Establishing Evaluation Criteria for Assessing Novices' Ability in Applying Object-Oriented Concept Using Delphi Approach

Norazlina Khamis

***Abstract*—There are many approaches in assessing students' ability in object-oriented programming, but little research has been discussed concerning the evaluation of novices' ability in applying object-oriented fundamental concepts in their source code. The main purpose of this study is to construct and validate through expert consensus, a set of evaluation criteria for assessing novices' ability through evaluation of the application of object-oriented concepts in program source code. These evaluation criteria are derived based on the common fundamental object-oriented concepts based on Malaysian object-oriented programming syllabuses. A three-round Delphi approach has been chosen in establishing the consensus and participated by a number of OOP experts. Based on the Delphi study, sixteen evaluation criteria have been established for assessing novices' ability in applying fundamental object-oriented concepts. The proposed evaluation criteria were then validated by associating them with established related object-oriented design heuristics and object-oriented design principles.**

***Index Terms*—Object-oriented programming, programming assessment, Delphi approach, object-oriented concepts.**

I. Introduction

The most influential programming paradigm today is object-oriented programming (OOP), and it has changed the art and practice of writing computer applications [1]. Recognizing the importance of equipping students with object-oriented knowledge and skills, most universities in Malaysia include OOP as their introductory course for programming [2]. Normally, students doing programming must be able to show ability in two areas [3]. First, they must be able to understand and verbalize the concepts involved in programming. Secondly, they must also be able to produce well-written, well-structured and understandable applications using the language involved. Kolling [4] believes that understanding key concepts is important in programming education. In the case of teaching and learning of OOP for example, it is important that students get a good understanding of the core concepts of object-oriented programming. Many studies point at the necessity of good understanding of central concepts within OOP. Fleury [5] found that students constructed their own understanding of concepts in their programming assignments, and those constructions are not always complete and correct as highlighted below:

"*Because students construct their own meanings during instructions, it is not surprising that students possess only partial conceptions even when provided with complete and accurate information* [5]."

In fact, number of studies regarding OOP [4], [6]-[9] reported of misconceptions related to the understanding of object-oriented concepts among students. How to assess students "ability in programming? Are students able to apply the fundamental concepts they have learned in their programs? One way to answer these questions is through assessment. Assessment is at the heart of education and educators use assessment to gauge students" academic strengths and weaknesses. According to Brown *et al*. [10], assessment is the key driver of student learning:

"*Assessment defines what students regards as important, how they spend their time and how they come to see themselves as students and then as graduates. If you want to change student learning then change the methods of assessment [10]*."

The OOP education community has also shown interest in assessment research. The goal of assessment research in most OOP education is to have valid ways of measuring novices ability in OO programming. Many assessment tools have been developed by educators and researchers (e.g., [11], [12]) to measure students ability in OO programming. These tools usually focus on assessing technical and didactic quality aspect, i.e. correctness of the output, appropriate programming process is followed or appropriate programming styles. Students ability in applying the fundamental OO concepts in source code is not addressed. Too often, educators have to develop their own assessment instrument every time they want to examine students learning in programming. This is due to unavailability of validated assessment instruments in CS disciplines specifically in OOP courses, compared to many science, technology, engineering and mathematics (STEM) disciplines [13]. It is difficult to evaluate student learning accurately or students conceptions on fundamental concepts in programming without a valid and reliable assessment instrument. Thus the goal of this research is to propose a set of evaluation criteria which will be incorporated into a new assessment instrument to support current assessment approach in OOP. The proposed assessment approach will be able to evaluate students understanding of fundamental object-oriented programming concepts at that particular time based on the source code

Manuscript received October 9, 2014; revised February 4, 2015.

Norazlina Khamis is with the Faculty of Computing and Informatics, Universiti Malaysia Sabah Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia (e-mail: azlinakhamis@gmail.com).

produced and are independent of any programming languages which currently not addressed in current assessment approaches.

One case for establishing the evaluation criteria is as follows. During assessment, educators seem to neglect to consider whether students have really applied object-oriented concepts in the appropriate way in source code. Apparently, the criteria they usually base on are like: does it compile, does it provide correct output, and does it contain sufficient useful comments? The source code might be working and produces correct output. However, it does not follow that object-oriented concepts have been correctly and appropriately applied. Another case for establishing the criteria is that different educators use different criteria during marking, thus leading to inconsistencies in assessment.

II. Methods

In the following subsection, we will discuss the methods used in establishing evaluation criteria. In order to establish evaluation criteria, the researcher utilized both qualitative and quantitative research methods with educators to construct and refine the evaluation criteria. The first step in this phase is to identify a set of fundamental object-oriented concepts that are covered by most OOP courses in Malaysian public universities. Next, the identified concepts will serve as a basis in establishing the evaluation criteria to assess novices ability in applying those concepts as well as contents validation through a Delphi study. The resulting evaluation criteria are further validated with object-oriented design heuristics and principles. The final output from this phase will be a set of validated evaluation criteria which will be used in designing the assessment guidelines. The forthcoming subsections describe in detail the process of establishing the evaluation criteria.

A. Identification of Commonly Accepted Fundamental Object-Oriented Concepts

Generally, there are many concepts when we talk about object-orientation paradigm. A literature search for definitions of object-oriented concepts will produce different results. Thus, it is not surprising that there is some confusion regarding definitions of object-oriented concepts and terms. For educators, one of the difficulties faced is to be able to prioritize them according to their importance. Four sources are used as a basis to identify the fundamental concepts:

1) Identification via frequency of occurrence of object-oriented terms
2) Identification of Object-Oriented Concepts via Analysis of Course Syllabuses.
3) Identification of Object-Oriented Concepts from ACM CC 2008.
4) Identification of Object-Oriented Concepts via Object-Oriented Misconceptions.

Based on the analysis and comparison between Amstrong's [14] object-oriented taxonomy and the various sources described above, eight frequently occurring object-oriented concepts are selected as the fundamental object-oriented concepts relevant to this research. Those concepts are **object**, **class**, **abstraction**, **polymorphism**, **encapsulation**, **inheritance**, **message passing** and **method**. Table I shows the commonly accepted concepts according to various sources used.

TABLE I: Commonly Accepted Concepts According to Various Sources Used

	Identification approaches			
	Frequency of Occurrences	Malaysian OOP Syllabus	ACM CC 2001	OO Misconceptions
Object	√	√	√	√
Class	√	√	√	√
Abstraction	√	√	√	√
Polymorphism	√	√	√	√
Encapsulation	√	√	√	√
Inheritance	√	√	√	√
Message passing	√	√	-	√
Method	√	√	√	√

Based on this concepts, we establish evaluation criteria for assessing novices' ability in applying these fundamental concepts in their source code.

B. Establishing Validated Evaluation Criteria to Assess Novices' Ability

This section describes the approach to establishing the evaluation criteria to assess the novices‟ ability based on the source code that they produced. It involves the incorporation of a Delphi study in order to elicit knowledge and experiences of experts. The Delphi study is a process used to survey and collect the opinions of experts on a particular subject. The main reason for integrating Delphi is that besides eliciting knowledge from experts, it also assesses content validity at the same time. Content validity refers to the degree to which the content of the items reflects the content domain of interest [15]. A content validity study can provide information on the representativeness and clarity of each item and a preliminary analysis of the factorial validity. In addition, the expert panel offers concrete suggestions for improving the measure. Thus, at the end of Delphi study, the content validity is determined.

A preliminary and a three-round Delphi study have been conducted and it involved the participation of educators who are experts in the object-oriented programming domain. In theory, the Delphi process should be iterated until a consensus is achieved. It has been found in the literature [16] that three iterations are often sufficient to collect the needed information and to reach a consensus in most cases. The major objective of this activity is to establish a set of generic evaluation criteria for assessing novices‟ ability in applying fundamental object-oriented concepts in their source code based on experts‟ knowledge and experiences. Through conducting the Delphi study, feedback from the experts is acquired which includes obtaining their ideas and suggestions based on their experiences.

The Delphi study assessed two sorts of agreement. First, the extent to which each expert individually agrees with a list of object-oriented concepts through preliminary Delphi. Second, the extent to which experts collectively agree on the evaluation criteria established during the three-round Delphi study. This sort of agreement is referred to as the "consensus" that is representing the distribution of opinions of all experts. All questionnaires of the Delphi study were pre-tested for

validation among experts in the field of object-oriented programming research.

The following discussion describes the process involved in the Delphi study in greater detail.

1) Selection of experts

Research and practice in object-oriented programming vary greatly; in the topics studied, the theoretical framework used, study methods, the materials developed and so on. Thus, choosing the appropriate experts for the Delphi study is the most crucial step in the entire process. Selecting the right experts directly relates to the quality of the data generated. According to literature, the criteria that can be used to select Delphi participants are individuals who have related background and experiences on the issues, capable of providing fruitful inputs and willing to revise their initial inputs to achieve the consensus [16]. The first round of the Delphi study was mainly for generating a list of possible evaluation criteria on novices application of the fundamental object-oriented concepts. As diversity was important, the first round of experts had to be representatives from a wide variety of research and practice. A number of experts involved in teaching OOP from a diverse set of institutions in Malaysian public universities were requested to help in setting the scope for these concept inventories. Therefore, homogeneous sample of experts among educators were recruited. They were experienced in theory-based teaching, research or practice related to object-oriented programming development. The experts who participated in this investigation had to fulfill these selection criteria:

Educators who have experience in teaching object-oriented programming courses for more than five years at the undergraduate level AND have conducted extensive research in the object-oriented programming domain for more than five years AND/OR have been involved in object-oriented software development for more than five years

Eight experts were selected as participants and the identity of the experts remains anonymous amongst themselves. Next process is preparing the questionnaire.

Two important aspects of questionnaire design are the structure of the questions and the decision on the type of response format for each question. Questions can be classified into three types: closed, open-ended and contingency questions.

2) Questionnaire design

Open-ended questions have been chosen to support the Delphi study. Open-ended questions are usually not followed by any choice of answers. The respondent must answer by supplying a response and it will be recorded in full. One of the main advantages of open-ended questions is that they allow the respondent to express their ideas in their own words. They can also suggest new information when there is very little existing information available about a topic. Nevertheless, the main disadvantage of open-ended questions is that they may be difficult to answer and even more difficult to analyze as well as it is time consuming.

The questionnaire design in this research will focus primarily on two objectives:

Validate the fundamental OO concepts identified in previous phase through subject-matter-expert.

Establishing the evaluation criteria through educator "knowledge and experiences in assessing novices" ability in applying the fundamental concepts based on Malaysian OOP syllabus.

The next section describes the implementation as well as analysis of the Delphi study.

III. Results and Discussion

The following subsections describe in detail implementation and the results obtained from the preliminary study and the three rounds of the Delphi study.

The main objective for the preliminary round is to have the previously identified fundamental object-oriented concepts further validated by the experts. A consensus towards those proposed concepts is sought in this phase. Experts were asked to identify each of the object-oriented concepts according to their importance based on OOP course syllabus. Based on analysis, it is found that all experts agree with the previously identified fundamental concepts. The questionnaire for Delphi round one is ready to be distributed among experts.

A. Delphi Round One

A questionnaire consisting of open-ended questions was used in the first round. The questions are related to establishing evaluation criteria for assessing novices" ability in applying each of the fundamental concepts in their source code. Experts give their opinions based on their experiences in teaching and researching, and/or experiences in object-oriented development. Each expert is given a certain period of time to complete the questionnaire in this round. The period needs to be specified to ensure that the establishment of the evaluation criteria can be completed on time. Feedback and response in this round were collected for subsequent analysis.

Based on analysis, a total of 106 evaluation criteria were constructed by experts. A summary of the number of evaluation criteria derived from the experts' feedback in Delphi Round 1 is presented in Table II.

The evaluation criteria derived from the experts were compiled and arranged according to their respective concepts. Any clear redundancies amongst the evaluation criteria were removed by the researcher. This task was validated with an expert in object-oriented domain who is not a participant in the Delphi study but fulfill the same criteria for Delphi participants. Based on the analysis, 66 out of 106criteria suggested by the experts were retained. Another 40 criteria were discarded mostly due to the reasons of clear redundancies.

B. Delphi Round Two

For Delphi round two, a second questionnaire consisting of eight sections representing each object-oriented core concepts was designed. For each section, the remaining evaluation criteria for the corresponding concept resulting from previous round were compiled and listed to further assess the content validity. In this round, the experts need to determine the relevance of the evaluation criteria listed for every concept using a Likert-scale ranging from 1 (Strongly disagree) to 4 (Strongly agree) and provide comments if necessary. In this

section, concept class will be used as an example in discussion.

TABLE II: NUMBER OF EVALUATION CRITERIA DERIVED IN DELPHI ROUND 1

Expert id	E01	E02	E03	E04	E05	E06	E07	E08	Total
	Number of evaluation criteria produced								
Class	2	4	2	2	2	1	2	1	16
Object	2	4	2	1	1	1	2	1	14
Inheritance	2	3	2	1	1	1	3	2	15
Method	2	3	2	2	1	1	1	1	13
Message passing	2	1	2	2	1	1	2	1	12
Polymorphism	2	2	2	1	1	1	2	1	12
Encapsulation	1	2	2	1	1	1	1	1	10
Abstraction	3	1	1	1	2	3	2	1	14

TABLE III: ANALYSIS OF THE RELEVANCE OF EVALUATION CRITERIA FOR THE CONCEPT OF CLASS

	Expert Response on Likert Scale	
Evaluation criteria	Disagree	Agree
1. Able to construct a message to an object based on the signature and description of the class	-	100%
2. Identify the proper class, method, and attribute to solve a particular problem	-	100%
3. Identify the abstraction of an entity and its relevant characteristics andbehaviours in the form of a class	-	100%
4. Identify the properties of a class which should consist of the class name	87.5%	12.5%
5. Class can be seen as a "template" for an object	87.5%	12.5%
6. Modify/create template classes	87.5%	12.5%
7. Modify/create classes and incorporate them in class libraries	87.5%	12.5%
8. Use class containment, where appropriate	87.5%	12.5%
9. Identify the properties of a class which should consist of the class name	87.5%	12.5%
10. Use overloaded constructors and operators where required to recast existing code and provide additional functionality to classes	87.5%	12.5%

The questionnaire was then emailed to the experts for the second round of Delphi for their feedback on the relevance of every listed evaluation criteria. Once feedback had been received from all the expert, a summary report was produced which contains the list of evaluation criteria from experts for according to each concept.

For each of the evaluation criteria, its relevance to the context of research was analysed based on the ratings of Likert scale provided by the experts in their feedback. Likert scales fall within the ordinal level of measurement [17]-[19]. For ordinal data, one should employ the median or mode as the measure of central tendency [20]. Using mean and standard deviation are inappropriate for ordinal data whereby the numbers generally represent verbal statements. In addition, ordinal data may be described using frequency of response in each category [21]. Thus, the relevance of the evaluation criteria in assessing novices ability was determined based on the frequency of responses for each fundamental concept. As an example, Table III presents the result of analyzing the relevance of evaluation criteria for the concept of class.

The main objective of the analysis was to determine the most relevant evaluation criteria for each of the concepts. Those evaluation criteria with a score of 3 = Agree or 4 = Strongly Agree were considered to be potentially included as a measurement item.

Next, the Content Validity Ratio (CVR) equation is used to further validate the findings using the previous approach. The input from the experts is used to compute the CVR for each of the candidate evaluation criteria in a measurement instrument (CVR_i) as follows:

$$CVR_i = \frac{ne - N/2}{N/2} \quad (1)$$

where

CVR_i = CVR value for the ith measurement item.

i = number of experts indication a measurement item is "essential," and

N = total number of experts in the panel.

The CVR equation 1 can be infer that it takes on values between -1.00 and +1.00, where CVR = 0.00 means that 50% of the expert in the panel of size N believe that a measurement item is "essential". A CVR > 0.00 would, therefore indicate that more than half of the experts believe that a particular measurement items is "essential," and thereby face valid.

Based on the analysis, only three evaluation criteria were included in the third round of the Delphi study, whereas the rest were discarded. This does not mean, though, that the eliminated evaluation criteria were not important since the main objective was to come out with those criteria that were most important for assessing novices ability in applying fundamental object-oriented concepts in their source code. Thus, in the case of the evaluation criteria related to the concept of class in Table III, only three of them achieved 100% agreement among the experts and score CVR 0.00 and above. Those criteria were subsequently selected to be included in the next round. The same process was repeated with the other fundamental concepts.

The remaining evaluation criteria were combined into a single list and reviewed again for any redundancies or ambiguities. Once again, the process was assisted by the same expert who did it for the previous round. After the experts

validation, only 16 out of the 23 potential evaluation criteria remained. Another summary report was prepared consisting of the final sixteen (16) evaluation criteria covering all the identified fundamental concepts. The report was emailed to the experts for the final round of Delphi for a final review where the conclusions of the Delphi study were presented.

C. Delphi Round Three

In Delphi round three, the experts were required to review the final version of the list of evaluation criteria. Feedback for each criterion, including any ambiguity raised, shall be specified. This was to ensure that the final result from this round would produce a complete list of evaluation criteria that all experts unanimously agree on were the most important in assessing novice ability in applying fundamental object-oriented concepts in their source code.

Further analysis and review on the feedback revealed that seven out of eight experts agreed on the final list of evaluation criteria. Expert 05 had issues on two of the evaluation criteria which needed to be resolved to reach consensus among all the experts. A face-to-face discussion between the researcher and Expert 05 was held to get clarification regarding those issues so that a full consensus could be achieved upon resolution. The expert agreed at last with the list at the end of the discussion and with that, the Delphi study was finally completed.

D. Final Evaluation Criteria

Table IV presents the final list of evaluation criteria for assessing students ability in applying fundamental object-oriented concepts in source code based on findings from the Delphi study. In this table, each criterion listed is associated with the concepts it is related to. The criteria listed are not arranged in any order of importance and the content validity has been assessed. The panel of experts considered these to be valid with respect to providing an adequate reflection of the novices ability, and relevant with respect to the object-oriented concepts being assessed.

E. Validation of Evaluation Criteria

To strengthen the findings gained from experts in the three-round Delphi study, the evaluation criteria is further validated by mapping them with established related object-oriented design heuristics and object-oriented design principles. Design heuristics refer to experience-based techniques using "rules of thumb," an educated guess, an intuitive judgement or common sense. There are more than sixty established design heuristics, which are language independent and allow one to rate the integrity of an object-oriented design. Heuristics are not written as hard and fast rules; they are meant to serve as warning mechanisms which allow the flexibility of ignoring them if necessary. Design principles refer to the suite of eleven principles, conceived by people such as Robert Cecil Martin [22], Bertrand Meyer [23] and Barbara Liskov [24]. The validation results are presented in Table V.

TABLE IV: FINAL VERSION OF EVALUATION CRITERIA WITH THEIR RELATED FUNDAMENTAL CONCPETS

Identifier & Evaluation criteria	Related OO Concepts
EC01: Able to identify classes at the proper level of abstraction with regards to the problem being solved (design level)	Class
EC02: Able to identify the proper classes, methods and attributes to solve a particular problem (implementation level)	Class, Method, Abstraction
EC03: Able to give appropriate names/attributes (nouns) and method (verbs)	Class
EC04: Able to create constructors as necessary for a class	Class
EC05: Able to define accessor and mutator methods (i.e. getter and setter methods) as necessary for a class	Method, Class
EC06: Able to send an appropriate message/method call to an object based on the type of that object and the interface of the corresponding method	Message passing, Method, Object
EC07: Able to manipulate heterogeneous container of objects by sending appropriate polymorphic messages to each of them	Object, Method, Message passing, Polymorphism
EC08: Able to pass correctly an object as a parameter in a message	Message passing, Object
EC09: Able to identify the proper level of access control (e.g private, public, protected) for the characteristics and behaviours of a class.	Encapsulation, Class
EC10: Able to identify "is-a" relationships between several related classes and implement inheritance between these classes correctly; these include super and sub-class, constructors, methods that should be inherited and their access control.	Inheritance, Class, Method
EC11: Able to call a method inherited from the ancestors of the class in which the call is made.	Method, Class, Inheritance
EC12: Able to appropriately define multiple related classes as opposed to defining a single class in solving the problem in hand	Inheritance, Class
EC13: Able to create aggregation relationships between related classes to correctly indicate whole-part relationship between the concepts/entities represented by the classes.	Class
EC14: Able to correctly create an object of a class using an appropriate constructor of factory method based on the documentation of the corresponding class.	Object, Class
EC15: Able to identify appropriately situations in which polymorphism can be applied	Polymorphism
EC16: Able to set up correctly a group of objects which work together among themselves in carrying out a certain task (vs. one object doing everything itself).	Object

As shown in Table V, most of the evaluation criteria derived from the Delphi study are backed by heuristics and principles of object-oriented design. Our further work will focus on the development of an assessment tool by integrating

these criteria. The tool will serve as a supporting platform for assessing students ability in implementing object-oriented concepts in their source code.

TABLE V: SUMMARY OF MAPPING EVALUATION CRITERIA WITH OBJECT-ORIENTED DESIGN HEURISTIC AND OBJECT-ORIENTED DESIGN PRINCIPLES

Evaluation Criteria ID	Object-oriented Design Heuristics (Arthur Reid, 1996)	Object-oriented Design Principles
EC01	Heuristics [2.1, 2.8, 2.11, 3.2, 3.6, 3.7, 3.8]	Principles [1]
EC02	Heuristics [2.1, 2.5, 2.6, 2.9, 3.4, 4.6, 4.13, 8.1]	Principles [1, 4]
EC03	Heuristics [3.9]	-
EC04	Heuristics [4.9, 4.10]	-
EC05	Heuristics [3.3, 4.9]	Principles [1]
EC06	Heuristics [2.2, 2.3, 2.7, 3.5, 4.2, 4.3, 4.4]	Principles [2, 3]
EC07	Heuristics [4.5, 4.8, 7.1]	Principles [2, 5]
EC08	Heuristics [4.2, 4.3, 4.4, 4.5, 4.6]	-
EC09	Heuristics [2.4, 2.7, 2.5, 2.9, 5.9]	Principles [1, 2]
EC10	Heuristics [5.1, 5.2, 5.3, 5.4, 5.9, 5.10, 5.11, 5.18, 6.03]	Principles [2, 3, 5]
EC11	Heuristics [2.4, 4.6, 5.10]	Principles [2, 3, 5]
EC12	Heuristics [5.8, 5.14, 5.15]	-
EC13	Heuristics [5.9]	Principles [2]
EC14	Heuristics [4.9, 4.10]	-
EC15	Heuristics [5.9, 5.10, 5.11, 5.12]	Principles [4, 5]
EC16	Heuristics [3.2, 3.3, 3.4]	-

IV. CONCLUSION

It is great importance that students have a good grasp of core object-oriented concepts during their object-oriented programming courses. What are fundamental object-oriented concepts? Searching the literature does not give a conclusive answer. However, there are a number of concepts often mentioned in the object-oriented literature. Based on the study, a few concepts has been identified a fundamental to object-oriented. Abstraction, class, method, message passing, inheritance, object, polymorphism and encapsulation are among the most common ones. These concepts were chosen as the basis for establishing the evaluation criteria for assessing novices ability in applying fundamental object-oriented concepts in their source code as well as to validate the content using a three-round Delphi study. Sixteen validated evaluation criteria were derived based on expert consensus and were further validated by associating them with related object-oriented heuristics and principles. The next stage of the research will concentrate on establishing assessment guidelines, based on those criteria, to assist educators during assessment of students' skills as well as the assessment tools.

REFERENCES

[1] J. Bennedsen, M. Caspersen, and M. Kölling, "Introduction to part i issues in introductory programming courses reflections," *Teaching of Programming*, vol. 4821, pp. 3-5, Heidelberg, Berlin: Springer, 2008.

[2] N. Khamis and S. Idris, "Investigating current object-oriented programming assessment method in Malaysia's universities," in *Proc. ICEEI*, Bandung, 2007.

[3] P. P. Smith, "Techniques of assessment pertinent to computer programming courses," 2005.

[4] M. Kolling, "The problem of teaching object-oriented programming, part 1: Languages," *Journal of Object-Oriented Programming*, 1999.

[5] A. E. Fleury, "Programming in Java: Students-constructed rules," presented at 31st SIGCSE Technical Symposium on Computer Science Education, 2000.

[6] E. Anna and M. Thune, "Novice Java programmers' conception of "object" and "class", and variation theory," presented at ITiCSE, Monte de Caparica, Portugal, 2005.

[7] S. Kate, B. Jonas, E. Anna, M. Robert *et al*., "Student understanding of object-oriented programming as expressed in concept maps," in *Proc. SIGCSE Bull*., vol. 40, no. 1, pp. 332-336, 2008.

[8] S. Kate and T. Lynda, "Checklists for grading object-oriented CS1 programs: Concepts and misconceptions," in *Proc. SIGCSE Bull*., vol. 39, no. 3, pp. 166-170, 2007.

[9] H. Simon, G. Robert, and W. Mark, "Avoiding object misconceptions," in *Proc. SIGCSE Bull*, vol. 29, no. 1, pp. 131-134, 1997.

[10] G. Brown, J. Bull, and M. Pendlebury, *Assessing Student Learning in Higher Education*, London: Routledge, Taylor and Francis, 1997.

[11] M. McCracken,V. Almstrum, D. Diaz, G. Mark, H. Dianne, K. Yifat Ben-David *et al*., "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," in *Proc. SIGCSE Bull*., vol. 33, no. 4, pp. 125-180, 2001.

[12] L. Raymond and L. John, "First year programming: Let all the flowers bloom," in *Proc. the Fifth Australasian Conference on Computing Education*, vol. 20. 2003.

[13] T. A. Elliott and G. Mark, "Developing a validated assessment of fundamental CS1 concepts," in *Proc. the 41st ACM Technical Symposium on Computer Science Education*, 2010.

[14] J. D. Amstrong, "The quarks of object-oriented development," *Communication of ACM*, vol. 49, issue 2, pp. 123-128, 2006.

[15] A. P. Association, "Technical recommendations for psycho-logical tests and diagnostic techniques," *Psychological Bulletin*, vol. 51, pp. 201-238, 1954.

[16] C. C. Hsu and B. A. Sandford, "The Delphi technique: Making sense of consensus," *Practical Assessment, Research & Evaluation*, vol. 12, no. 10, pp. 1-8, 2007.

[17] R. Göb, C. Mc Collin, and M. Ramalhoto, "Ordinal methodology in the analysis of likert scales, quality," *Quantity*, vol. 41, no. 5, pp. 601-626, 2007.

[18] G. Norman, "Likert scales, levels of measurement and the 'laws' of statistics," *Advances in Health Sciences Education*, vol. 15, no. 5, pp. 625-632, 2010.

[19] M. A. Pett, *Non Parametric Statistics for Health Care Research*, London: SAGE Publications, 1997.

[20] F. Clegg, *Simple Statistics*, Cambridge: Cambridge University Press, 1998.
[21] N. Blaikie, *Analyzing Quantitative Data*, London: SAGE Publications, 2003.
[22] M. C. Robert. (1996). Design Principles. [Online]. Available: http://www.objectmentor.com/resources/publishedArticles.html
[23] B. Meyer, *Object-Oriented Software Construction*, Prentice Hall, 1997.
[24] B. Liskov, "Keynote address - data abstraction and hierarchy," presented at Object-Oriented Programming Systems, Languages and Applications (Addendum), 1987.

Norazlina Khamis is a senior lecturer and she is currently attached to the Faculty of Computing and Informatics, Universiti Malaysia Sabah since October 2014. Previously she was a senior lecturer at the Department of Software Engineering, University of Malaya, Kuala Lumpur from 2001 to 2014. She received her MSc in realtime software engineering from Universiti Teknologi Malaysia in year 2001 and her PhD award in computer science from Universiti Kebangsaan Malaysia in year 2012. Her main research interests are in sustainable software engineering, software quality, software engineering education and educational technology.