So, H. J. et al. (Eds.) (2020). Proceedings of the 28th International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

A Unified Approach for Analysing Computer Science Courses: An Australian Case Study

Sameera JAYARATNA[a*], Caslon CHUA[a], Mohommed Eunus ALI[b] & Timos SELLIS[a]
[a]*Swinburne University of Technology, Australia*
[b]*Bangladesh University of Engineering and Technology, Bangladesh*
*mjayaratna@swin.edu.au

Abstract: A large variety of Computer Science (CS) and Information and Communications Technology (ICT) programs are offered in different institutions across Australia. Even the same institution has several majors of CS programs due to its recent popularity among students and demand in the job market. Current practices in CS education lack a unified approach to analyse and compare these programs, and consequently cannot assess the students in different programs based on a common platform. In this paper, we propose a unified and systematic approach to analyse CS units and courses. Our approach is based on CS curriculum guidelines, according to which each subject (or unit) of a program (or course) can be mapped to knowledge areas. The insights gained through the proposed unified approach could help education providers, students, and governing bodies to understand the fundamentals of CS and ICT programs. A case study involving data from CS programs offered by two Australian higher education institutions show the efficacy of our proposed approach.

Keywords: Computer science education, course analysis, course comparison

1. Introduction

Computer Science education is increasingly becoming popular across the globe. According to the Australian Computer Society (ACS), the annual enrolments in Computer Science (CS)/Information and Communications Technology (ICT) related programs at universities in Australia are 50 percent higher than a decade ago (ACS & Deloitte Access Economics, 2019). With the increasing demand, most higher education providers offer variants of CS/ICT programs. Currently, around 40 education providers, with a total of 340 CS/ICT programs, hold ACS accreditation (Grayston, 2019).

A *course* refers to a program of study such as Bachelor of CS or Bachelor of ICT. Courses offered in both undergraduate and postgraduate levels by different institutions tend to differ in terms of structure, the number of units taught, and the topics covered. A *unit* refers to any subject such as Introduction to Programming or Software Engineering, offered under a course. It is unlikely to encounter courses in two different institutions that are identical in terms of the units they are composed of. Even the units with similar names offered by different institutions may not be similar in terms of the topic coverage (the number of topics and the breadth of each topic covered in the unit). Thus, a significant variation can be observed among courses in the same discipline. Also, CS curricula change over time due to the rapid change of technology.

The ability to effectively analyse multiple courses is paramount to all the participants of CS education: education providers, prospective students, current students, industry professionals, and professional bodies. For education providers, course analysis is vital to assess their courses according to design guidelines and compare against other similar courses. The prospective students can benefit from an effective comparison of the differences in CS programs when making choices. Students who are currently enrolled in a CS program can benefit from an effective analysis of units offered in the program. For instance, an effective comparison of units available as elective units of the program can be vital for a student's overall performance. The industry professionals' ability to understand how a CS program would cater to their requirements may enable the selection of the most suitable graduates. Professional bodies such as ACS can gain insights from effective course analysis to ensure the quality of CS courses offered by education providers. Current practices in CS education lack a unified approach

to analyse and compare CS programs. Consequently, cannot assess the students in different programs uniformly. To address the above issues, we propose a unified approach to analyse CS courses and units.

Our approach is motivated by the curriculum guidelines for CS programs published by the Association for Computing Machinery (ACM), IEEE Computer Society, and ACS. These curriculum guidelines fundamentally use two components to describe the content of a CS unit: Body of Knowledge (BOK) and Learning Taxonomy. ACM and IEEE jointly published curriculum guidelines (Sahami et al., 2013), which discuss the BOK relevant for undergraduate programs in computing. Similarly, ACS published Core Body of Knowledge (CBOK) for ICT professionals (ACS, 2015). These curriculum guidelines recommend a unit to cover the knowledge areas defined in a BOK at appropriate levels of difficulty (or mastery) defined by a learning taxonomy. These organisations have also recommended learning taxonomies, like Bloom's taxonomy (Bloom, 1956) and Bigg's SOLO taxonomy (Biggs & Collis, 2014). Most CS programs are designed according to a set of curriculum guidelines, such as those stated above. Therefore, the coverage of a unit can be mapped to a BOK and a learning taxonomy. In this study, we present a unified representation of units that can be adopted for any given BOK and learning taxonomy combination. For instance, Australian CS programs follow ACS guidelines, which involve ACS CBOK and Bloom's taxonomy. Thus, the proposed approach enables all CS units and courses that follow ACS guidelines to be uniformly assessed using ACS CBOK and Bloom's taxonomy.

In this paper, we propose a unified approach for analysing and comparing CS units and courses mapped to a BOK with respective levels of difficulty defined in a learning taxonomy. First, we explain two fundamental elements, namely: BOK and learning taxonomy, in representing the units. Then, we explore how the stakeholders of CS education, such as educators and students, can gain valuable insights on units and courses using the proposed unit representation. We demonstrate the applicability of the proposed approach via an Australian case study. CS programs in Australia follow ACS curriculum guidelines based on ACS CBOK and Bloom's taxonomy. Therefore, we adopt the proposed approach to the Australian context by using the ACS CBOK and Bloom's taxonomy as the instances of BOK and learning taxonomy, respectively. The dataset used is generated by universities as part of the ACS course accreditation process. The availability and the quality of the dataset facilitate the immediate adoption of the proposed approach in any institution that follows ACS guidelines. The contribution of this paper can be highlighted in three aspects: (i) we represent CS units based on knowledge area coverage, (ii) we propose a novel unified approach to analyse CS courses and units based on the unified unit representation, and (iii) we present a case study with a dataset obtained from Australian universities to show the applicability of the proposed approach.

The next section briefly reviews the related work. Then, Section 3 explains how units are represented based on BOK and learning taxonomy. Section 4 presents the case study, which demonstrates how the unified representation discussed in Section 3 can be used to analyse courses and units. Finally, Section 5 draws conclusions from the case study performed and discusses the potential extensions of the proposed approach in future work.

2. Related Work

Many studies have discussed methodologies for analysing CS course curricula to provide insights to educational providers (Méndez, Ochoa, & Chiluiza, 2014; Oliver, Dobele, Greber, & Roberts, 2004; Pedroni, Oriol, & Meyer, 2007; Sekiya, Matsuda, & Yamaguchi, 2015). Méndez et al. (2014) presented a set of techniques that relies on historical academic data for various applications like estimating course difficulty and identifying dropout paths. The results were then used to gather recommendations for curriculum design. Pedroni et al. (2007) proposed a more systematic approach to define courses intuitively. However, their approach requires teachers to define courses with units of knowledge, which takes a lot of time and effort. Sekiya et al. (2015) compare CS-related courses offered across universities using natural language processing to analyse how a course curriculum is distributed over knowledge areas defined by ACM & IEEE. However, it does not capture the depth in which a course covers the knowledge areas. In contrast, Oliver et al. (2004) analyse the cognitive difficulty of different courses offered by a university using Bloom's taxonomy while ignoring the aspect of knowledge areas.

Learning taxonomies such as Bloom's taxonomy and Bigg's SOLO taxonomy are used in computing education (Fuller et al., 2007; Shuhidan, Hamilton, & D'Souza, 2009). They are often used

to indicate different stages of learning development, which is useful to distinguish the appropriateness of learning outcomes at different levels in courses. Bloom's taxonomy appears to be the most widely used taxonomy to state learning goals in computing studies (Johnson & Fuller, 2006; Masapanta-Carrión & Velázquez-Iturbide, 2018). This taxonomy is used as a reference for the classification of knowledge and competencies to be acquired via an education module. Bloom's taxonomy has also inspired studies on course comparison in areas of education other than CS (Hoffmann, 2008).

3. Unified Representation of Units

In this section, we first describe the fundamental elements used to describe the content of a CS unit: BOK and Learning Taxonomy. Then, we discuss how to represent a CS unit based on these elements.

3.1 Body of Knowledge and Learning Taxonomy

Multiple organisations have produced documents describing BOK for CS-related disciplines. IEEE computer society published Software Engineering Body of Knowledge (SWEBOK) (Bourque & Fairley, 2014), while ACM and IEEE Computer Society jointly published curriculum guidelines (Sekiya et al., 2015) discussing the BOK relevant for undergraduate programs in computing. Similarly, ACS published CBOK for ICT professionals (ACS, 2015).

The ACM and IEEE computer society's document on curriculum guidelines (Sahami et al., 2013) has organized the BOK into a set of 18 knowledge areas corresponding to topical areas in computing. These knowledge areas include software engineering, operating systems, and programming languages (Sekiya et al., 2015). Each knowledge area is further organized into a set of knowledge units. The guidelines illustrate how knowledge areas may be covered and combined in courses. As guidance, each knowledge unit lists a set of topics and learning outcomes students are expected to achieve. Each learning outcome is associated with a level of difficulty. This document defines three difficulty levels based mainly on Bloom's taxonomy. The levels of difficulty are defined as familiarity, usage, and assessment. In summary, these curriculum guidelines provide a framework for universities to map the courses they offer to either knowledge areas or knowledge units with the respective levels of difficulty.

ACS provides guidelines for course design using the CBOK for ICT professionals (ACS, 2015). ACS has incorporated these guidelines into their course accreditation process to ensure the quality of computing education in Australia (ACS, 2020). The CBOK introduces 5 knowledge areas: ICT Problem Solving, ICT Professional Knowledge, Technology Resources, Technology Building, and ICT Management (ACS, 2015), which are further divided into topics. ACS accreditation process requires the coverage of these knowledge areas by each unit of a course to be captured according to the ACS guidelines. Like ACM, ACS mainly utilises Bloom's taxonomy to indicate levels of difficulty (ACS, 2020). ACS also allows the use of alternative indicators such as Biggs SOLO taxonomy. As per the ACS guidelines, a unit is designed such that each unit activity covers a set of knowledge areas at specific levels of difficulty. Unit activities may include lectures, tutorials, and exams. In summary, ACS provides a framework to map a unit's activities to knowledge areas with respective levels of difficulty.

The main difference between ACS guidelines and ACM guidelines is the structure of the BOKs. In addition, ACS guidelines capture a unit's coverage in terms of unit activities while ACM guidelines capture the same in terms of a unit's learning outcomes. Unit activities and learning outcomes may be considered as two types of unit components. Thus, in summary, both guidelines map *unit components* (learning outcomes or activities) to a BOK with respective levels of difficulty defined in a learning taxonomy. Accordingly, we describe a unified representation of units in the next section.

3.2 Unit Representation

Units can be considered as the building blocks of a course. Therefore, we first investigate how the coverage of a unit is presented against a BOK, which defines a set of Knowledge Areas (KA). A unit can be described using a set of unit components, which may be either a set of learning outcomes or a set of activities (e.g., tutorials, assignments, exams). Each component of a unit can be mapped to the

defined KAs with respective levels of difficulty defined in a learning taxonomy. Each component of a unit covers one or more KAs.

In this study, a unit is represented based on the summarised levels of difficulty across all the unit components per KA. A unit's overall coverage of a KA is indicated by the *Difficulty Rating* of that KA. Given a set of *n* KAs and a unit *X* with *c* components, the *Difficulty Rating* of the i^{th} KA, $DR_{i,X}$ is calculated using the level of difficulty at which each component of unit *X* covers that KA.

$$DR_{i,X} = median(KA_{i,X,1}, KA_{i,X,2}, \dots, KA_{i,X,c}) \quad \text{(Eq. 1)}$$

where $KA_{i,X,c}$ refers to the difficulty level at which the c^{th} component of unit *X* covers the i^{th} KA. Only the components that evaluate a given KA are used to calculate the median. In the case of an even number of components, the lower of the two middle values will be taken as the median, because difficulty levels defined in learning taxonomies are of ordinal scale. Then, unit *X* is presented as an n-tuple.

$$Unit\_X = < DR_{1,X}, DR_{2,X} \dots, DR_{n,X} > \quad \text{(Eq. 2)}$$

Table 1 presents the coverage of a sample unit *Q* with three components against a given BOK with five KAs. Assume the learning taxonomy used defines levels of difficulty as *L1*, *L2*, and *L3*, where *L1* is the least difficult and *L3* the most. The value in each cell indicates the level of difficulty at which the respective component evaluates a KA. An empty cell indicates that the component does not cover the respective KA. Based on Eq. 1, Table 1 presents the *Difficulty Ratings* of unit *Q* in the last column. Then, unit *Q* can be represented as a 5-tuple: $< L3, L3, L2, L1, L2 >$ according to Eq. 2.

Table 1.*Coverage and Difficulty Rating of Unit Q*

Knowledge Area	Unit Components			Difficulty Rating
	$Comp_1$	$Comp_2$	$Comp_3$	
KA_1	*L1*	*L3*		*L3*
KA_2	*L2*	*L3*	*L3*	*L3*
KA_3		*L2*		*L2*
KA_4	*L1*		*L1*	*L1*
KA_5	*L3*	*L3*	*L2*	*L3*

4. Case Study

In this paper, we present a case study set in the Australian context. Therefore, we implement the unit representation defined above based on the data from Australian universities. As mentioned in Section 3.1, Australian universities follow the ACS course accreditation guidelines to design their computing courses. The accreditation process requires a university to submit a document indicating how units of a CS course covers the CBOK. Therefore, each CS program that applies for ACS accreditation maps each unit's components to KAs defined in the CBOK with respective levels of difficulty. ACS uses Bloom's levels for indicating the levels of difficulty. In this section, we first describe two fundamental elements: ACS CBOK and Bloom's levels, which indicate levels of difficulty. Then, based on the CBOK and Bloom's level, we represent units used for measuring and comparing courses.

4.1 ACS Core Body of Knowledge

In this section, we discuss the specifics of ACS CBOK, because while our work can be applied to CS courses based on any BOK, it is primarily focused on the Australian context. ACS accreditation guidelines indicate 5 knowledge areas which are further divided into 19 topics. In this paper, we refer to the 5 knowledge areas as *'Categories'* and 19 topics as *'Knowledge Areas (KA)'* to improve readability. Table 2 presents the categories and KAs under each category. An Australian university program should provide appropriate coverage of the KAs defined by the ACS CBOK to receive course accreditation (ACS, 2015). A program consists of multiple units, and a unit is designed with multiple

activities (e.g., assignments, tutorials, exams). Each activity assesses one or more KA at varying levels of difficulty. ACS recommends Bloom's levels as the indicator of difficulty levels.

Table 2. *ACS CBOK Knowledge Areas*

Category	**Knowledge Area**
ICT Problem Solving	1. Abstraction
	2. Design
ICT Professional Knowledge	3. Ethics
	4. Professional expectations
	5. Teamwork concepts & issues
	6. Interpersonal communications
	7. Societal issues
	8. Understand of ICT profession
Technology Resources	9. Hardware & software fundamentals
	10. Data & information management
	11. Networking
Technology Building	12. Programming
	13. Human factors
	14. Systems development
	15. Systems acquisition
ICT Management	16. IT governance & organisational issues
	17. IT project management
	18. Service management
	19. Security management

4.2 Bloom's Taxonomy

Bloom's taxonomy is a well-known learning taxonomy from pedagogy. The learning taxonomy devised by Bloom divides the cognitive aspects of learning into six hierarchical levels, starting from the simplest to the most complex. Here, 'the levels' indicate the levels of difficulty. That is, the first ones must typically be mastered before the next one can take place. Bloom's levels are presented in Table 3. The model was revised in 2001 by Anderson and a team of cognitive psychologists (Anderson et al., 2001). Bloom's taxonomy appears to be the most widely used taxonomy to state learning goals in computing studies (Johnson & Fuller, 2006; Masapanta-Carrión & Velázquez-Iturbide, 2018). Even though there are concerns about the appropriateness of Bloom's taxonomy in CS education, they are mostly about the difficulties of usage (Masapanta-Carrión & Velázquez-Iturbide, 2018). Studies also focus on ways to improve the understanding of educators to ensure the effective use of Bloom's taxonomy (Richard, Judy, Raymond, Simon, & Sabina, 2013; Starr, Manaris, & Stalvey, 2008).

Table 3. *Bloom's Taxonomy*

	Level	**Description**
1	Knowledge	Recall facts and basic concepts
2	Comprehension	Explain ideas or concepts
3	Application	Use information in new situations
4	Analysis	Draw connections among ideas
5	Synthesis	Justify a stand or decision
6	Evaluation	Produce new or original work

4.3 Data Set

The data analysed in this article are collected from two higher education institutions in Australia. The composition of the data set is shown in Table 4. Institution A offers a Bachelor of Computer Science course with multiple majors, including Cybersecurity, Datascience, Networking, and Software

development. Curriculum for each major (A1-A4) consists of 8 core units, 8 major-specific units, and 8 electives. The core units are common to all the majors. Bachelor of Networking course from institution B (B1) consists of 21 core units and 3 electives. Like Institution A, Institution B allows the electives to be either CS or non-CS units. Non-CS units (e.g., Introduction to Management) offered by non-CS departments do not have ACS CBOK mapping. However, the enrolments of such units can be considered low compared to the enrolments of CS units by the students. The units in the dataset use multiple activities (components) to assess the knowledge of CS students. Table 5 presents one of the units used in this study. The value in each cell indicates the Bloom's level at which the respective activity evaluates a KA. An empty cell indicates that the activity does not cover the respective KA.

Table 4. *Data Set*

ID	**Institution**	**Course Name**	**Major**
A1	Institution A	Bachelor of Computer Science (UG-CS1)	Cybersecurity
A2			Datascience
A3			Networking
A4			Software development
B1	Institution B	Bachelor of Networking (UG-CS2)	-

Table 5. *Coverage and Difficulty Rating of a Unit*

		Knowledge Areas																	
		Abstraction	Design	Ethics	Prof. expectations	Teamwork concepts	Interpersonal comm.	Societal issues	Understand of ICT prof.	Hardware & software	Data & info. management	Networking	Programming	Human factors	Systems Development	Systems acquisition	IT gov. & org. issues	IT project management	Service management	Security management
Unit Components	Lectures	2	2							3	3		3							
	Assignment	3	3							3	3									
	Tutorials	3	3							3	3		3							
	Exam	3	3							3	3									
Difficulty Rating		3	3							3	3		3							

4.4 Unit Analysis

The units offered by a CS program tend to cover combinations of KAs at various levels of difficulty. A unit would typically have one or more common KAs with another unit of the same CS program. However, a unit description is often textual and includes topics covered, learning outcomes, and assessment tasks. The comparison of textual descriptions to understand similarities or dissimilarities between units can be tedious and time-consuming. In this section, we demonstrate unit analysis from two perspectives: a current student's perspective and an education provider's perspective.

First, we look at the scenario of a current student. A student is currently studying a CS program with several elective units. She intends to select either Programming Fundamentals unit or Mobile Programming unit as one of the electives. She needs to understand the differences between the units to make an informed selection. Figure 1 presents the comparison of units based on their coverage of KAs. As per Figure 1, Programming Fundamentals unit covers a larger number of KAs (11) than Mobile Programming (7). More specifically, the Mobile Programming unit covers the Networking KA while the other unit does not. Also, activities of Mobile Programming unit assess students in Data & information management KA at a higher level (4) than that of Programming Fundamentals unit (2). Accordingly, the student may select a unit that aligns with her preferences.

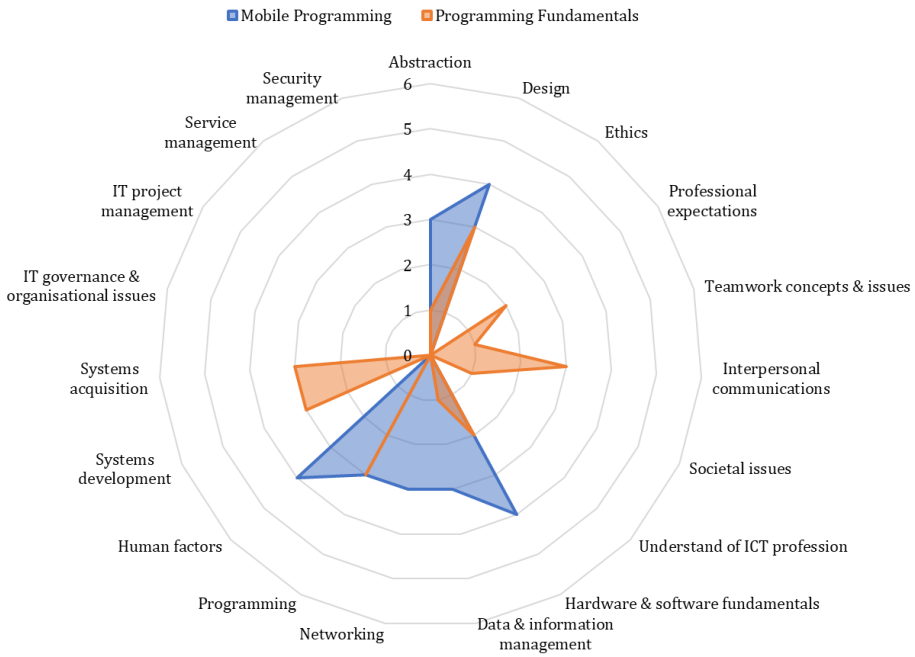

Figure 1. Comparison of units.

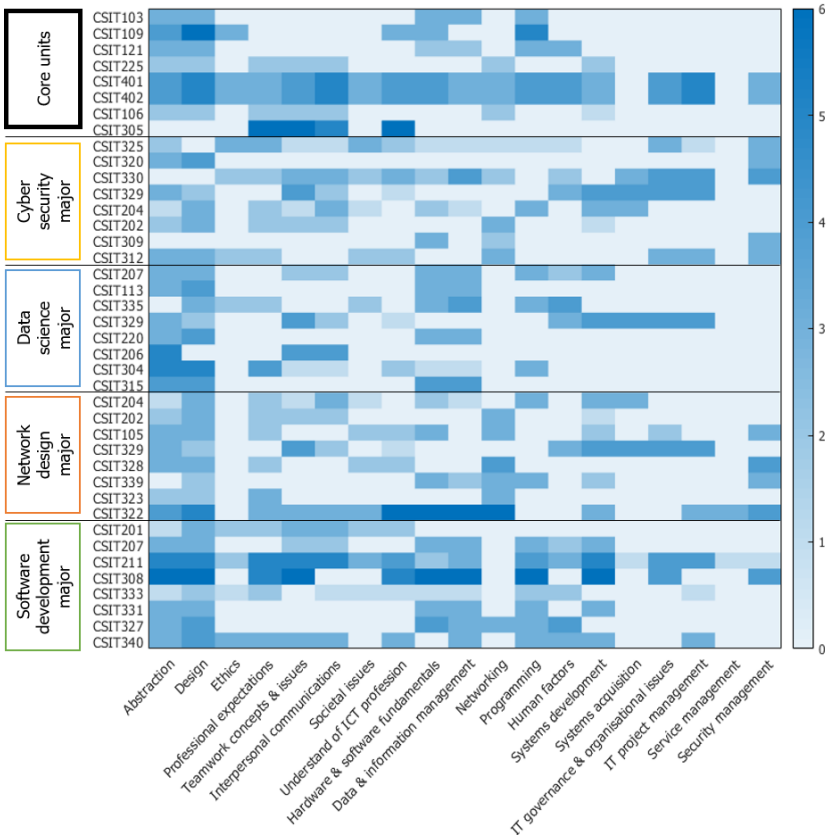

Figure 2. Spread of units over Knowledge Areas.

Next, we consider a scenario of an education provider. An education provider is offering four majors under a CS program. Course designers are expected to update the program for which they need to review the current course composition. Figure 2 shows the spread of all units offered as core units, which are common to all majors and major-specific units for each major (A1-A4) over 19 KAs. The values of cells indicate the Difficulty Rating of units calculated as per Section 3.2. The values are indicated in a colour scale ranging from light blue to dark blue as they vary from 1 to 6. This visualization helps the course designers gain insights into KA coverage of units. For example, one can

observe that almost all the units cover the *Problem Solving* KA category. In addition, Security Management KA is covered not only by security-related units but also by networking related units. Interestingly, there is a lack of units that cover *Service Management* KA. Such insights may help course designers identify gaps in the program that can be filled by introducing new units.

4.5 Course Analysis

Systematic representation of courses can be beneficial for many parties, including education providers, prospective students, and professional bodies. For instance, universities can gain insights into courses offered by them. Prospective students get the ability to make an informed decision when choosing a course. The overwhelming process of browsing numerous course curricula may be avoided by using a summary of vital information. Additionally, professional bodies such as ACS can gain insights that can contribute to shaping the overall landscape of CS education by assessing all the courses through a systematic representation. In this section, we demonstrate the benefits of course analysis from a prospective student's perspective and an employer's perspective. For each perspective, we illustrate a scenario based on the dataset shown in Table 4.

First, we look at the scenario of a prospective student. A student interested in studying computing has selected a university which offers multiple majors under its Bachelor of Computer Science program. The student has a preliminary understanding of the domains that would provide future job opportunities. Accordingly, he is interested in Datascience, Cybersecurity, and Software development majors. He needs to understand the differences between the majors to select the suitable major for him. Figure 3 uses a box and whisker plot to summarise the units of each major across the 19 KAs. Each major is analysed based on the set of core and major-specific units. This plot indicates the minimum, maximum, and median Difficulty Ratings of each major. Figure 3 indicates that the majors are relatively different in terms of their coverage of KAs. This student is particularly interested in the Security management KA and Programming KA of the majors. He would like to gain significant exposure to Security management KA and extensive exposure to Programming KA. As per Figure 3, Security management KA is covered by each of the three majors. Units of Datascience and Cybersecurity majors are concentrated on Difficulty Rating 3 in Security management KA. However, Figure 2 indicates that in contrast to the Cybersecurity major, the Datascience major does not cover Security management KA in any major-specific units. The units of Software development major are distributed between Difficulty Ratings of 1 and 4 in Security management KA. Also, Figure 2 indicates that the Software development major includes a few major-specific units that cover Security management KA. As per the analysis, the Software development major offers significant exposure to Security management KA while the Cybersecurity major provides extensive exposure. Furthermore, Figure 3 indicates that Programming KA is covered by each major, and 50% of units in each major have a Difficulty Rating of 3 or 4. Also, units of Datascience major and Cybersecurity major appear to be similarly distributed in Programming KA with a minimum Difficulty Rating of 3. In contrast, the minimum Difficulty Rating within Software Development major units is 2. According to Figure 2, most of the units in Software Development major cover Programming KA. Therefore, it is evident that the Software Development major would offer the most exposure to Programming KA in comparison to the other two majors. As per the above findings, the Software development major would best cater to the student's requirements regarding Security management KA and Programming KA.

Next, we look at the scenario of an employer. An industry professional is looking to employ networking graduates according to specific criteria. She wants to find the CS program that is most likely to produce graduates fitting her criteria. Programs under consideration are UG-CS2 program of Institution B and UG-CS1 program of Institution A. The KAs of most interest in her criteria include Networking and Interpersonal communication. The industry professional can utilise Figure 4 to make an informed choice between the programs. Significant differences can be observed between the courses, even though both are specialised in networking. Regarding the coverage of Networking KA, the maximum Difficulty Rating of UG-CS2 is higher at 6, while the value of Institute A's UG-CS1 is only at 4. Also, the minimum Difficulty Rating of UG-CS2 is higher at 3, while the value of UG-CS1 is only at 2. Also, the inter-quartile range of UG-CS2 is larger than that of UG-CS1. Therefore, we can observe that the coverage of Networking KA is consistently higher in Institute B's UG-CS2 than in Institute A's UG-CS1. The minimum (2) and maximum (5) Difficulty Ratings of Interpersonal communication KA are similar in both programs. However, 50% of the units in UG-CS2 cover the Interpersonal

communication KA at Difficulty Ratings of 3 or 4. Accordingly, the employer may consider the UG-CS2 program better with regards to Networking KA and Interpersonal communication KA. Similarly, the employer can compare the two programs based on their coverage of KAs relevant to her criteria.

In summary, the course analysis showed the ability to capture the contrast in CS programs through visualizations. It demonstrated how stakeholders, such as students and industry professionals, can explore a set of CS programs for a variety of purposes.

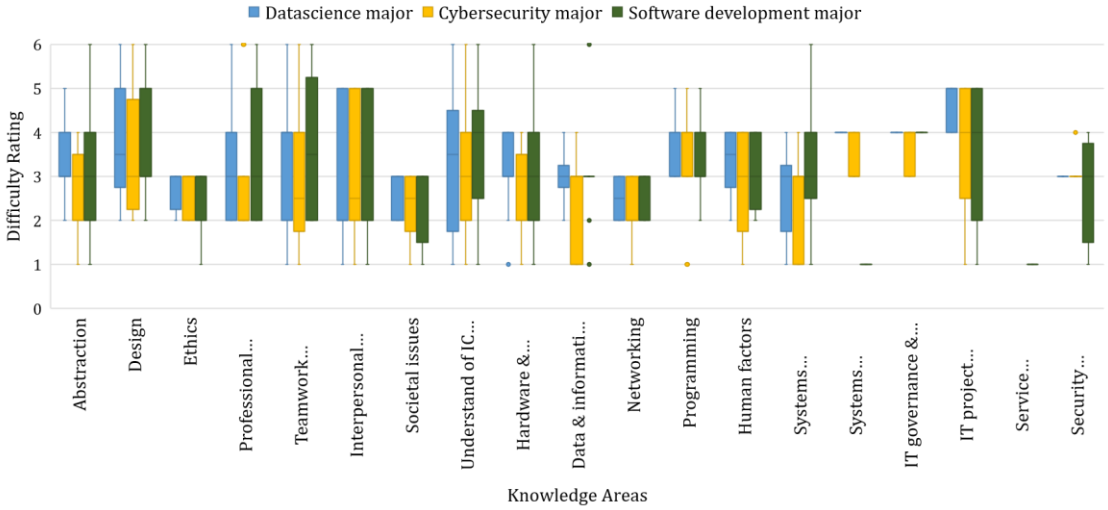

Figure 3. Comparison of majors.

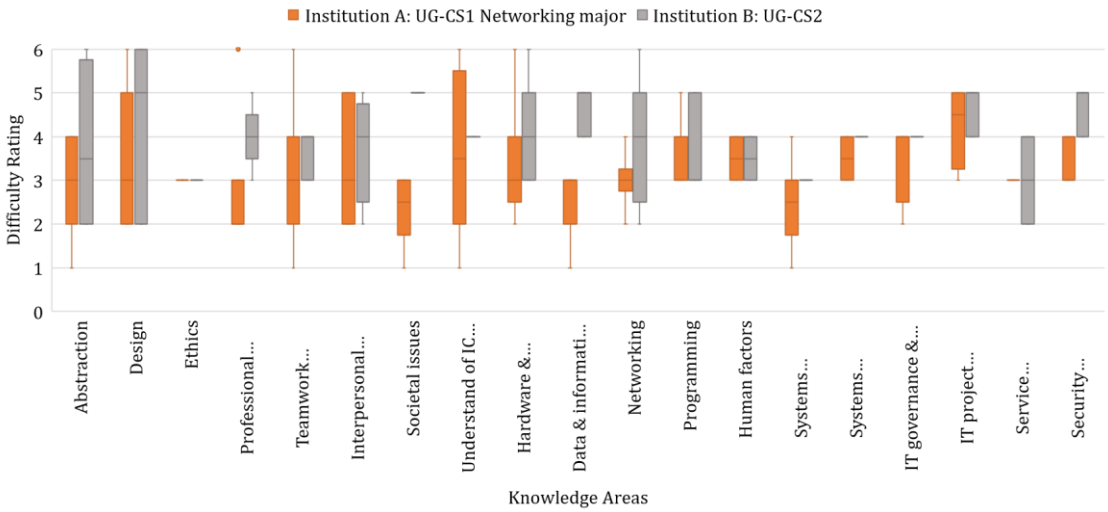

Figure 4. Comparison of networking courses across institutions A and B.

5. Conclusion and Future Work

We proposed a unified approach for analysing CS courses based on a BOK and a learning taxonomy. The case study based on the Australian context demonstrated the adaptability of the proposed approach for any specific context. Thus, ACS CBOK and Bloom's taxonomy was used as the BOK and the learning taxonomy of the unit representation, respectively. The use of ACS CBOK and Bloom's taxonomy in the ACS accreditation process facilitates the immediate adoption of the proposed approach. The case study demonstrated the use of the proposed approach for unit and course analysis. The proposed unified unit representation has multifaceted benefits. For example, education providers can analyse the courses objectively, which will help them improve/redesign existing courses; professional bodies such as ACS can compare different programs and assess their strengths and weaknesses. Furthermore, a comparison of courses across different universities and courses with different majors

can provide multiple perspectives that can be beneficial to educators, students, and industry professionals. In addition, unified unit representation enables effective unit comparison, which overcomes the limitations inherent to textual description comparison.

Though we present a preliminary study based on our unified approach, it opens several avenues for future works. In the future, we intend to extend this approach to assess student performance in each KA. Students may be represented in a unified manner based on all the units completed under a course. This representation may indicate the variation of KA coverage among students due to the choice of elective units. Also, we plan to analyse a larger number of courses to get a broader perspective on CS education in Australia.

References

Australian Computer Society (ACS). (2015). *ACS Core Cody of Knowledge for ICT Professionals (CBOK)*.

Australian Computer Society (ACS). (2020). *ACS Accreditation Manual*.

Australian Computer Society (ACS) & Deloitte Access Economics. (2019). *ACS Australia's Digital Pulse 2019*. https://www.acs.org.au/insightsandpublications/reports-publications/digital-pulse-2019.html

Anderson, L. W., Bloom, B. S., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., . . . Wittrock, M. C. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*: Longman.

Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*: Academic Press.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals* (Vol. 1): D. McKay.

Bourque, P. & Fairley, R. E. (Eds). (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*: IEEE Computer Society.

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., . . . Thompson, E. (2007). Developing a computer science-specific learning taxonomy. *SIGCSE Bull., 39*(4), 152–170.

Grayston, R. (2019). *2019 Accreditation Process Reform*. Australian Computer Society.

Hoffmann, M. H. (2008). *Using Bloom's Taxonomy of learning to make engineering courses comparable.* Paper presented at the 2008 19th EAEEIE Annual Conference.

Johnson, C. G., & Fuller, U. (2006). *Is Bloom's taxonomy appropriate for computer science?* Paper presented at the Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, Uppsala, Sweden.

Masapanta-Carrión, S., & Velázquez-Iturbide, J. Á. (2018). *A Systematic Review of the Use of Bloom's Taxonomy in Computer Science Education*. Paper presented at the Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, Maryland, USA.

Méndez, G., Ochoa, X., & Chiluiza, K. (2014). *Techniques for data-driven curriculum analysis*. Paper presented at the Proceedings of the Fourth International Conference on Learning Analytics And Knowledge, Indianapolis, Indiana, USA.

Oliver, D., Dobele, T., Greber, M., & Roberts, T. (2004). *This course has a Bloom Rating of 3.9*. Paper presented at the Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30, Dunedin, New Zealand.

Pedroni, M., Oriol, M., & Meyer, B. (2007). *A framework for describing and comparing courses and curricula*. Paper presented at the Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, Dundee, Scotland.

Richard, G., Judy, K., Raymond, L., Simon, & Sabina, K. (2013). Mastering cognitive development theory in computer science education. *Computer Science Education, 23*(1), 24-57.

Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthorne, E., . . . Roach, S. (2013). Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. *Association for Computing Machinery (ACM)-IEEE Computer Society*.

Sekiya, T., Matsuda, Y., & Yamaguchi, K. (2015). *Curriculum analysis of CS departments based on CS2013 by simplified, supervised LDA.* Paper presented at the Proceedings of the Fifth International Conference on Learning Analytics And Knowledge.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). *A taxonomic study of novice programming summative assessment.* Paper presented at the Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95.

Starr, C. W., Manaris, B., & Stalvey, R. H. (2008). Bloom's taxonomy revisited: specifying assessable learning objectives in computer science. *ACM SIGCSE Bulletin, 40*(1), 261-265.