Generic Assessment Rubrics for Computer Programming Courses

Aida MUSTAPHA, Noor Azah SAMSUDIN, Nureize ARBAIY, Rozlini MOHAMED, Isredza Rahmi HAMID
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia
Parit Raja, 86400 Batu Pahat, Johor, Malaysia
{aidam, azah, nureize, rozlini, rahmi}@uthm.edu.my

ABSTRACT
In programming, one problem can usually be solved using different logics and constructs but still producing the same output. Sometimes students get marked down inappropriately if their solutions do not follow the answer scheme. In addition, lab exercises and programming assignments are not necessary graded by the instructors but most of the time by the teaching assistants or lab demonstrators. This results in grading inconsistencies in terms of the marks awarded when the same solution is being graded by different person. To address this issue, a set of assessment rubric is necessary in order to provide flexibility for critical and creative solutions among students as well as to improve grading consistencies among instructors and teaching assistants or demonstrators. This paper reports the development of assessment rubric for each domain in computer programming courses; cognitive, psychomotor, and affective. The rubrics were then implemented for one academic semester consisting of 14 weeks. An interrater reliability analysis based on Kappa statistic was performed to determine the consistency in using the rubrics among instructors The weighted kappa is 0.810, therefore, the strength of agreement or the reliability of the rubric can be considered to be 'very good'. This indicates that the scoring categories in the rubrics are well-defined and the differences between the score categories are clear.
Keywords: Scoring, assessment rubric, computer programming, cognitive, psychomotor, affective, Kappa statistics.

INTRODUCTION
Grading programming assignments and projects are similar to grading traditional assignments such as written essays. The primary distinctions between them are the unique keywords or constructs across different programming languages and the diverse possible solutions associated with a particular problem solving techniques. Traditional assessment for computer programming assignments and projects usually depends on an answer scheme that includes the source code as a model answer with marks allocated to specific lines of code. This model answer is then used by the instructors to allocate marks to the students' programs based on the provided source code in the answer scheme.

The problem with the traditional schema-based approach of awarding marks according to a "point-per-correct-statement" is that students are being graded based similarity of their solution to the answer scheme. This leads to little or no consideration given to creativity and originality in the student solutions. In programming, the same problem can usually be solved using different constructs but still producing the same output. Students often get marked down inappropriately if their solution is not exactly the same as the instructor's solution or alternatively marked up if their solution is similar to the provided solution. In addition, lab exercises and programming assignments are not necessary being graded by the instructors but most of the time by the teaching assistants or lab demonstrators. This results in grading inconsistencies in terms of the marks awarded when the same solution is being graded by different person. Instructors, for example, may emphasize on the design of the solutions. Demonstrators, on the other hand, may emphasize on the programming syntax.

To address this issue, a set of assessment rubric is necessary in order to provide flexibility for critical and creative solutions among students as well as to improve grading consistencies among instructors and teaching assistants or demonstrators. The literature has revealed that strategies used to grade programming assessments has evolved from grading students based on an answer scheme where marks are allocated to individual programming statements to a more holistic and inclusive methodology using rubrics. A rubric is a set of ordered categories to which a given piece of work can be compared. Scoring rubrics specify the qualities or processes that must be exhibited in order to assign a particular evaluative rating for a performance (McDaniel, 1993). As a grading tool, rubrics have successfully enable the instructors to assess the student's understanding and creativity to produce a solution in programming courses (Becker, 2003; Ahoniemi and Karavirta, 2009; Payne et al., 2012) as well as evaluating research skills in strategic management (Whitesell and Helms, 2013), ethical behavior (Carlin et al., 2011), critical thinking in engineering (Ralston and Bays, 2010; Loon and Lao, 2014), and reflective writing in medicine (Wald et al., 2012).

This study hypothesizes that rubrics provide the necessary structure and guidance that enable instructors to award marks as a whole for students' ability in problem solving, creativity, and aesthetics of any graphical user interface as well as the use of good programming practice and standards. The central focus of this research will be on creating a set of rubrics as a benchmark to measure student learning outcomes in introductory computer programming courses offered by the Faculty of Computer Science and Information Technology (FCSIT) at Universiti Tun Hussein Onn Malaysia (UTHM). At present, UTHM has to cope with very large first year classes with average of 70 students per section with multiple sections to cater four specializations of undergraduate Computer Science programs: Software Engineering, Information Security, Web Technology, and Multimedia Computing. This necessitates for more than one instructor and teaching assistants for lab sessions in each program. Due to the high number of student enrollment and diverse background of the instructors or demonstrators, grading lab assignments and group projects is particularly a challenge especially in ensuring fair delivery to all students.

The main goal for this study is to promote critical and creative thinking skills and to improve grading consistencies in programming subjects by introducing a generalized programming rubric to be used across all programming languages such as C, C++, and Java. The outcome of this research is able to increase the effectiveness in teaching and learning activities in terms of consistent assessment of the course learning outcomes. The rubric developed in this study is presented in the section following the related works. Next, the research methodology is detailed out to explain the validation process of the developed rubrics followed by the findings. Finally, the paper is concluded with some indication for future research.

RELATED WORK

The Outcome-based Education (OBE) system emphasizes the importance of a curriculum content to be driven by learning outcomes (Spady, 1994). In OBE, the learning outcomes are expressed as statements of knowledge and skills individual students should possess at the end of the course they enrolled. An OBE system offers a comprehensive approach to organize and operates an education system that is focused on successful demonstration of learning sought from students at the end of the learning cycle (Murphy and Duncan, 2007).

The OBE system has been introduced to the Faculty of Computer Science and Information Technology (FCSIT) at Universiti Tun Hussein Onn Malaysia (UTHM) since 2004. The learning outcomes of a program are set by various level of academic management team at FCSIT. There are three primary components of the OBE system; Program Educational Outcome (PEO), Program Learning Outcome (PLO), and Course Learning Outcome (CLO). The PEO expresses statements of long term objectives that describe what a Computer Science should be able to demonstrate as a result of attending its program. Clearly, the achievement of the PEO at faculty level is geared to the achievement of the vison and mission of UTHM. Table 1 shows the PEO for one of the Computer Science undergraduate program offered at FCSIT, which is the Bachelor of Computer Science (Software Engineering).

Table 1: Program Educational Outcome (PEO).

PEO 1	Apply basic knowledge, principles and skills in the field of Computer Science to meet the job specification. (Knowledge / Practical Skills)
PEO 2	Implement the responsibility for solving problems analytically, critically, effective, innovative and market-oriented. (Critical Thinking and Problem Solving / Life-long Learning and Information Management / Enterpreneurship Skills)
PEO 3	Acts effectively as an individual or in a group to convey information within the organization and community. (Team Working Skills / Communication Skills)
PEO 4	Practicing good values and ethics in a professional manner in the community and able to act as a leader. (Profesional, Social, Ethics, and Humanity / Leadership Skills)

The PEO statements are further refined to establish PLO. The PLOs highlight individual student's abilities that reflect their learning experiences at FCSIT. In addition, the management team of FCSIT is also required to consider the general learning objectives set by the Malaysian Qualifications Agency (MQA, 2008) and the Ministry of Higher Education (MOHE) in expressing the PLO. As a result, the PLO are expressed to satisfy components of MQA standards which include knowledge, practical skills, communication, critical thinking and problem solving, teamwork, life-long learning and information management, entrepreneurship, moral, professional and ethics and finally leadership. Students of the undergraduate programs at FCSIT are expected to

acquire the PLO upon completion of their studies. The implementation of the PLO is he PLO is then distributed across individual courses in the undergraduate programs. Table 2 shows the PLO for Computer Science programs at FCSIT.

Table 2: Program Learning Outcome (PLO).

PLO 1	Applying knowledge and understanding of essential facts, concepts, principles and theories in the field of Computer Science Software Engineering. (Knowledge – K)
PLO 2	Implementing Software Engineering knowledge in analyzing, modeling, designing, developing and evaluating effective computing solutions. (Practical Skill – PS)
PLO 3	Communicate in spoken and written form in order to convey information, problems and solutions to the problems effectively. (Communication – CS)
PLO 4	Analyze the appropriate techniques in the field of Software Engineering to solve problems using analytical skills and critical thinking. (Critical Thinking, Problem Solving – CTPS)
PLO 5	Demonstrate teamwork skills, interpersonal and social effectively and confidently. (Team Work – TS)
PLO 6	Using the skills and principles of lifelong learning in academic and career development. (Life Learning and Information Management – LL)
PLO 7	Fostering entrepreneurship in career development. (Enterpreneurship – ES)
PLO 8	Adopt values, attitudes and responsibilities in a professional manner from ths aspects of sosial, ethics and humanity. (Moral, Professional and Ethics – EM)
PLO 9	Effectively carry out the responsibilities of leadership. (Leadership – LS)

The PLOs serve as the basis of determining the course learning outcomes (CLO) for every course offered. Each set of programming CLO in the course syllabus is mapped to the PLO of FCSIT. The mapping is known as CLO-PLO matrix. The CLO shall be constructed in such a way to accommodate the PLO. The establishment of the CLO in programming courses applies principles of Bloom's Taxonomy which covers three learning domains outlined by MQA standard: cognitive, affective, and psychomotor (Bloom et al., 1994). Table 3 presents the complete set of levels in each domain.

Table 3: Levels in cognitive, psychomotor, and affective domain based on Bloom's taxonomy.

Level	Cognitive Domain
C1	Knowledge (KN)
C2	Comprehension (CO)
C3	Application (AP)
C4	Analysis (AN)
C5	Synthesis (SY)
C6	Evaluation (EV)

Level	Psychomotor Domain
P1	Perception
P2	Set
P3	Guided response
P4	Mechanism
P5	Complex overt response
P6	Adaptation
P7	Origination

Level	Affective Domain
A1	Receiving phenomena
A2	Responding to phenomena
A3	Valuing
A4	Organizing values
A5	Internalizing values

Eventually, to measure the achievement of cognitive, psychomotor, and affective domain in each CLO, a student is evaluated using one to five assessment tools: quiz, test, laboratory assignments, project, and final exam. Each of the assessment tool is assigned to ensure positive achievement for the courses. Indeed, such information has implication on the achievement of CLO and PLO that are usually evaluated at the end of the learning process. Table 4 shows a sample of specification table to evaluate the cognitive domain in an object-oriented programming course. The specification table is designed to plan the distribution of marks based on taxonomy level mapping. Such constructive mapping is valuable to evaluate how the CLO and PLO are evaluated and related and finally implies the PEO.

Table 4: A specification table for an object-oriented programming course.

Question No.	Course Content/ Topic	Marks Distribution based on Bloom's Taxonomy						Subtotal
		KN	CO	AP	AN	SY	EV	
		Level 1		Level 2		Level 3		
Q1 (a)	Chapter 2: Primitive data types	3						24
Q1 (b)	Chapter 3: Fundamental of OO	6						
Q1 (c)	Chapter 3: Fundamental of OO	6						
Q1 (d)	Chapter 4: Object and classes					9		
Q2 (a)	Chapter 3: Fundamental of OO				12			27
Q2 b)	Chapter 3: Fundamental of OO				15			
Q3 (a)	Chapter 5: Inheritance and polymorphism		5					25
Q3 (b)	Chapter 5: Inheritance and polymorphism			20				
Q4 (a)	Chapter 4: Object and classes				5			24
Q4 (b)	Chapter 4: Object and classes					10		
Q4 (c)	Chapter 4: Object and classes					9		
Subtotal based on taxonomy (Marks)		15	5	20	32	28	0	100
Subtotal for each level (Marks)		20		52		28		40%
Cognitive level (%)		20%		52%		28%		100%
Distribution of cognitive level (%)		5%		35%		60%		100%

At FCSIT, the specification table is used to assess only the cognitive domain via quizzes, tests, and final exams. The assessment method is still using the answer scheme. However, assessments for lab assignments and projects are not necessary being graded by the instructors but most of the time by the teaching assistants or lab demonstrators. This calls for the need of a generalized rubric to cover all continuous learning assessments other than tests and final exams.

RESEARCH METHODOLOGY

A rubric is a set of categories developed based on a specific set of performance criteria. As an assessment tool, a rubric should cover all learning domains offered in computer programming courses. The purpose of such classification is to categorize different objectives that educators set for the students because educators have to focus on all three domains to create a more holistic form of delivery. In order to develop the rubric, the first step is to identify the learning outcomes at the program level followed by the course level before the types of assessments could be determined. The rubric can then be developed for a specific type of assessment such as lab assignments or group projects. In this study, the rubric development and validation process are founded on the principle of continuous feedback and improvement involving the following steps:

Step 1: Identify Program Learning Outcomes (PLO) and Course Learning Outcomes (CLO)

From the curricula, all programming courses are selected involving different languages (i.e. C, C++, Java). The PLOs and CLOs for each course were tabulated and compared. At FCSIT, UTHM, each course has three CLOs in average. Next, the assessment types were determined across all the courses and the percentage of each assessment type according to the PLO and CLO were distributed. Again, the types of assessment include tests, assignment, practical/lab, group project and final examination. Table 5 shows the mapping of PLOs and CLOs across all programming courses. The types of assessments are also indicated for each learning objective.

From the list of assessment methods provided in the table, quiz, test, and final examinations in CLO1 are graded based on traditional schema-based approach because the tools are only assessing the cognitive learning domain in computer programming. Lab assignments (CLO2) and projects (CLO2, CLO3), however, are designed to assess all three learning domains; cognitive, psychomotor, and affective. Because each CLO assess only one learning domain, the rubrics developed will be categorized according to the CLO. For each CLO, the level of domain for cognitive, psychomotor, affective are also assigned.

Table 5: Mapping of course learning outcomes to program learning outcomes across all programming courses.

		Program Learning Outcome (PLO)									
		Knowledge	Knowledge & Practical	Communication Skills	Critical Thinking & Problem Solving	Team Working Skills	Life-long Learning	Entrepreneurship Skills	Professionalism, Social, Ethics and	Leadership Skills	
Course Learning Outcomes (CLO)		PLO 1	PLO 2	PLO 3	PLO 4	PLO 5	PLO 6	PLO 7	PLO 8	PLO 9	Assessment
CLO 1	Design problem solving process based on object oriented concept.				C5						Quiz, Test, Lab, Project, Final Examination
CLO 2	Construct an object oriented computer application using Java programming language.		P4								Lab, Project
CLO 3	Demonstrate the implementation of object oriented concept using any high level programming language.						A3				Project Presentation

Step 2: Formulate the rubric

In formulating the rubric, one or more dimensions that serve as the basis for judging the student work were determined. Each CLO was broken into one or more objectively measurable performance criteria along with its sub-criteria. The basic dimension in the rubric is the assessment type, whether delivered by the students in the form of written reports or via presentation. Next, for each dimension, a scale of values from 1 to 5 on which to rate each dimension is assigned; 1 is being very poor, 2 is poor, 3 is fair, 4 is good, and 5 is excellent. Finally, within each scale, the standards of excellence for specified performance levels accompanied were provided. Table 6 to Table 8 show the rubric for CLO1 (cognitive), CLO2 (psychomotor), and CLO3 (affective), respectively.

Table 6: Rubric for CLO1. Design problem solving process using algorithm/object-oriented concepts (Cognitive – C5, PLO4 – CTPS).

Assessment	Criteria	Sub-criteria	Level	1	2	3	4	5
Report	Ability to analyze problem and identify requirements	Identify correct input/ output	C2	Unable to identify any input and output	Able to identify only one input or output	Able to identify correctly some input and output	Able to identify correctly all input and output	Able to identify correctly all input and output and provide alternative
	Ability to	Construct	C3	Unable	Able to	Able to	Able to	Able to

	demonstrate design solution	correct flowchart or pseudocode		to construct	construct but mistake on symbol	construct correctly	construct correctly and use proper elements	construct correctly, use proper elements and documenta-tion

Table 7: Rubric for CLO2. Construct a computer application/object oriented computer application using object:-oriented concepts (Psychomotor – P4, PLO2 – Practical Skill)

Assessment	Criteria	Sub-criteria	Level	1	2	3	4	5
Report	Ability to apply required data type or data structure	Appropriate choice of variable names or data structure (i.e. array/ linked list)	P3	Unable to identify required data type or data structure	Able to identify required data type or data structure but does apply correctly	Able to apply required data type or data structure but does not produce correct results	Able to apply required data type or data structure and produce partially correct results	Able to apply required data type or data structure and produce correct results
	Ability to apply required control structure	Correct choice of sequential, selection or repetition control structure	P4	Unable to identify required control structure	Able to identify required control but does apply correctly	Able to apply required control structure but does not produce correct results	Able to apply required control structure and produce partially correct results	Able to apply required control structure and produce correct results
	Ability to run/debug	Free from syntax, logic, and runtime errors	P3	Unable to run program	Able to run program but have logic error	Able to run program correctly without any logic error	Able to run program correctly without any logic error and display inappropriate output	Able to run program correctly without any logic error and display appropriate output
	Ability to perform input validation	Validate input for errors and out-of-range data	P3	The program produces incorrect results	The program produces correct results but does not display correctly Does not check for errors and out-of- range data	The program produces correct results but does not display correctly. Does little check for errors and out-of- range data	The program works and meets all specifica-tions. Does some checking for errors and out-of- range data	The program works and meets all specifica-tions. Does exceptional checking for errors and out-of- range data
Presentatio	Ability to	Comment /	P1	No	Docume	Docume	Document	Document

n	produce readable program	Description		docume ntation	ntation is simple comment in code	ntation is simple comment s embedde d in code with header separatin g the codes	ation is simple comments and header that useful in understan ding the code	ation is well-written and clearly explains what the code is accomplis hing
		Indentation / Naming Convention	P2	Unable to organiz e the code	The code is poorly organize d and very difficult to read	The code is readable only by a person who already knows its purpose	The code is fairly easy to read	The code is extremely well organized and easy to follow

Table 8: Rubric for CLO3. Demonstrate the implementation of problem solving process/object-oriented concepts using high-level programming language (Affective – A3, PLO6 – Lifelong Learning)

Assessment	Criteria	Sub-criteria	Leve l	1	2	3	4	5
Presenta-tion	Ability to demonstrat e program in group	Demonstrat e understand-ing on program design	A3	Unable to explain program design	Able to explain a little program design	Able to explain some program design	Able to explain entire program design correctly as it is	Able to explain program design correctly and provide alternativ e solutions
		Organizatio n of group presentatio n	A4	Materials are not organize d with missing infor-mation	Materials are partially organize d with missing infor-mation	Material s are partially organize d with required infor-mation	Materials are highly organized with required infor-mation	Materials are highly organize d with additiona l infor-mation
		Cooperatio n from all members	A2	Unable to cooper-ate in a group	Forced coopera-tion through interven-tion	Demon-strate coopera-tion after interven-tion	Demon-strate coopera-tion through personal dominanc e	Demon-strate coopera-tion through group hierarchy

The rubrics have been developed as a 2D grid in Microsoft Excel sheet, where each row describes one evaluation criteria and the columns indicate the level of achievement. Since the rubric is already in an Excel form, the instructors simply fill in the student performance according to the desired column and the form will add up the corresponding values to produce a final score.

Step 3: Test the reliability of the rubric

Reliability refers to the consistency of assessment scores. On a reliable test, a student would expect to attain the same score regardless of when the student completed the assessment, when the assessment was scored, and who

scored the assessment. In order to measure the reliability of the rubrics, the rater reliability in the form of reliability coefficient is measured. Raters reliability refers to the consistency of scores that are assigned by two independent raters (inter-rater reliability) and that are assigned by the same rater at different points in time (intra-rater reliability) (Moskal and Leydens, 2000). According to Jonsson and Svingby (2007), the consensus agreement among raters depends on the number of levels in the rubric, whereby fewer levels lead to higher chance of agreement.

This study adopted the measurement of inter-rater reliability based on Kappa statistics (Cohen, 1960). In Cohen's kappa, values between 0.4 and 0.75 represent fair agreement beyond chance. Values ≤ 0 as indicating no agreement and 0.01–0.20 as none to slight, 0.21–0.40 as fair, 0.41– 0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1.00 as almost perfect agreement (McHugh, 2012).

EVALUATIONS

The rubrics developed in this study was implemented in three programming courses are offered during the First Semester of 2015/2016. The courses were Computer Programming (BIT10303) using C programming language, Object-Oriented Programming (BIT20603) using C++ programming language, and Java Programming (BIT33803). The rubrics were consistently used for grading lab assignments and group projects throughout the 14-week period of the semester. All the assignments and projects were graded independently by two random instructor or lab demonstrator using the same rubric. Table 9 shows the total number of students works/artifacts being compiled and graded based on the rubrics.

Table 9: Summary of total written artifacts graded using the rubrics. The artifacts for lab assignments and groups projects are in the form of source codes.

Course	No. of Students (a)	No. of Instructors/ Demonstrators (b)	No. of Lab (c)	No. of Assignments (d)	No. of Projects (e)	Total Artifacts (a * (c + d + e))
BIT10303	60 (S1) + 37 (S2) = 97	2	9	1	1	1,067
BIT20603	73 (S1) + 37 (S2) = 110	2	7	1	1	990
BIT33803	76 (S1) = 76	1	5	0	1	456
					Total	2,513

*S*i* indicate section number.

Based on Table 9, all sets of scores (i.e. four sets for BIT10303, two sets each for BIT20603 and BIT33803) are then statistically analyzed for inter-rater reliability using the Cohen's Kappa (Cohen, 1960). According to this metric, a Kappa of 1 indicates a perfect agreement, whereas a kappa of 0 indicates agreement equivalent to chance. The analysis was performed using the program Statistical Package for the Social Sciences (SPSS), version 20.0. Note that the instructors or demonstrators are referred as raters in calculating the kappa values. Two raters were randomly picked to evaluate the each artifact. Table 10 presents the results for both raters on every artifact.

Table 10: Assessment results for 2,513 artifacts by two independent raters.

	Rater #2					
Rater #1	1 (very poor)	2 (poor)	3 (fair)	4 (good)	5 (excellent)	Total
1 (very poor)	364	207	0	0	0	571
2 (poor)	161	349	55	1	0	566
3 (fair)	0	6	295	108	2	411
4 (good)	0	1	18	312	109	440
5 (excellent)	0	0	3	107	415	525
	525	563	371	528	526	2,513

Based on Table 10, the total number of observed agreements is 735, which constitutes 69.04% of the observations. The number of agreements expected by chance is 509.1, which is 20.26% of the observations. The kappa value is 0.612 with 95% confidence interval from 0.589 to 0.634. Based on the kappa value, the reliability of the rubrics is considered to be 'good' based on the strength of agreement between the two raters.

However, this calculation only considered exact matches between the two raters. Since the scale of dimensions

(very poor, poor, fair, good, excellent) are ordered, close matches were also being considered. This means if the first rater assessed an artifact as fair and the other as good, this is closer than if the rater assessed the artifact as poor and the other excellent. The calculation of weighted kappa assumes the categories are ordered and accounts for how far apart the two raters are. The weighted kappa is 0.810, therefore, using this approach the strength of agreement or the reliability of the rubric can be considered to be 'very good'. This indicates that the scoring categories in the rubrics are well-defined and the differences between the score categories are clear.

CONCLUSIONS

A generic programming rubric is proposed to be used across all programming courses offered by FCSIT at UTHM involving a variety of high-level programming languages such as C, C++, and Java. The rubrics are shared with the students every time a lab exercise or assignment is assigned to help them better understand the balance of the different activities in their final grade. From the rubrics, students are able to estimate the amount of effort that are required to achieve the perfect score. In this way, students are also playing active role of becoming independent in determining their own learning objectives. In the future, the rubrics will be used in establishing benchmarks for the programming courses and analyzing student performance to improve the learning and learning process including making adjustments to the curriculum.

ACKNOWLEDGEMENT

This project is sponsored by the Contract Research Grant from the Centre for Academic Development and Training (CAD) at Universiti Tun Hussein Onn Malaysia (UTHM).

REFERENCES

Ahoniemi, T. & Karavirta, V. (2009) Analyzing the use of a rubric-based grading tool. *14th Annual ACM SIGCSE Conference on Innovation and Technology in CS Education* (pp.333-337).

Becker, B. (2003). Grading programming assignments using rubrics. *8th Annual Conference on Innovation and Technology in Computer Science Education* (pp.253-253). ACM, New York, NY, USA.

Bloom, B. S., Anderson, L., & Sosniak, L. (1994). "*Bloom's taxonomy*: *A forty-year retrospective*". Assessing Scholarly. Chicago: NSSE. Ball, CE (2012).

Carlin, N., Rozmus, C., Spike, J., Willcockson, I., Seifert, W., Chappell, C., Hsieh, P.-H., Cole, T., Flaitz, C., Engebretson, J., Lunstroth, R., Amos, C., & Boutwell, B. (2011). The health professional ethics rubric: Practical assessment in ethics education for health professional schools. *Journal of Academic Ethics*, vol. 9, no. 4 (pp.277-290).

Cohen, J. (1960). A coefficient for agreement for nominal scales. *Education and Psychological Measurement*, vol. 20 (pp.37-46).

Herman, J.L. (1992). *A practical guide to alternative assessment*. Association for Supervision and Curriculum Development, 1250 N. Pitt Street, Alexandria, VA 22314.

Jonsson, A. & Svingby, G. (2007) The use of scoring rubrics: Reliability, validity and educational consequences. *Educational Research Review*, vol. 2 (2007) (pp.130-144).

Loon, J.E.V. & Lai, H.L. (2014). Information literacy skills as a critical thinking framework in the undergraduate engineering curriculum. *2014 ASEE North Central Section Conference.*

McDaniel, E. (1993). *Understanding educational measurement*. Dubuque, IA: William C. Brown.

McHugh, M.L. (2012) Interrater reliability: the kappa statistic. *Biochem Med*, vol. 22, no. 3 (pp.276–282).

Moskal, B.M. & Leydens, J.A. (2000). Scoring rubric development: validity and reliability. Practical Assessment, *Research & Evaluation*, 7(10).

Malaysian Qualification Agency (MQA). (2008). Code of Practice for Programme Accreditation.

Murphy, J.J. & Duncan, B.L. (2007). Brief intervention for school problems (2nd ed.): Outcome-informed strategies. New York: Guilford Press.

Payne, N., Kolb, D., & Kotze. G. (2012). "Scheming" to optimize marking in computer programming: From memos to rubrics. *ICERI 2012* (pp.869-877).

Ralston, P. & Bays, C. (2010). Refining a critical thinking rubric for engineering. *American Society for Engineering Education*.

Spady, W.G. (1994). Outcome-based education: Critical issues and answers. American Association of School Administrators, 1801 North Moore Street, Arlington, VA 22209.

Wald, H.S., Borkan, J.M., Taylor, J.S., Anthony, D., & Reis, S.P. (2012). Fostering and evaluating reflective capacity in medical education: Developing the REFLECT rubric for assessing reflective writing. *Academic Medicine*, vol. 87, no. 1 (pp.41-50).

Whitesell, M. & Helms, M.M. (2013). Assessing business students' research skills for the capstone project in the strategic management course. *Journal of Business & Finance Librarianship*, vol. 18, no. 1.