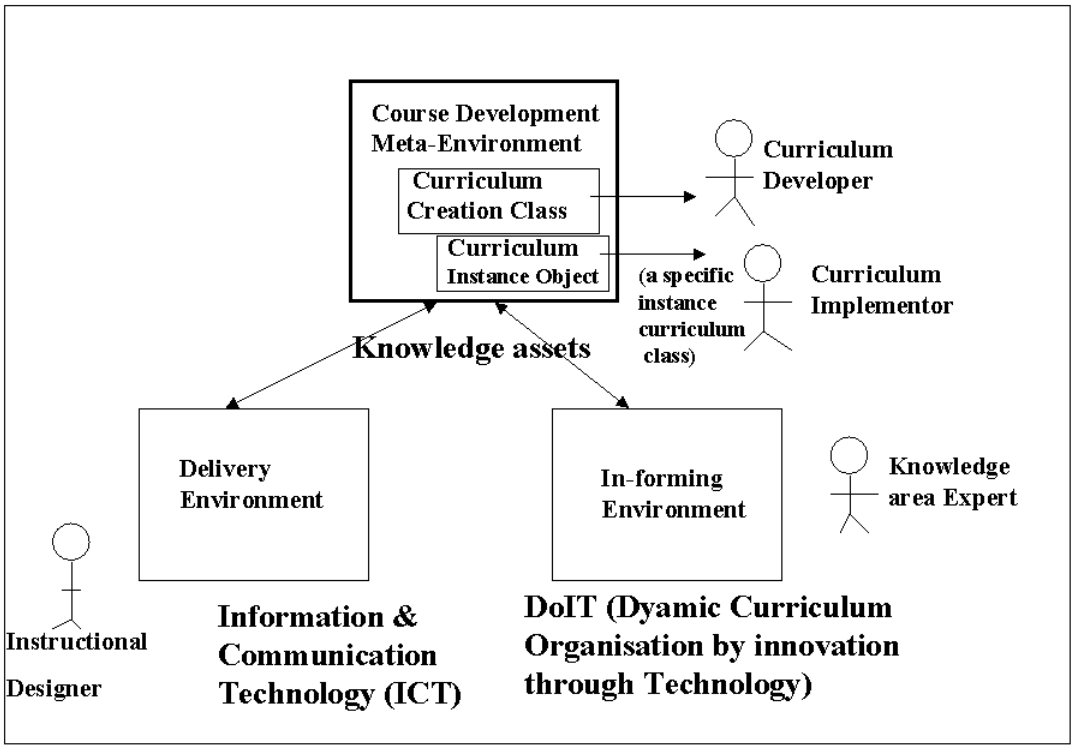


**Figure 1: Meta-Environment of Knowledge Assets**

## Meta-Environment of Knowledge Assets

We view the course development activities at educational institutions as a meta-environment for creating knowledge assets (see Figure 1). The actors involved in the curriculum design, development, implementation and delivery include knowledge area experts, pedagogical experts and instructional designers and lecturing academics. The knowledge assets derived for each degree program consists of a delivery environment and will benefit from an in-forming environment. Conducting lectures face-to-face or in virtual classrooms, assessment, feedback, and innovative use of information and communication technology in delivery is standard practice as part of the delivery environment these days. However, in-forming environment that is part of an educational knowledge framework (see Figure 2) is becoming increasingly important to show clearly what exactly are the key knowledge areas covered in a particular degree program and the core competencies of that discipline. For example, many of the accreditation of university curricula and the licensing and certification of practising professionals are taken very seriously in many engineering disciplines. Recognition of the core body of knowledge in a discipline is crucial to the development and accreditation of university curricula.

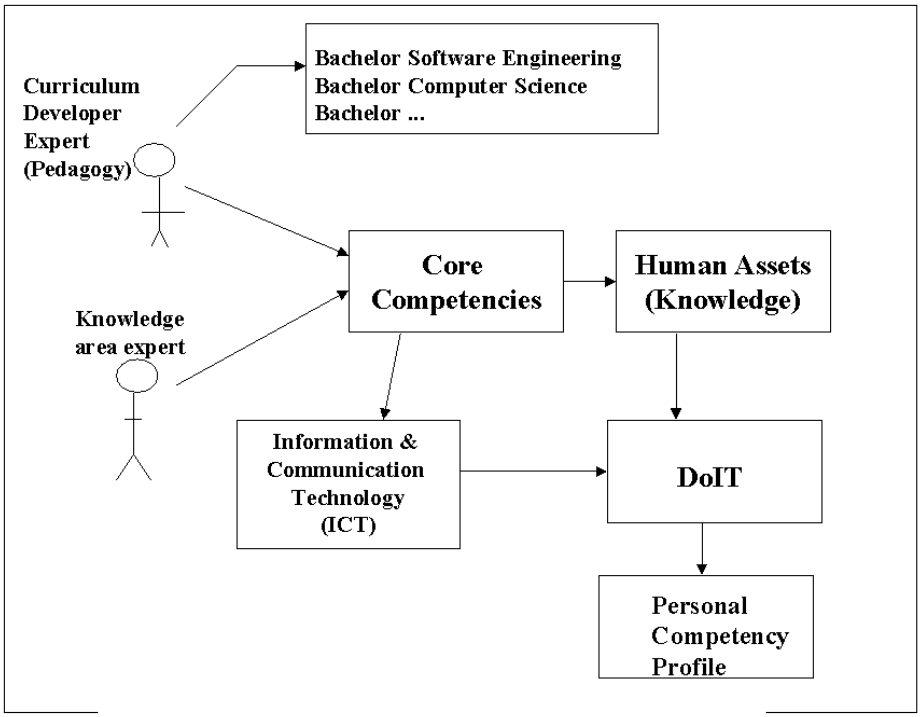


**Figure 2: Educational Knowledge Framework**

## *Educational Knowledge Framework*

Educational institutions offering degree programs such as Bachelor of Software Engineering, Bachelor of Computer Science, Bachelor of Information Systems and other programs highlight the contents of their courses with a course map and a statement of what key areas are covered in the particular course. Using the recommendations of key bodies such as ACM/IEEE CS committee, knowledge experts can list the core competencies and associated fundamental knowledge deemed mandatory for a particular degree program, and produce an in-forming environment such as DoIT for the clients (see Figure 2). The clients may be students, staff and quality audit personnel in the institution. DoIT can provide personalised competency profiles for students as they progress through their undergraduate programs.

Our DoIT project has used the SWEBOK classification to map the core units in our current BSE curriculum against the SWEBOK knowledge key areas. DoIT is an innovative learning system for students to learn about what skills they have learnt as they move through the course. DoIT is also an innovative curriculum-planning system.

The next section provides some key engineering aspects of DoIT.

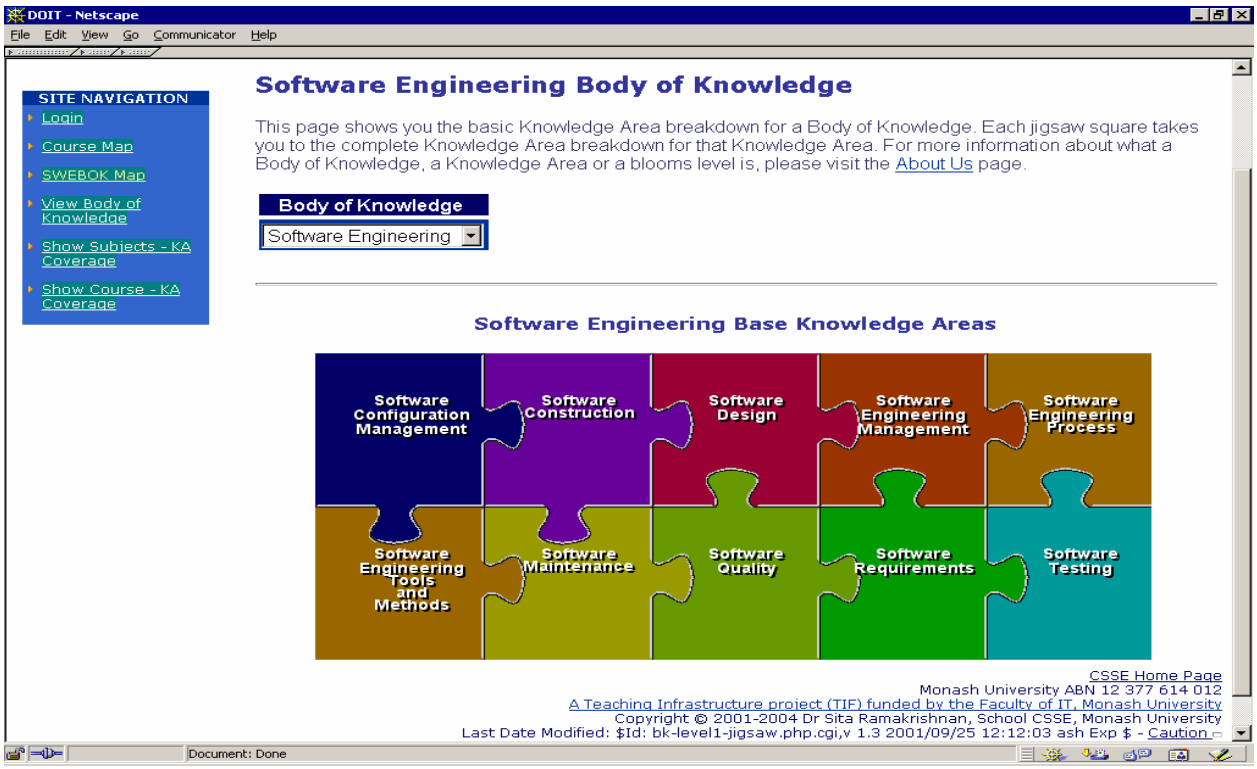


**Figure 3 – Software Engineering Body of Knowledge (SWEBOK)**

## *DoIT - Dynamic Curriculum Organisation by Innovation through Technology*

Our Bachelor of Software Engineering program contains a number of core subjects, which mirror the core knowledge required of a S.E. professional as specified in SWEBOK (see Figure 3). Each student can request for his/her personal competency profile (PCP) as he/she progresses through the course. The system responds with a dynamic curriculum organisation for that student: DoIT produces a PCP by noting the bloom levels (Huitt, 1996) for each SWEBOK key and subkey area as defined for each of our BSE core subjects.

We outline some of the key considerations in engineering DoIT next, such as entity relationships, security aspects of student data, choice of language, standards w.r.t. web design, documentation, configuration management, usability, user interface issues and access to student record systems.

### Entity relationships

At present, we have considered only the Bachelor of Software Engineering and hence only SWEBOK as the body of knowledge. The Body of Knowledge is broken down into Knowledge Areas (KAs), which represents a sub-section of the entire body of knowledge. For example, Software Design is a KA of SWEBOK. Software Design can (and is) also broken down into sub-KAs. DoIT can be accessed from http://www.csse.monash.edu.au/~doit to view its capabilities.

From this overall idea of BK containing KAs (and each KA possibly having sub-KAs), we have sketched out a relation database ER diagram that modeled this well. Matching subjects to KAs has also been represented in a similar way, except that the relationship is a many-to-many relationship (BK → KA). All other database design aspects of the system have followed suit as many-to-many relationships between components; Students ↔ Subjects, Subjects ↔ Courses. (see Figure 4).

## Security aspects

The site would be storing sensitive data about students, and so had to be designed with security and data protection in mind. There must also be different classes of users using the site, so that administration aspects of the site could be "locked" off from other users. The student's database table has been abstracted slightly into a more general Users table, and a User Type table added which has enabled condition linking to a Student Information table. This simple structure has provided the basis for the multi-rolled access needed.

## Choice of Language and DB

It was decided that if the system could be implemented as a web application, it would allow fast and portable access to the system to anyone with an Internet connection and a web browser. A survey of programming language options suitable to be run on available resources was carried out, and PHP was chosen, due to its speed, availability on various operating systems, and its ability to connect to a multitude of databases. Oracle was chosen as the database system due to its availability in the department and robust nature.

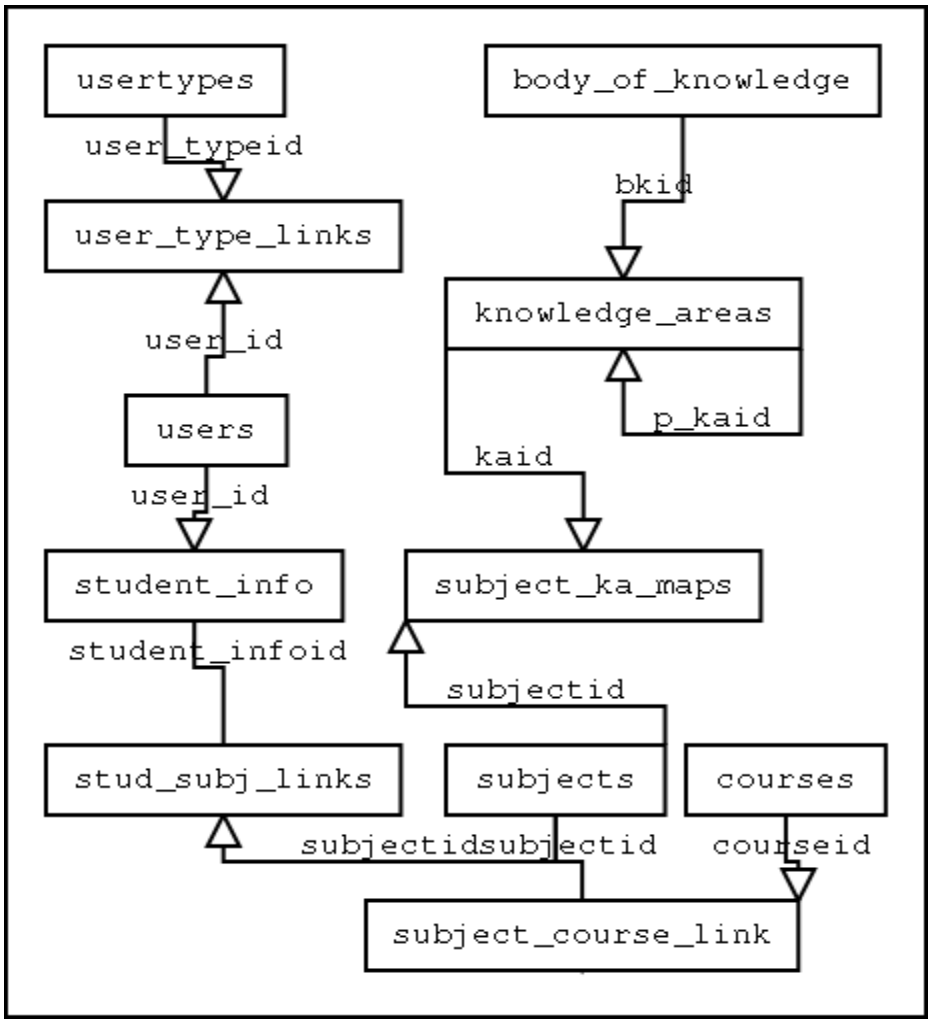


**Figure 4: Database Schema**

## Standards w.r.t. web design and documentation

Monash University has long has strict guidelines as to how web pages that are accessible from the outside world should look. These guidelines provide detailed example templates and predefined Monash style sheets for various browsers. This greatly simplified the visual design of the site. There was one example design, which lent itself very well to the navigation needs of the site.

Research on how to ensure code quality and maintainability with PHP revealed there are no official standards or recommendation on coding. This is mainly due to PHP flexibility and its ability to be directly embedded in HTML much like ASP. Due to PHP similarity with C, but without C's strict types, a similar style to the ANSI C coding style has been chosen. Code commenting has followed PHP Doc recommendations, which is based on Java's Java doc style of code commenting. This enables the code to be later run through a documentation generator, which would automatically create indexed and cross-referenced HTML documentation.

## Configuration Management

CVS has been chosen as the method for configuration management, due to its availability and proven reliability. WEBDAV (Web-based Distributed Authoring and Versioning) is the emerging standard in web-based collaboration. IETF Delta-V Working Group is about extending the web with versioning and configuration management (Amsden, 1999; Whitehead, 2001). Although alternatives such as WebDAV (using DeltaV versioning) and RCS (4.5 Configuration management) were looked at, it was found that DeltaV was still in IETF approval stage with no working code (Whitehead, 2001), and that RCSs only deals with files individually, which would complicate configuration management.

## System Design

The overall system design has lent itself well to a logical breakdown of more or less independent sections. All high-level entities handling functions can be separated into their own “libraries”. This has also allowed the abstraction of database queries so that there are no direct database calls, only calls made from within functions. This allows the site to be shifted fairly easily to other database systems.

Basic load, save (which handled new object creation as well as updating), and general debug routines have been created for each entity type. A library has also been created to handle functional overlaps between entities. These have included Subject ↔ KA mapping functions, Student ↔ Subject, Course ↔ Subject mapping and linking functions. These libraries contain the necessary functions to recurse through the tree structure that the KA ↔ sub-KAs design made possible.

## Usability and User Interface Issues

Originally the representation of the Subject ↔ KA mappings for students was going to be a jigsaw that symbolized the relation between knowledge components, however this would require multiple levels of jigsaws as KAs possibly had sub-KAs. Students are notorious for losing interest if they have to trawl to far to find out information. Thus it was decided that a tree representation of the KAs would be the best way for students to see a complete overview of the KAs and their relationship to subjects and to the Body of Knowledge.

## Access to student records Database

A critical design aspect of the system we had to face was whether we directly linked to the student records database (if we were even allowed to), or whether we imported the data into the system. If we imported the data, what would be the best method, and what format could we get the data in. The first of these

questions was answered fairly quickly and predictably; we were not even getting near the centralized student records. As this was the case, this left us with the "Callista" system, which enabled us to export queries from the student records system as CSV (comma delimited) files. The next step was to decide what information we needed from student records. From the Users table, we needed basic first name, surname, username, and an email address so we could contact users if needed. To fill in the Students table we needed Student ID numbers, their academic year, and course start date. We also needed a list of all the subjects the student has successfully completed.

A web upload and record importer was designed that would be able to:

- Handle columns of needed data at any location. As the reporting format sometime changes, we could not rely on columns being at fixed positions.
- Handle the different ways a subject can be “completed” There are a couple of ways to “complete” a subject; by doing and passing the subject, or, by obtaining a credit transfer from another subject or course. The student records system stores these completions with entirely different fields.
- Handle updates to students past subject records without munging any existing data from a previous import.

Interfaces have also been created to handle other data input requirements. These included Body of Knowledge, Knowledge Areas (and sub-KAs), Course and Subject information, and interfaces that would allow the mapping of KAs to Subjects and Subjects to Courses.

## Related Work

IEEE-CS and ACM have been working on developing a comprehensive body of knowledge or software engineering (SWEBOK) (Dupuis, 1998). Carnegie-Mellon Software Engineering Institute [CMU/SEI-99-TR-032] has produced a document that effective curriculum design.

In 1998, the Amsterdam Faculty of Education (EFA) (Wielenga, 2000; Wielenga et al. 2000) adopted a new curriculum structure for Teacher Education as part of their national curricula and innovation strategies in the Netherlands. Some of the publications from that group are on: professional development of faculty, constructivistic learning, and quality assurance (Odenthal et al. 2000; Terwindt, 2000), and integrating competency based assessment to prove competence by integrative assessment and a web-based Portfolio system in a dynamic curriculum (Wielenga, 2000; Wielenga et al. 2000).

DoIT system applies these ideas and recasts our BSE curriculum and content with the objectives expressed in the documents cited above. The important principles of the SWEBOK project have been to establish a transparent process by fully documenting and publishing the development process. The SWEBOK project also aims to build over time, consensus in industry, amongst professional societies, academia and standard-setting bodies on a list of knowledge areas relevant to SE discipline and profession. The list includes topics based on engineering taxonomy, Bloom's taxonomy and knowledge areas of related disciplines.

We have designed our BSE course in part on earlier published work of SEI. DoIT links SWEBOK [CMU-SEI-99-TR032] and our BSE curriculum and is a vehicle for the students to learn what SE core (SWEBOK) areas they have learnt in our Bachelor of Software Engineering program by integrating innovation through technology.

The way the term "in-forming" is used in this paper comes from Boland (1987) who wrote: "… information is the **in** ward- **forming** of a person that results from an engagement with data. Figures 1-2 reflect some of the ideas from Cohen [1989]. As shown in Figures 1-2, this environment can be used to create personalised competency profiles by students.

# Conclusion

Since we have produced a generic design and have followed a template approach in building DoIT for our Bachelor of Software Engineering Course, we have produced a customisable in-forming environment. The digital portfolio available from DoIT enables our BSE students to view their progression in learning, manages their knowledge capabilities and also contributes to innovation in institutional quality audit process. DoIT works on a number of levels:

- Innovative learning system *for students* to learn about what skills they have learnt (as per SWE-BOK) as they move through the course.
- Active curriculum where *the students and academics* teaching into our BSE can view whereabouts in the course across all subjects, a theme (knowledge areas as per SWEBOK) is taught.
- Assist in the *accreditation process of our BSE* by the accrediting bodies such as ACS and IE Australia as our course is mapped to the core knowledge areas as articulated by SWEBOK 2000 by ACM/IEEE CS Society.
- Customizable to produce a curriculum tracking system along the lines of what is required by *CHEQ's Monash Graduate Attributes project* (*Monash 2020 vision of Monash graduate capabilities*) as part of Australian University Quality Agency's (AUQA) audit requirement.

# Acknowledgements

The authors wish to thank the Dean of the Faculty, Prof. John Rosenberg and Assoc. Dean Teaching, A. Prof. John Hurst for funding the DoIT project, acknowledge the support of the School of CSSE, Prof. David Abramson, Assoc Profs Trevor Dix and Christine Mingins for their support. Thanks also to Ms Karen Fenwick and Silvia Scolaro for their administrative support.

# References

Amsden, J. (1999, Sep.) *IETF Delta-V Working Group*, Extending the Web with versioning and configuration management, Retrieved Aug. 2001 from the World Wide Web, http://www.webdav.org , http://www.webdav.org/deltav/

Bourque, P, Dupuis, R. and Abran, A. (2000) Developing Concensus on the Software Engineering Body of Knowledge, *World Computer Congress*, Beijing, China, Aug 21-23

Bagert, D. et al. (1999 Oct.) Guidelines for Software Engineering Education, Version 1.0

Boland, R.J. (1987) The In-Formation of Information Systems, *Critical Issues in Information Systems Research*, ed. By R.J. Boland and R.A. Hirchheim, Wiley, N.Y.

Cohen, E. (1999) From Ugly Duckling to Swan: Reconceptualizing Information Systems as a Field of the Discipline Informing Science, *Journal of Computing & IT*, 7(3), pp. 213--219

Dupuis, R. et al. (1998, Sep.) A Guide to the Software Engineering Body of Knowledge: A Straw Man Version. [CMU-SEI-99-TR032] Los Alamitos, California: *IEEE Computer Society*

Huitt W.G. (1996) Bloom's Taxonomy of the Cognitive Domain, Retrieved Feb. 2001, http://www.valdosta.peachnet.edu/~whuitt/psy702/cogsys/bloom.html

Odenthal, L, Kuiper, W and van den Akker (2000 Sep.), J. Curriculum Design as professional development for teacher educators, *ECER Conference*, Edinburg

Terwindt, S. (2000, Feb.) Constructivist Learning: also for the Faculty!, *SITE Conference*, San Diego

Whitehead, J. (2001, May), Distributed authoring and versioning, *IETF WEBDAV Working Group*, World Wide Web Distributed Authoring and Versioning, Retrieved Aug. 2001 from the World Wide Web, http://www1.ics.uci.edu/pub/ietf/webdav/

Wielenga, D. (2000, Feb.) Proving Competence: Integrative Assessment and Web-based Portfolio System in a Dynamic Curriculum, *SITE Conference*, San Diego

Wielenga, D, Ritzen, M and Kosters, J. (2000, Feb.) EFA's web-based Portfolio System, *SITE Conference*, San Diego

# Biography

**Sita Ramakrishnan** is a senior academic in the School of Computer Science and Software Engineering, Monash University, Australia. She holds a PhD in Validating Interoperable Distributed Software and Systems. She has active interests in Object-Oriented Software Engineering research in the validation of distributed software components, compliance and conformance testing of software components, validation of software testing tools, concurrent and distributed systems and software engineering, component-based architectures and development, O-O method, process, quality, metrics and testing, web engineering, web technologies in education and teaching. She has published refereed papers in International Journal & conferences on software engineering on quality, reuse, software metrics, evaluation, testing and SE Education. She has played a key role in the curriculum development of Bachelor of Software Engineering course at Monash University. She has been the Organizing Chair and a Program committee member for the Annual International Conference on Technology of Object-Oriented Languages & Systems (TOOLS Pacific) for several years.

**Ashley Cambrell** is an applications analyst – programmer in the School of Computer Science and Software Engineering Monash University, Australia. He completed a Bachelor of Computing and has since worked on various projects centered around developing web based systems that spur on teaching innovations. Recent projects include a dynamic web base system to manage Industrial Experience projects whilst prompting code and resource reuse to students, improving systems to help facilitate student feedback, and re-engineering and extending a telerobotics system to allow the use of mechatronic devices for education over the internet. He has co-authored a paper on artifact reuse in an educational environment.