An architecture for systematic tracking of skill and competence level progression in Computer Science

Richard Gluga, Judy Kay, Tim Lever
School of Information Technologies
University of Sydney
Sydney, Australia
richard@gluga.com, judy.kay@sydney.edu.au,
tim.lever@sydney.edu.au

Raymond Lister
School of Software
University of Technology Sydney
Sydney, Australia
raymond@it.uts.edu.au

***Abstract* — A typical Computer Science degree is three to five years long, consists of four to six subjects per semester, and two semesters per year. A student enrolled in such a degree is expected to learn both discipline-specific skills and transferable generic skills. These skills are to be taught in a progressive sequence through the duration of the degree. As the student progresses through the subjects and semesters of a degree, his skill portfolio and competence level for each skill is expected to grow. Effectively modeling these curriculum skills, mapping them to assessment tasks across subjects of a degree, and measuring the progression in learner competence level is, largely, still an unsolved problem. Previous work at this scale is limited. This systematic tracking of skills and competence is crucial for effective quality control and optimization of degree structures. Our main contribution is an architecture for a curriculum information management system to facilitate this systematic tracking of skill and competence level progression in a Computer Science context.**

Keywords-educational technology; outcome-based approach; curriculum; competence level; learning objectives;

I. INTRODUCTION

As part of a university degree, a student is expected to develop a collection of transferable and discipline specific skills. In Computer Science, the relevant discipline-specific skills to be taught are a set of learning objectives specified by curriculum bodies such as the ACM/ACS/IEEE. As an example, the ACM/IEEE Computer Science Curriculum 2008 lists the following learning objectives under the Data Structures knowledge area [1]:

- Describe the representation of numeric and character data.
- Understand how precision and round-off can affect numeric calculations.
- Discuss the use of primitive data types and built-in data structures.
- Describe common applications for each data structure in the topic list.
- Implement the user-defined data structures in a high-level language.
- Compare alternative implementations of data structures with respect to performance.
- Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, and hash tables.
- Compare and contrast the costs and benefits of dynamic and static data structure implementations.
- Choose the appropriate data structure for modeling a given problem.

This list of learning objectives is based on the Bloom Taxonomy, which is a framework for specifying the behavioral sophistication of each objective [6]. A Computer Science student is expected to attain these learning objectives in a progressive sequence. That is, a CS1 student should not be expected to operate at the higher levels (Synthesis or Evaluation) [2].

The collection of generic transferable skills is often referred to as Graduate Attributes and is usually specified internally by each institution [7][8]. As an example, the University of Sydney Faculty of Engineering specifies the following seven high-level Graduate Attributes [3]:

- Design and Problem Solving Skills
- Discipline Specific Expertise
- Fundamentals of Science and Engineering
- Information Skills
- Professional Communication
- Professional Values, Judgment and Conduct
- Teamwork and Project Management

Additionally, these seven attributes contain a maturity dimension, which is described in terms of five levels, numbered from one (least sophisticated) to five (most sophisticated). Each level, for each attribute, has its own description. As an example, the Design and Problem Solving Skills attribute from above has the following level descriptor [3]:

- Level 1 - Ability to analyse standard technical problems and evaluate potential causes and solutions.
- Level 2 - Ability to analyse ambiguous and unfamiliar technical problems with appropriate consideration of assumptions made and their reliability.
- Level 3 - Ability to undertake a major design exercise to achieve a substantial engineering outcome at professional standards, to given specifications.
- Level 4 - Ability to develop creative design solutions for given technical problems.

- Level 5 - Ability to undertake a major design exercise to achieve a substantial engineering outcome at professional standards, with specifications determined by independent analysis of situation and requirements.

A student is expected to attain these graduate attributes progressively over the course of his or her study [4].

The mapping and tracking of these discipline-specific and transferable skills to individual assessment tasks throughout the subjects that form whole of university degrees is currently still an unsolved problem. This mapping is essential for quality control audits, and formal accreditation where applicable, to ensure that a degree has sufficient curriculum coverage. Additionally, this mapping of skills is needed to ensure that content is taught and assessed in an appropriate and effective progressive sequence. Our contribution is an architecture and design for implementing a skill and competence level mapping solution and evaluating it in a Computer Science context.

II. BACKGROUND

Mulder et al. highlight the trend towards competency-based development of curricula in European nations in [11]. The authors discuss the intentions of the ECVET (European Credit System for VET) and the ECTS (European Credit Transfer System in Higher Education) to describe learning outcomes as topologies of knowledge, skills and competencies through the Tuning Educational Structures in Europe project, which "proposes programs based on learning outcomes [that] are described in terms of subject specific and generic competencies, [where] competencies serve as reference points for the design of curricula and evaluation in order to make study programs comparable." Mulder et al. also note that "in June 2002, the Federal Ministry of Education and Research [in Germany] decided to establish national standards of education [but] it remains unclear how these standards can be realized and assessed.". Likewise in Netherlands, "although many institutions claim to have a competence-based curriculum, there is a lot of window-dressing going on, in various cases only superficial changes have taken place and learning processes have not changed".

This highlights the main problem that we are trying to address in this paper. That is, while many learning goal frameworks exist, these learning goals are not being systematically integrated or tracked in university degree programs due to a lack of supporting technology to do so.

Similar trends in standardization and integration of discipline-specific learning goals into higher education degree programs can be found in many other parts of the world as well. The Australian Learning and Teaching Council, for example, is currently in the process of drafting a collection of Threshold Learning Outcomes for each main discipline of study [15], which all Australian universities will need to show compliance with during future accreditation and quality control review processes. The intention of this is to create a standardized framework against which students can be compared to across the country. Similarly, the Tuning-AHELO project is currently drafting a set of similar standards, but at an international level [16].

There is a clear and growing need for technology to support a more principled design of university degree programs, which supports the mapping and tracking of learning and teaching activities to the various institutional, national and international learning goal frameworks, such as those presented above. A large number of educational support systems already exist, and these go by different names such as Learner Management Systems (LMS), Learner Course Management Systems (LCMS), Course Management Systems (CMS), e-Portfolio systems, etc. Moodle (http://moodle.org), for example, is a widely deployed open source CMS. It enables the delivery and assessment of learning content via a web-based interface. While assessment tasks can be mapped to learning outcomes, Moodle operates mostly at the individual course/subject level. It does not support the definition of flexible degree programs that span multiple years, or the visualization of these degree programs in terms of different learning goal frameworks. These same characteristics are shared by other popular LMS/CMS systems such as Blackboard (formerly WebCT) and Sakai.

To address this growing need for big-picture curriculum visualization in terms of different learning goal frameworks, the Faculty of Engineering at the University of Sydney adopted a customized implementation of Curriculum Central [17]. While this system was a step in the right direction, it was limited to a single learning-goal framework. It is becoming increasingly important to be able to show how a degree program complies to multiple frameworks however, and these frameworks vary greatly in the level of granularity (generic transferrable skill statements such as graduate attributes vs. fine-grained discipline specific learning objectives such as the ACM/IEEE CS 2008 curriculum guidelines).

The CUSP system (Course and Unit of Study Portal) has been created to supersede Curriculum Central. It allows degree coordinators and subject lecturers to map and visualize transferable generic skills and accreditation competencies across whole degree programs [4][5]. CUSP captures the representation of multiple sets of graduate attributes and accreditation competencies (named curriculum goals or curriculum goal frameworks) and maps these to the relevant degrees. Each degree structure is modeled into the system as a collection of core subjects plus the rules governing the selection of elective subjects. The high-level skill-mapping design of CUSP is shown in Figure 1.

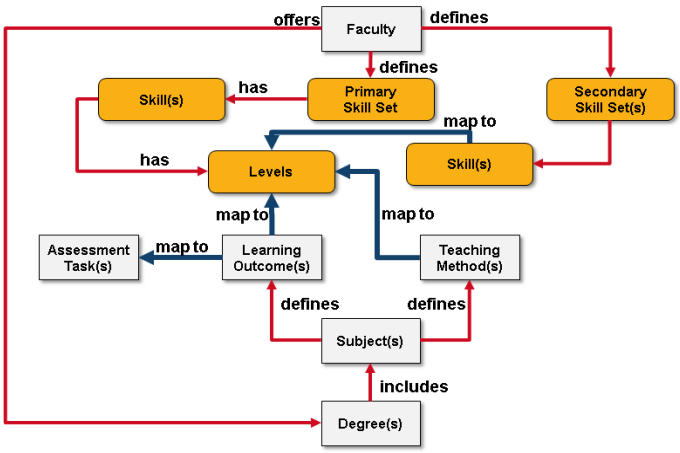

Figure 1 - CUSP high level design

As seen in Figure 1, a primary skill set is selected for each degree, usually the internal Faculty Graduate Attribute Framework. The skill levels from this primary set are then mapped to individual subject learning outcomes, which then map to assessment tasks. Additionally, secondary skill sets are semantically related to those from the primary framework. For example, the *Teamwork and Leadership Skills* goal from the Faculty of Engineering Graduate Attributes policy statement is mapped to semantically similar goals from the Engineering Australia Stage One Accreditation Competencies.

This design enables the CUSP system to generate reports that visualize the curriculum coverage for entire degrees against any of the curriculum goal frameworks attached. These reports in turn enable quick identification of any gaps in goal coverage or any sequencing problems in the degree structure and facilitate accreditation or other quality control review processes. Figure 2 is an example of a chart generated by CUSP which shows the assessment weight associated with each graduate attribute across a full engineering degree. Along the bottom are the seven engineering graduate attributes, and each is broken down into its five constituent levels. Along the vertical is the percentage of assessment weight for the degree as a whole. This chart shows the *Information Skills* attribute is the most under-assessed in this particular degree program, which is valuable information that can be used to optimize the curriculum to comply with accreditation or internal requirements.

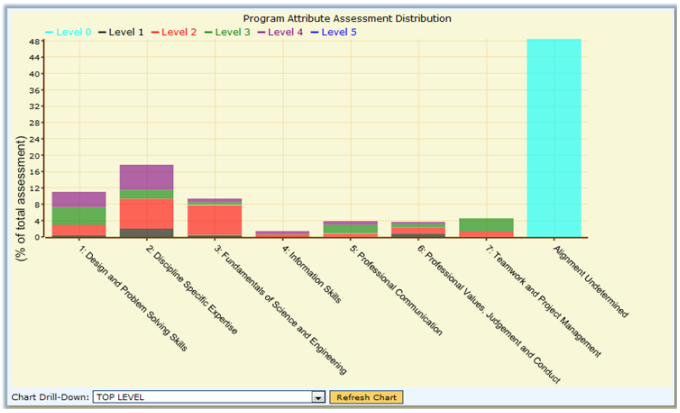

Figure 2 - CUSP attribute coverage chart

To generate these reports, CUSP algorithmically calculates the minimum set of skills that a student can be assessed on based on the elective subject options that they make. That is, if one elective subject has a 50% assessment task associated with, say, *Teamwork and Project Management*, while another elective subject only has a 20% assessment task associated with this skill, CUSP will generate the report based on the latter subject. This ensures that the reports show the 'worst case' scenario for the assessment weight of each skill, or the lowest skill profile possible of a student who was trying to game the elective choices (i.e. a student who is actively selecting elective subjects that have the lowest assessment of a particular skill or attribute).

This 'worst case' report is identified from perspective of the proportionate weighting of the learning goal within program assessment as a whole. This does not necessarily represent the proportionate representation of this goal in the attained competencies of individual students. Indeed, where the proportion of assessment weighting is low for a particular assessment goal, the actual learning achievement may be even lower. Students failing to attain that goal may readily compensate that failure by attainment in other areas that are more heavily weighted. A further 'worst-case' analysis that identifies where the representation of particular learning goals in the attainment of graduating students may be reduced through the process of assessment grading is highly desirable, but not supported by the CUSP system.

As an example, a student who is very poor at *Teamwork and Project Management* could pick the elective subjects with the lowest assessment weight of this skill (e.g. 20% as above), and even if the student scores zero in this assessment but is strong in all other skills, s/he can complete the subject with say a final mark of 80% (distinction). If this student does this consistently across all subjects, s/he can graduate with a distinction average and be regarded as a top-student, while his or her strong weakness in *Teamwork and Project Management* goes by undetected.

It is thus crucial to empower curriculum designers with the systematic tools to model the worst-case scenario not only for the student who games the elective choices, but also for the student who tries to game the assessment tasks. That is, the curriculum information management system should be able to show reports that differentiate between the skill profile of a top-decile student and that of a bare-pass student. This is required to answer important questions such as *"which minimum set of learning objectives from the ACM curriculum are bare-pass students required to complete, and at what level of competence, in order to graduate as computer scientists?"*

Additionally, the CUSP system has highlighted new challenges in specification of the learning goals that are to be attained. The CUSP system provides the framework for an open-ended range of possible ways of conceiving and representing the breakdown and progressive development of key learning goals. The task of determining what sort of goals should be deployed in the first place, and what scale of competence level should be used to measure progression, is left to the curriculum designer. In completing this vital curriculum planning task, designers have a variety of theoretical models to choose from, but so far very little assistance in choosing among them or determining how they might be best applied in practice.

In Computer Science, the ACM uses Bloom's Taxonomy to specify the level of each learning objective [1]. Research in Computer Science education also discusses the use of other progression frameworks including SOLO [9][11] and Neo-Piagetian Theory of Cognitive Development [10]. A curriculum information management system should thus be able to support multiple frameworks of measuring competence level and should support the users in applying these frameworks effectively and consistently.

III. METHOD

We are building a system called ProGoSs (Program Goal Progression), which will enable Computer Science educators to map out the relevant ACM learning objectives to individual

assessment tasks throughout subjects across a degree. The high-level architecture of ProGoSs is shown in Figure 3.

Curriculum Requirements are the collection of syllabus documents from relevant bodies in each discipline. For Computer Science, these would primarily be the ACM/IEEE Curriculum Guidelines as well as internal institutional learning goal statements. These learning requirements drive the design of the Degree structure in each institution, and can be represented as a set of Curriculum Goals that the Degree aims to enable students to achieve. As seen in Figure 3, a Degree structure is a collection of Core and Elective subjects, and each subject is broken down into Entry Requirements, Assessments and Exit Conditions, all of which are mapped against Curriculum Goals and Competence Levels.

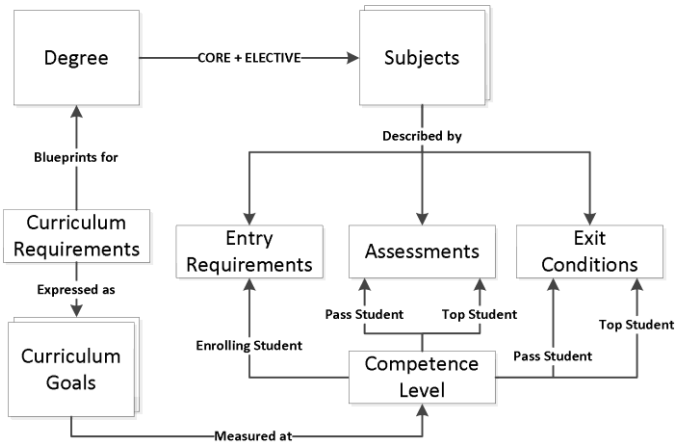

Figure 3 - High Level Architecture

Using Curriculum Goals and Competence Levels as the fundamental glue that links all assessments and subjects together enables the creation of advanced whole-of-degree quality control reporting and curriculum optimization tools. Additionally, by differentiating between bare-pass students and top-decile students in terms of these goals and competence levels allows for the visualization of the overall degree curriculum in terms of both the minimum standards and the aspirational standards.

IV. PROGRESS AND RESULTS

ProGoSs is currently under active development. The first challenge we tackled was the creation of an interactive Bloom Taxonomy tutorial as part of the system. This tutorial was intended to quickly familiarize computer science educators on the use of Bloom to classify programming assessment questions. The tutorial proved successful in this respect. However, the evaluation also revealed some important weaknesses of Bloom when used in this context that require further research [12]. This has led us to think more deeply about the measurement of progression, which appears to be a key trend in numerous active research projects [9][10][11][13].

1. Which framework is most appropriate for specifying the competence level of intended learning outcomes and assessment tasks?
2. Can one framework be used in all contexts, or are multiple frameworks appropriate?
3. How do we ensure that two educators who classify a programming question at, for example, the same Bloom level, both have the same understanding of what that level describes?

A screenshot of this is shown in Figure 4. We are now considering the use of a Naïve Bayes approach [14] to associate a series of weighted evidence or criteria statements with each Bloom (or SOLO, or any other framework), level description. ProGoSs will use these criteria to disambiguate the classification of learning objectives or assessment tasks.

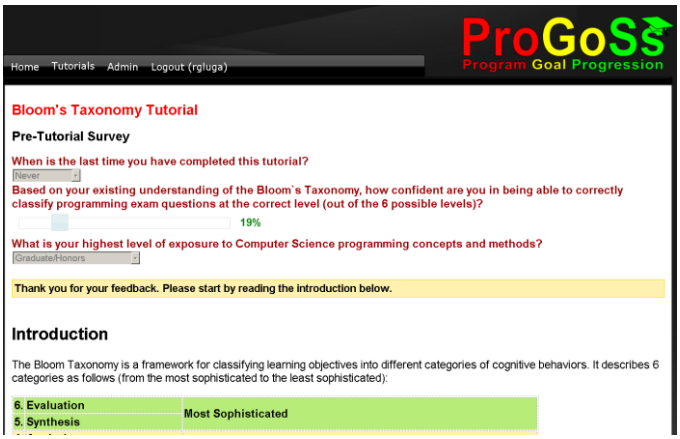

Figure 4 - ProGoSs Bloom Tutorial Screenshot

Once the Naïve Bayes model is discussed and agreed upon by a Faculty or Department, it can be imported into ProGoSs, after which the interface will guide participants in reaching the correct classification through answering or acknowledging the agreed upon criteria items. This generic approach can be applied to any framework of cognitive development, can be used to reach a departmental agreed-upon understanding of the chosen framework, and can be used as a means of documenting the rationale and reasoning in the classification of each learning objective and assessment task.

We have now extended ProGoSs to also offer a Neo-Piagetian Theory of Cognitive Development [10] tutorial, as an alternative framework to Bloom, for use in measuring learning progression, which is in the process of being evaluated. These tutorials are made freely available as part of the live ProGoSs deployment at http://progoss.com.

The next phase in ProGoSs development is to enhance our cognitive framework tutorial structures to further enhance classification disambiguation (even with the training received from the tutorials, computer science educators still do not reach unanimous agreement as to which, say, Bloom level a particular assessment question maps to [12]). Following this, we are mapping actual computer science degrees and assessments from a range of Australian universities to ACM learning objectives at the pass vs. top-decile levels.

These capabilities, which were not previously possible in CUSP, are made possible by the new ProGoSs architecture. Namely, CUSP was restricted to tagging generic skills and

higher-level accreditation skills to assessment tasks. ProGoSs enhances this approach to allow mapping of fine-grained discipline-specific skills such as the ACM/IEEE learning objectives, and further allows for differentiation between the minimal expectations of pass students vs. the aspirational expectations of top performing students. In order to support this differentiation, ProGoSs must support principled and systematic methods of measuring progression and competence levels, hence the inclusion of interactive contextualized tutorials on learning taxonomies such as Bloom and Neo-Piagetian Theory of Cognitive Development.

V. CONCLUSION

University degrees teach a combination of transferable and discipline specific skills over a period of three to five years. These curriculum goals can be used as a glue to link together all subjects and assessment tasks in degree sequences. The motivation for doing this is to track student progression and maturity in a systematic way that enables advanced reporting tools for use in quality control and curriculum optimization. Additionally, this approach can be used to discern between the intended learning outcomes of a bare-pass student vs. that of a top-decile student in the curriculum design of a degree.

Current research at this scale is limited. We initially developed CUSP to deal with transferable skills, but this did not scale well to discipline specific skills. CUSP also could not discern between different student performance profiles. Our ProGoSs system architecture is designed to specifically address these problems, and is actively being developed and evaluated in a Computer Science context. Initial results in the use of Bloom's Taxonomy for classifying programming exam questions indicate that we can help people to quickly learn to use Bloom with moderate reliability but that there are some serious limitations in using Bloom in this way. This is particularly important given the key role that Bloom plays in defining curricula like the current and planned ACM/IEEE CS curriculum guidelines. Upcoming evaluations in the use of other competence level measurement frameworks and in the mapping of actual Computer Science degree structures are expected to provide insights into alternatives that may be easier to use in supporting more effective design of the Computer Science curriculum.

Our key contributions are in the exploration of principled approaches to formulating curricula so that the long term learning over a full degree program is more effectively planned and monitored; this will mean that students can be assured a coherent series of learning experiences that build to achieve the key learning goals.

A deployment of the ProGoSs system is being made publicly available at http://progoss.com. This will allow Computer Science educators from around the world to use our Bloom and Neo-Piagetian tutorials, to comment on the use of these frameworks in a programming context, and to develop more principled and systematic design of teaching and learning in Computer Science.

ACKNOWLEDGMENT

We would like to thank the Smart Services CRC who is partially sponsoring this project and all our colleagues that we have collaborated with throughout the project.

REFERENCES

[1] ACM Computer Science Curriculum 2008, http://www.acm.org/education/curricula/ComputerScience2008.pdf

[2] Lister, R., (2001) Objectives and Objective Assessment in CS1. Proc. SIGCSE Technical Symposium on Computer Science Education, Charlotte NC, USA, 292-296, ACM Press.

[3] Engineering & IT Graduate Attribute Matrix, University of Sydney, http://cusp.sydney.edu.au/attributes/view-attribute-set-pdf/competency_set_id/20/ [Last Accessed - Sep 15 2011]

[4] Gluga, R., Kay, J., and Lever, T (2010). Modeling long term learning of generic skills. In V. Aleven, J. Kay, and J. Mostow, editors, ITS2010, Proceedings of the Tenth International Conference on Intelligent Tutoring Systems, pages 85-94. Springer, 2010.

[5] Gluga, R., and Kay, J. (2009) Largescale, long-term learner models supporting flexible curriculum definition. In Proceedings of the Workshop on Scalability Issues in AIED, held in conjunction with AIED2009, pages 10-19, 2009.

[6] Bloom, B.S. (Ed.) (1956) Taxonomy of Educational Objectives: Handbook I: Cognitive Domain, Longmans, Green and Company.

[7] Barrie, S. C. (2004). A research-based approach to generic graduate attributes policy. Higher Education Research & Development. Vol. 23, No. 3, 2004.

[8] Barrie, S. C. (2007). A conceptual framework for the teaching and learning of generic graduate attributes. Studies in Higher Education, Vol 32, No. 4, pp. 439-458, 2007.

[9] Sheard, J., Carbone, A., Lister, R. and Simon, B. (2008), Going SOLO to assess noice programmers. Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE2008), pp. 209-213, Vol. 40.

[10] Lister, R. (2011). Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer. Australasian Computing Education Conference (ACE2011). Pp. 9-14. Vol. 114.

[11] Brabrand, C. and Dahl, B. (2009). Using the SOLO taxonomy to analyze competence progression of university science curricula. Journal of Higher Education. Vol 58. No. 4. Pp: 531-549.

[12] Gluga. R., Kay. J., Lister, R., Kleitman. S., Lever. T. (2012). Coming to terms with Bloom: an online tutorial for teachers of programming fundamentals. Proc. Fourteenth Australasian Computing Education Conference (ACE2012), Melbourne.

[13] Oliver, D., Dobele, T., Greber, Tm., Roberts, T. (2004). This course has a Bloom Rating of 3.9. Proceedings of the sixth conference on Australasian computing education. Vol. 30

[14] Shinghal, R. (1992). Plausible Reasoning in Expert Systems. Formal concepts in artifical intelligence: findamental. London; New York : Chapman & Hall, 1992

[15] A. Learning and T. Council, "Engineering and ICT learning and teaching academic standards statement,"http://www.altc.edu.au/system/files/altc standards ENGINEERING 090211.December 2010.

[16] A. C. for Educational Research (ACER), "Assessment of higher education learning outcomes (AHELO)," http://www.acer.edu.au/aheloau.

[17] Calvo R, Carroll N, Ellis R. Curriculum Central: A portal system for the academic enterprise. International Journal of Continuing Engineering Education and Life-Long Learning. 2007 ;17(1):43-56.