PROGOSS: MASTERING THE CURRICULUM

Richard Gluga[a], Judy Kay[a], Raymond Lister[b]

Presenting author: Richard Gluga (richard@gluga.com)
[a]School of IT, University of Sydney, Camperdown NSW 2006, Australia
[b]School of Software, University of Technology Sydney, Ultimo NSW 2007, Australia

KEYWORDS: curriculum design, curriculum mapping, learning standards, mastery, assessment

ABSTRACT

In education, we need to design effective degree programs of study that meet authoritative curricula guidelines. This is challenging because of the size of the curriculum and complexity of degree program structures. When dealing with data of this size and complexity, traditional spreadsheets are a clumsy way of storing the data. A database is a better option, especially when the database is accessible over the web. We created ProGoSs to effectively tackle this complexity. ProGoSs is a web-based system that maps curricula learning goals and mastery levels to individual assessment tasks across entire degree programs. ProGoSs enables academics to answer important questions such as: Does our degree teach the essential core defined in a recommended curriculum? Where in our degree are particular parts of the recommended curriculum taught? Does our degree ensure a solid progression in building skills? Where and how do we assess the learning achieved by bare-pass students on particular parts of the recommended curriculum? We present the design and implementation of ProGoSs and report on its evaluation by mapping multiple programming subjects from multiple universities to the ACM/IEEE Computer Science 2013 topics and learning objectives. This includes a mapping to various levels of Bloom's Taxonomy to capture mastery.

Proceedings of the Australian Conference on Science and Mathematics Education, University of Sydney, Sept 26th to Sept 28th, 2012, pages 92-98, ISBN Number 978-0-9871834-1-5.

INTRODUCTION

Hausman (1974) proposed that by 'mapping' out all the elements of the curriculum and their relationships to learning activities, 'it should be possible to analyse and compare the structural qualities of a continuing series of lessons' and to better plan more effective programs of study. Eisenberg (1984) noted that 'those interested in curriculum need accurate assessments of the current state of affairs in an educational institution ... mapping is intended to reveal the bottom line, the actual curriculum being taught to students'. However, Eisenberg also acknowledged that doing so is 'cumbersome and rarely undertaken'.

Until recently, most published degree-level curriculum mappings were in the medical domain. Britton et al. (2008) described a system developed at the University of Oklahoma that was designed to 'make the implicit curriculum explicit and transparent to ... identify gaps or unnecessary redundancies in course content [and to] link elements of the curriculum together within a course, a semester, a professional year, and the entire program'. Faculty members were required to define specific learning objectives for their subjects, map these to relevant professional accreditation syllabus documents and program outcomes, and specify how they were taught and assessed in each subject. This data was then used to generate semester-level and program-level reports outlining 'course integration in the curriculum, sequencing of courses, adequacy of prerequisite coursework, course effectiveness in holding students accountable for prior knowledge and skills, assessment methods'. The system helped the faculty identify subjects that needed 'revision and renewed alignment with program outcomes', and also subjects that 'required re-sequencing in the curriculum in order to build students knowledge and skills more intentionally and effectively'. Willett (2008) provided a review of similar systems also in the medical educational domain.

Within Australia, the academic attitude toward defining curricula and learning standards has recently taken on a greater sense of increased importance due to the creation of the Australian government's Tertiary Education Quality and Standards Agency (TEQSA, 2011). This agency will register and accredit all higher education providers. As part of its role, the TEQSA standards framework will assess graduate outcomes against agreed academic disciplinary standards. As a consequence of the creation of TEQSA, a number of projects have been started in Australia, to devise learning standards in various disciplines (Krause, Barrie, & Scott, 2012). The Australian Learning and Teaching Council (ALTC) conducted the Learning and Teaching Academic Standards Project. The project defined threshold learning outcomes in eight discipline groups. For example, the discipline group for Engineering and ICT (ALTC, 2010) devised a broad set of five outcome areas: (1) Needs, context and systems, (2) Problem solving and design, (3) Abstraction and modelling, (4) Coordination and

communication, and (5) Self management. While providing a useful, high level categorization of learning outcomes, those five broad outcomes leave implicit much of the detail of recognized Engineering and ICT curricula. We illustrate that complexity in the next section, for the computer science curriculum.

CURRICULA COMPLEXITY: COMPUTER SCIENCE AS A CASE STUDY

Approximately every 10 years, the Association for Computing Machinery and IEEE Computer Society (ACM/IEEE) release curricula recommendations that specify a list of topics and learning outcomes for Computer Science. The first draft ('Strawman') of the Computer Science Curricula 2013 (CS2013) is now available (Computer Science Curricula 2013: Strawman Draft). This 172 page document lists 1366 topics and 1041 learning outcomes. These are categorised into 18 top-level Knowledge Areas (KAs) and 155 sub-level Knowledge Units (KUs). The 1366 topics are further categorised into Tier-1 Core (257), Tier-2 Core (328) and Electives (781). Tier-1 Core topics are considered essential for all Computer Science programs in every institution. That is, 'a curriculum should include all topics in the tier-1 core and ensure that all students cover this material'. Tier-2 topics are also regarded as highly important, and institutions are required 'to include at least 80%' of these in their Computer Science programs. Institutions are free to select whichever Elective topics are most relevant for their individual programs. The learning outcomes defined in each KU relate to the topics in that KU, but also contain a 'level of mastery' component. The CS2013 Strawman draft proposes a three level mastery scale that appears to be loosely based on Bloom's Taxonomy (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). The aim of these objectives is to capture the maturity and proficiency of students as they progress through the subjects of a Computer Science program.

The Knowledge Areas and Knowledge Units of the CS2013 are not designed to, or expected to, have a one-to-one mapping to actual program subjects. That is, subjects may teach and assess topics from multiple KUs, and even multiple KAs. It is thus up to the program designers and individual subject lecturers at each institution to coordinate which of the 585 core topics and 781 elective ones are taught and for those that are to be included, where this will happen. This includes coordinating the order and the level of mastery that topics are taught and assessed in. The program design process is critically important. An ineffective design may result in serious flaws in the curriculum. For example, it may unintentionally omit even core topics, or it may fail to ensure that students reach the necessary level of mastery, or it may lack continuity in a student's study path (that is, insufficient pre-requisite knowledge in going from one subject to the next).

Program design is further complicated by the need to differentiate between the aspirational outcomes that are typically only achievable by the top-performing students vs the minimal outcomes that are to be achieved by all passing students. That is, while the subjects of a program may cover all core topics at an appropriate level of mastery, how many of these topics and at what level do bare-passing students graduate with?

PREVIOUS WORK ON COMPUTER SCIENCE CURRICULUM MAPPING

CITIDEL (Knox, 2002) is a Digital Library repository. It includes a Syllabus section that has a collection of resources that are mapped against the Computer Science Curriculum 2008 (CS2008) Knowledge Areas and Knowledge Units. These resources however are mostly subject outline documents or lecture schedules from different institutions. The granularity of mappings is very coarse, and there is no notion of a full program or a higher-level structure above a subject. CITIDEL in itself does not help visualise and design actual Computer Science degree programs.

COMPASS (Abunawass et al. 2004) is a Moodle plugin that allows mapping of Computing Curricula 2001 outcomes to subject assessments, using Bloom's Taxonomy to classify the level of mastery. The system was developed to support review and accreditation of the Computer Science curriculum at the University of West Georgia. This system has several limitations. Subject lecturers were directed to external websites and Word documents to pick relevant outcomes and fill in the COMPASS web-forms. Also, all reporting was done manually via running custom SQL queries against the system database.

THE DATABASE APPROACH: *ProGoSs*

Many of the learning standards projects currently under way in Australia are using spreadsheets to record curriculum mapping data. However, spreadsheets are a clumsy solution when dealing with data of this nature, size and complexity. Introductory textbooks on databases often describe the

advantages of databases over spreadsheets, so in this paper we shall only briefly describe those advantages. Databases separate how the data is stored inside the computer and how it is presented to the user. Furthermore, this separation allows many different views for the same data, so that data is presented in the most appropriate way for different types of users, or even for the same user when that user requires the data for different purposes. The methods of presentation need not be anticipated in advance, as new ways of presenting data can be added without altering existing presentations, provided the underlying way of storing the data in the computer was well designed in the first place (i.e. the database is 'normalized', to use computer science terminology). In contrast, many creators of large spreadsheets will have found themselves having to laboriously reorganize a spreadsheet, as their project progresses, in response to needs that were not initially anticipated. (Or, alternately, they find themselves using clumsy quick fixes to adapt a spreadsheet to an unanticipated need.) Database systems are also designed to handle efficiently very large quantities of data; unlike spreadsheets which can slow dramatically as the quantity of data increases. Another advantage of databases is the ability of perform sophisticated checks on the validity of new data when it is entered.

Three broad types of checks can be done: (1) domain integrity (i.e. the datum entered has a legal value or form), (2) entity integrity (i.e. a value intended to uniquely identify something has not already been used to identify something else) and (3) referential integrity (e.g. a subject number entered corresponds to a real subject). Finally, a web enabled database delivers instantly the most recent data to a large number of users, in contrast to the clumsiness of distributing new versions of a spreadsheet via email – not to mention the problems that can arise when people are accessing different versions of the spreadsheet, or even worse, entering data into incompatible versions of the spreadsheet.

ProGoSs (Program Goal Progression) is the web enabled database driven system we created to support systematic modelling of degree programs. It provides effective interfaces for users to model their programs, and generates big-picture visualizations that enable users to answer key questions about a curriculum. Figure 1 shows the *ProGoSs* system architecture, which is comprised of three main components: program, goals and progression. The top-half of the figure represents degree programs within institutions. As shown, *ProGoSs* supports multiple institutions; each with multiple degree programs. Each degree program is in turn modelled as a collection of subjects (called units or courses at some universities). For each subject, it is possible to define the pre-requisite knowledge, the intended topics and outcomes, and a collection of assessed activities (e.g. exams, in-class quizzes, take-home projects). Other activities such as lectures or labs could also be modelled in the system; although these were not included as part of this study as we chose to focus on assessed activities only. The bottom-left part of Figure 1 deals with the representation of curriculum guidelines or goals, such as the CS2013.

The bottom-right of Figure 1 shows the use of mastery scales, such as Bloom's Taxonomy, as a way of representing progression. In this study, we chose to use Bloom's Taxonomy, as it has been used widely in computer science education, such as the Australian Computer Society (ACS) Body of Knowledge (Gregor, Konsky, & Wilson 2008). Note, however, that *ProGoSs* is not limited to a specific curriculum definition or a specific mastery scale. We use Bloom's Taxonomy and CS2013 in this study as a validation of *ProGoSs*. Other discipline curricula, or any other list of topics/outcomes/competencies/goals, can be loaded into ProGoSs. Likewise other mastery scale frameworks, such as the SOLO taxonomy (Biggs & Collis, 1982), or an institution's own internally defined scale, could be used instead of Bloom's Taxonomy.

When a subject pre-requisite (or outcome or assessed-activity) is mapped against a curriculum topic, the user also selects a relevant mastery level for that specification. For example, a lecturer may specify that the CS2013 curriculum topic 'Conditional and iterative control structures' Tier-1 Core topic is a pre-requisite for a particular subject, and that students are expected to have a minimum Bloom mastery level of 'Comprehension' for that topic.

Figure 2 shows one part of the data-entry interface. Here a lecturer has defined Question 1.a) of a Final Exam in a Data Structures subject, and has mapped one Key Assessed Topic and two Key Assessed Learning Objectives from the ACM/IEEE CS2013 curriculum guidelines. The sliders on the right are used to set the mastery level for each topic and objective (in this case, Bloom's Taxonomy). The interface is optimised for rapid data entry. A user may simply type a keyword to search through an entire curriculum, and then selecting from among the matches found. Additionally, for each assessment question, the subject lecturer specifies: (1) if the student is familiar with the task (i.e. has

practiced similar examples in lectures/labs); (2) if bare-passing students are expected to be able to answer correctly; and (3) the estimated expected time required to answer the question by bare-passing students vs. top-performing students. This additional metadata is used to distinguish between the aspirational curriculum (i.e. what top-performing students achieve) and the threshold curriculum (i.e. what bare-passing students achieve).

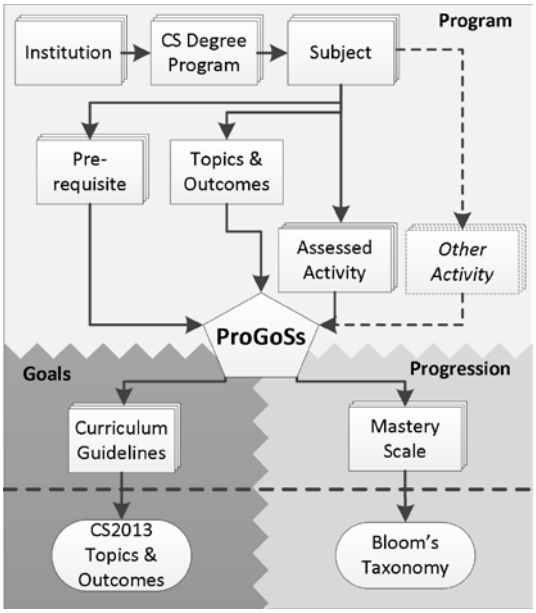

Figure 1: ProGoSs system architecture.

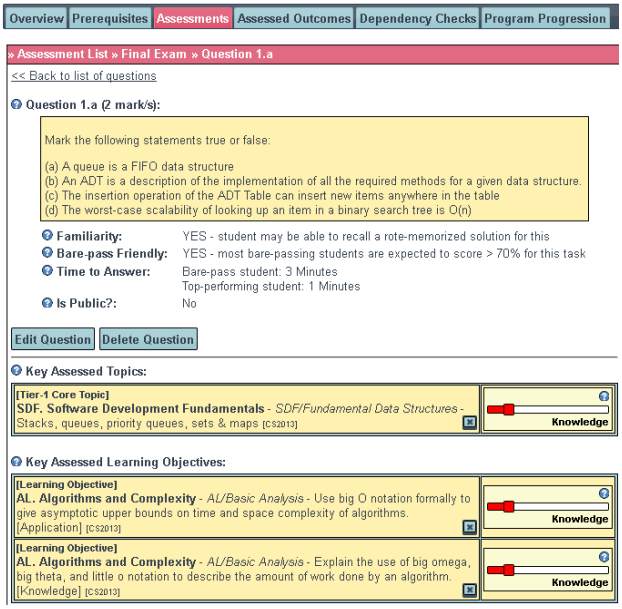

Figure 2: Interface for mapping assessed outcomes

Figure 3 contains a subject-level report that shows the different levels of mastery between bare-passing students (top-left chart) and top-performing students (top-right chart) based on (in this case) Bloom's taxonomy (y-axis). The x-axis represents the percentage-weight of overall subject assessment targeting each mastery level. Figure 3 also shows the bare-pass vs. top-performing Bloom assessment for (in this case) each ACM/IEEE CS2013 topic and objective that is assessed in the subject.

Additional interface screens not shown here allow a user to inspect the dependencies between subjects in terms of the mapped topics and objectives. That is, a subject lecturer is able to immediately identify where each pre-requisite topic or objective is taught and assessed in the degree program. Additionally a subject lecturer is shown which successive subjects rely on the topics and objectives assessed in his or her own course. This provides lecturers with a clear picture of the abilities of students as they progress through the semesters of a degree program.

EVALUATION

We began our evaluation of the *ProGoSs* architecture by translating the Body of Knowledge from the ACM/IEEE CS2013 Strawman curriculum guidelines into our system database. The next step was to define a mastery scale, and so we loaded the six levels of Bloom's Taxonomy into our database. We then used *ProGoSs* to model the core sequence of programming subjects from a computer science degree from one institution. Finally, exam papers from seven computer science subjects were entered into *ProGoSs*. Each exam question was mapped to relevant CS2013 topics and outcomes, at appropriate Bloom's Taxonomy mastery levels. The time taken to model a full exam paper from each subject was between one hour and two hours, with minor additional iterative refinement as required. An exam with 26 plus multiple-choice questions, for example, took longer than an exam with 10 extended questions. The overall time required to map an entire exam paper is not overly demanding, and can be done in a single sitting by a subject lecturer.

EFFECTIVENESS OF CURRICULUM REPORTING INTERFACES

The goal of *ProGoSs* is to facilitate the program design and curriculum review process and to enable stakeholders to easily do the following:

1. Determine overall coverage of core/elective topics and outcomes.

2. Identify where in a program a specific topic/outcome is covered.
3. Identify topics/outcomes that are not covered anywhere in a program.
4. Identify the mastery level at which a topic/outcome is covered in a subject.
5. Differentiate between bare-passing students vs. top-performing students.
6. Inspect program sequence in terms of prerequisite dependencies.

To evaluate the effectiveness of the system for allowing users to answer these questions we invited nine computer science educators to complete a printed questionnaire about *ProGoSs*. The questionnaire guided users through logging into the system and then asked them to answer a series of eleven specific questions by interrogating the *ProGoSs* database. Some of the questions were 'What total percentage of Tier-1 Core topics does our program cover in the top-level Software Development Fundamentals knowledge area', and 'Which subject teaches/assesses the prerequisite topic SDF. Software Development Fundamentals - SDF / Fundamental Programming Concepts / Simple I/O, and at what level of mastery?' Each of the eleven questions mapped to one or more of the six goals above.

The participants comprised of three tutors and six lecturer/professors. Two of the tutors were from the same institution as the subjects modelled in the system. The remaining seven participants were from various other institutions and had limited to no knowledge of the content of the modelled subjects. Some of the nine participants had some previous exposure to *ProGoSs*, but none had used the reporting screens before. All nine participants answered all eleven questions, except for one slip, where the participant's answer revealed that s/he was looking at a neighbouring bar-chart category and hence wrote down an incorrect response.

Participants were not working under any time restriction, nor were they instructed to complete the exercise quickly. Participants took between 15 and 30 minutes (avg. 24 minutes) to learn to use the *ProGoSs* interfaces for the first time and to answer the eleven questions about the modelled curriculum. Thus, the evaluation demonstrated that *ProGoSs* enables users to answer complex questions about the curriculum of a degree program with limited to no prior knowledge of the program content or of the *ProGoSs* interface. It also shows that the *ProGoSs* architecture is an effective approach to fine-grained systematic curriculum design.

Participants were also asked if they knew of any other system or method that would allow them to answer similar questions about their own degree programs. All answered "no". Participants were asked if they would use *ProGoSs* in their own institutions. Eight out of nine said "yes".

CROSS-INSTITUTIONAL PROGRAM COMPARISON

After the above evaluation, three participating computer science educators from institutions other than our own accepted our invitation to enter into *ProGoSs* a zero-prerequisite computer programming subject from their own university. *ProGoSs* was then able to generate comparison reports showing differences in coverage of CS2013 topics/outcomes between any two of those subjects. This is presented as two charts, shown side-by-side, where each chart is like the single chart shown in Figure 4. These charts show the percentage of topics or objectives covered in each Knowledge Area. A user is also able to drill-down for more details, by clicking on appropriate parts of the chart, revealing percentage weights associated with each topic, mastery levels and bare-pass vs. top-performing student expectations.

FACILITATING SYSTEMATIC COMMENTING ON DRAFT CURRICULUM GUIDELINES

While mapping exam questions to topics and outcomes from the CS2013, we encountered a number of cases in which we could not find appropriate matches. For example, an introductory programming subject had a question relating to the 'scope' of a variable. The scope of a variable is a well known concept that is often taught in an introductory programming course. However, the *ProGoSs* interface enables exhaustive keyword searching through a curriculum, and we found that the scope of a variable did not contain that topic. (We also tried commonly used variations on this term, such as 'visibility', without success.)

To capture feedback on the CS2013 proposal itself, such as missing topics, *ProGoSs* includes an integrated form that allows the user to leave comments in each screen that involves mapping topics/outcomes to subject assessment or pre-requisites. Thirteen such comments were captured in the mapping of questions from one Introductory Programming Final Exam alone. These issues are

exportable into a single file. We sent such a file to the CS2013 review committee through appropriate channels. Thus *ProGoSs* is also a useful tool for assessing the relevance and completeness of a draft curriculum specification.

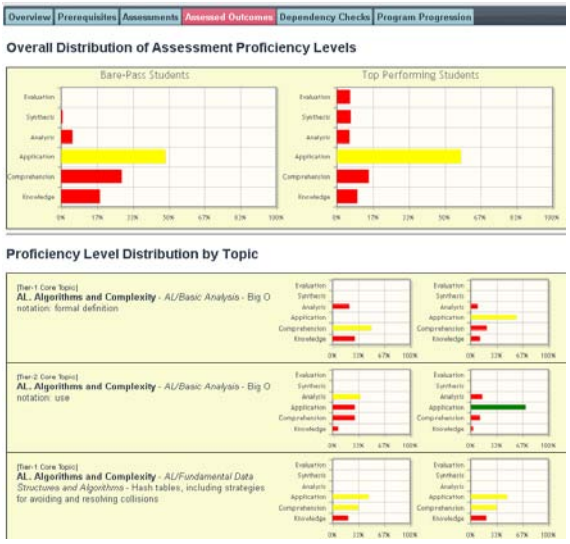

Figure 3: Bare-pass vs. top-performing students

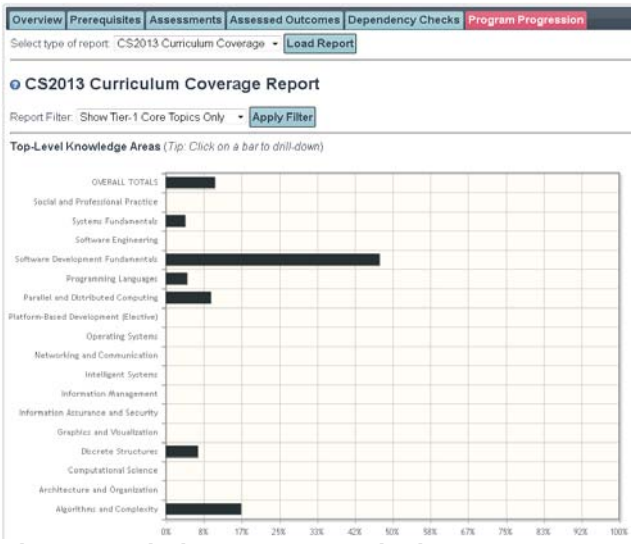

Figure 4: Whole-program curriculum coverage

CONCLUSIONS

In this paper, we have presented the architecture and implementation of our *ProGoSs* system, along with an evaluation demonstrating its effectiveness. *ProGoSs* enables educators to represent a degree program in terms of a given curriculum specification. It enables educators to answer key questions about the mapping between a degree program and a curriculum specification. *ProGoSs* also allows educators from different institutions to systematically compare the teaching and assessment in related subjects. And finally, *ProGoSs* allows a community to systematically provide feedback on a draft curriculum specification. While we evaluated *ProGoSs* in a computer science context, it is easily adaptable to other disciplines.

The tradition of university departments largely setting their own curricula and standards appears to be giving way to an environment of accountability to externally mandated curricula and standards. Many of the difficulties in making this transition have been identified, not least of which is changing the academic culture. However, one difficulty that has not yet received sufficient attention is the difficulty of managing the large amounts of information generating when degree programs are mapped to externally mandated curricula and learning standards. It is not enough to document curricula and document learning standards. In conjunction with those activities, suitable web-enabled database systems will need to be developed to support the mapping between curricula and learning standards. Current projects relying on spreadsheets are already struggling to manage the data generated and we have only just begun the process of generating and handling all this data. Systems like *ProGoSs* are essential for mapping curricula against TEQSA and other learning standards.

REFERENCES

Abunawass, A. (2004). COMPASS: a CS program assessment project. *SIGCSE Bulletin, 36*(3), 269-269.

ALTC. (2010). Engineering and ICT: Learning and teaching academic standards statement. Retrieved September 1, 2012, from http://www.olt.gov.au/resource-engineering-ict-ltas-statement-altc-2010.

Biggs, J. & Collis, K. (1982). *Evaluating the quality of learning: the SOLO taxonomy*, New York: Academic Press.

Bloom, B. S. Engelhart, M. B. Furst, E. J. Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain.* New York: Longmans Green.

Britton, M. Letassy, N. Medina, M., & Er, N. (2008). A curriculum review and mapping process supported by an electronic database system. *American Journal of Pharmaceutical Education*, *72*(5), Article 99. Retrieved September 1, 2012, from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2630156/

Computer Science Curriculum 2008 (CS2008). Retrieved September 1, 2012, from http://www.acm.org/education/curricula/ComputerScience2008.pdf.

Computer Science Curricula 2013: Strawman Draft. Retrieved September 1, 2012, from http://cs2013.org/strawman-draft/cs2013-strawman.pdf

Eisenberg, M. (1984). Microcomputer-based curriculum mapping: A data management approach. Paper presented at *Mid-Year Meeting of the American Society for Information Science*, Bloomington, Indiana.

Gregor, S., von Konsky, B., & Wilson, D. The ICT profession and the ICT body of knowledge (vers. 5.0). Sydney:

Australian Computer Society.
Hausman, J. (1974). Mapping as an approach to curriculum planning. *Curriculum Theory Network, 4*(2/3), 192–198.
Knox, D. (2002). CITIDEL: making resources available. *Proceedings of the 7th annual conference on Innovation and technology in computer science education (ITiCSE 2002)*, (pp. 225-225). Retrieved July 31, 2012, from http://doi.acm.org/10.1145/544414.544493.
Krause, K., Barrie, S., & Scott, G. (2012) Mapping learning and teaching standards in Australian higher education: An issues and options paper. Retrieved September 1, 2012, from http://www.uws.edu.au/__data/assets/pdf_file/0008/294137/KerriLee_website.pdf.
TEQSA, (2011). *Developing a framework for teaching and learning standards in Australian higher education and the role of TEQSA: Discussion paper*. Retrieved September 1, 2012, from http://www.deewr.gov.au/HigherEducation/Policy/teqsa/Documents/Teaching_Learning_Discussion_Paper.pdf, 2012, from http://www.deewr.gov.au/HigherEducation/Policy/teqsa/Documents/Teaching_Learning_Discussion_Paper.pdf
Willett, T. (2008). Current status of curriculum mapping in Canada and the UK. *Medical education, 42*(8), 786–793.