Electronic Communications of the EASST
Volume 52 (2011)

Proceedings of the 7th Educators' Symposium @ MODELS 2011: Software Modeling in Education (EduSymp2011)

Threshold Concepts in Object-Oriented Modelling

Ven Yu Sien, David Weng Kwai Chong

11 Pages

Guest Editors: Marion Brandsteidl, Andreas Winter
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Threshold Concepts in Object-Oriented Modelling

Ven Yu Sien[1], David Weng Kwai Chong[2]

HELP University[1], Monash University Sunway Campus[2], Malaysia

Abstract: Proponents of the object-oriented (OO) paradigm frequently claim that the OO paradigm is '*more natural*' than the procedural paradigm because the world is filled with objects that have both attributes and behaviors. However students in higher education generally experience considerable difficulty in understanding OO concepts and acquiring the necessary skills in object-oriented analysis and design. This paper proposes OO modelling to be a set of threshold concepts and describes a study that sought to improve undergraduate students' learning of OO modelling by adopting concept maps as 'stepping stones' to facilitate the development of analysis class and sequence diagrams.

Keywords: Threshold concept, OO modelling, Concept maps, Class diagram, Sequence diagram

1 Introduction

Object-oriented analysis and design (OOAD) is not easy to learn, and a particular challenge for students is developing the ability to abstract real-world problems [EMM+06a]. Despite claims made by advocates of the OO approach [ADS+00], [SBG06] novices in OO techniques in general have difficulties understanding OO concepts. Colbert [Col94] has found in his teaching and consulting experience that software developers have problems coping with the OO paradigm, especially with the concept of abstraction. When trying to conceptualise real-world problems as abstractions within the context of OO analysis and design (OOAD), he found that software developers experience particular difficulties in trying to classify real-world objects in generalisation-specialisation hierarchies or whole-part associations.

In this paper we introduce the notion of threshold concepts in OO modelling. A threshold concept or skill is one which when grasped, changes the way in which the learner views the discipline or approaches a task; i.e. it is *transformative*, and is thus likely to be *irreversible* or difficult to unlearn. This is certainly true of OO modelling. The journey across a conceptual threshold is likely to be difficult or *troublesome*, and may involve traversing the conceptual space (recursiveness) until 'the penny drops'. Difficulty may also arise from a concept being counter-intuitive or contrary to common sense. Threshold concepts are *integrative*, bringing together different aspects of a subject that previously appeared unrelated. Finally, a threshold concept delineates a particular conceptual space, i.e. it has *boundaries*, and to move beyond these boundaries the learner needs to grasp other threshold concepts.

The paper is organized as follows: Section 2 describes studies that investigated the problems novices have in understanding and developing OO models; and research on threshold concepts within computing education. In Section 3, we present our proposal on OO modelling to be a set of threshold concepts. Section 4 illustrates some of the difficulties experienced by students in OO modelling and Section 5 provides a strategy on helping students overcome the thresholds. We conclude in Section 6 with a summary of the findings.

2 Background Review

2.1 OO Modelling

During the OOAD phases, models are produced to show the type of information processing that is required of the new system. A model of an OO system is an abstract representation of the system. It represents the problem domain and emphasises some characteristics of the real-world. Modelling a system, however, requires the representation of different perspectives or views of the system and therefore there are different types of diagrams for modelling each of these views. Models used for OO development can take the form of graphics, narratives, or formulae.

Proponents of the OO paradigm frequently claim that it is '*more natural*' than the procedural paradigm because the world is filled with objects that have both attributes and behaviors. Neubauer and Strong [NS02] assume that '*more natural*' implies more intuitive, that is, more easily understood and more consistent with existing patterns of thought. Martin [Mar93] similarly notes that we '*have a natural way of organising our knowledge about the world*' – hence, we will not find it difficult to '*analyse the world in a way that seems natural to human thinking*'.

However, although OO is representative of real-world problems, it does not always reflect the way in which people think [ADS+00], [NS02]. Rosson and Alpert [RA90] reported end-users' conceptual confusion about objects: OO analysts refer to objects which may not be considered 'natural objects' to users. For example, a 'customer' may be intuitively seen as a natural object as 'it' can perform some action. But seeing an 'order' an entity with responsibilities and ability to perform operations is likely to be counter-intuitive, possibly requiring crossing a conceptual threshold. Wirfs-Brock [Wir06] argues that it is a myth that objects in a computer system are representations of real-world entities – the domain concepts identified in the system are at '*best loosely connected to their real-world counterparts*'.

Bolloju and Leung's [BL06] study of errors produced by novices in class and diagrams reported errors to be

- incorrect multiplicities;
- misassigned attributes;
- incorrect usage of generalisation-specialisation hierarchies;
- missing messages;
- missing objects; and
- incorrect delegation of responsibilities.

An OO system consists of interacting objects to fulfill certain responsibilities. This therefore implies that objects are 'animated' in some way. However, Neubauer and Strong [NS02] argue that we do not usually view our world as a collection of animated objects *interacting* with one another to create processes. Instead, we view our world as containing many inanimate objects that we control and manipulate. Similarly, Détienne [Det97] conducted a review of empirical research on OO design (OOD) and discovered that the results of her research do not support the claims made for the naturalness and ease of OOD. Some of her findings are:

- The identification of objects from the problem domain is not easy. The entities that are identified may not necessarily be useful in the design solution.

- The mapping between the problem and programming domains is not easy. The analysis of the problem domain is not sufficient to structure the solution in terms of objects.

2.2 Threshold Concepts

The notion of threshold concepts was first presented in 2002 at the Tenth Improving Student Learning Conference in Brussels, and its conceptual framework emerged from the United Kingdom Economic and Social Research Council funded project 'Enhancing Teaching and Learning Environments in Undergraduate Courses' [DE05]. Seminal work within economics posited that certain concepts e.g., cost and elasticity held by economists to be central to the mastery of their discipline could be described as threshold concepts. Subsequent work has largely focussed on development of the conceptual framework and the identification of threshold concepts in various disciplines e.g.,

- complex numbers and limits in mathematics [ML03];
- confidence intervals in statistics [CB06]; and
- evolution in biology [TC07].

The following candidate threshold concepts within computing education research have been proposed:

- OO programming (OOP). This topic has been known to be both difficult to teach and to learn. Boustedt et al. [BEM+07] identified this to be a broad area within which thresholds exist. First-year students [ET05] who were interviewed after their first OOP course, reported that they found OOP to be troublesome to learn (it took a long time to 'click'), the knowledge gained is irreversible (once understood the OO paradigm cannot be forgotten) and transformative (knowledge gained can be transferred to another programming language).
- Pointers. Boustedt et al. [BEM+07] identified this to be another threshold concept. First-year students reported in [ET05] they found pointers to be troublesome to learn (difficult to understand), the knowledge gained is irreversible (unforgettable); and integrative and transformative (knowledge gained can be used in other subjects).
- Abstraction. Eckerdal et al. [EMM+06b] identified this to be a candidate threshold concept as the ability to abstract and to be able to move from one level of abstraction to another is a key skill in computer science. In a later paper, Mostrom et al. [MBE+08] suggested that abstraction per se may not be a threshold concept – however, specific forms of abstraction e.g., modularity, data abstraction, object-orientation, etc. can be considered as threshold concepts.
- OO at its most basic – including classes, objects and encapsulation [EMM+06b]. First-year students [BEM+07] found basic OO troublesome to learn. Learning OO is transformative (it requires a change in mind-set when viewing OO as representative of real-world problems) and integrative.
- Recursion. Rountree and Rountree [RR09] proposed recursion to be a candidate threshold concept. Many novice programmers have found recursion to be an example of troublesome knowledge (they have difficulty with the concept of `self reference'). However, once the students have grasped this concept, the understanding is irreversible (new understanding cannot be unlearnt); and transformative and integrative (the student is able to identify relationships with other materials e.g. all loops can be expressed as

recursion). Recursion is a boundary marker for both Software Engineering and Theoretical Computer Science (it is useful in programming, and also defines what is computable).

3 OO Modelling: Thresholds in OOAD

Undergraduate IT students have in general found difficulty in grasping OO concepts and the role that UML diagrams play in the analysis and design phases of a software development lifecycle. It has been observed that students frequently produce class and sequence diagrams that are incomplete [BL06], [SBG06]. Some of these problems were discussed in Section 2. Most of the students are unable to effectively build class diagrams from the problem domain because they essentially do not know '*what*' to model.

The class diagram is fundamental to the object modelling process, representing the key concepts and relationships (e.g., generalisation-specialisation hierarchy, aggregation, composition and association) in the problem domain of the system. The class diagrams produced during the OO analysis are a logical model and shows classes of objects that are required to represent the problem domain. The sequence diagram is a kind of interaction diagram produced during the OO design phase and describes the flow of messages between objects. It is useful in describing the dynamic design, the invocation of methods, and the order of invocations.

Several aspects of OO modelling are consistent with the defining criteria for threshold concepts. The conceptual grasp of OO modelling is difficult to unlearn, it is a solid basis for effective application, and clearly distinguishes between practitioner and learner. Simultaneously, OO modelling is difficult both to teach and learn: it represents troublesome knowledge (concepts appear alien and counter intuitive and initially difficult to understand), not least because of the requirement for abstraction. Modelling is an important activity in system development: Models represent the system at different levels of abstraction. OO modelling transforms and integrates understanding of analysis and design of information systems. This transformative learning produces advanced learners who understand the complex notions underlying the development of information systems. Topics such as systems analysis and design, software engineering and advanced topics in OO programming languages are likely to have little meaning to students who do not grasp OO modelling.

It was previously discussed in Section 2.2, that OOP, abstraction and OO are broad areas within which thresholds exist. Based on data analysed from student interviews, Boustedt et al. [BEM+07] argued that more specific threshold concepts in OOP '*might include the way in which objects work together (i.e. concurrency), or the ability to see large problems as composed of a set of small sub-problems*'. Eckerdal et al. [EMM+06b] noted in their experiment that both lecturers and students considered OO as a threshold concept, but the authors argued that OO is too broad an area and some of the more specific threshold concepts could be polymorphism or object interactions.

OO modelling is also clearly a key concept within OOAD, but OO modelling may represent a family or group of associated thresholds including:

- classes;
- generalisation-specialisation hierarchies; and
- object interactions.

4 Difficulties in OO Modelling

This section presents some examples of the types of problems that novices face when producing UML class and sequence diagrams:

- identifying classes from functional requirements;
- assigning attributes to a class;
- selecting appropriate generalisation-specialisation hierarchies for classes; and
- assigning responsibilities to objects to realise a use case.

Participants in our study were fifty-one Year 2 IT undergraduate students enrolled in a BSc (Hons) programme in Computing at two universities in Kuala Lumpur, Malaysia. They worked on a Vehicle Rental case study containing four expanded use cases (detailed descriptions of processes) that describe the functional requirements of the system. The participants were asked to individually produce an analysis class diagram based on all the expanded use cases. Rubrics were used to describe assessment criteria for evaluating the appropriateness of diagrams, which were assessed for appropriate classes, attributes, associations and multiplicities.

4.1 Analysis of Class and Sequence Diagrams

The class diagram is the key artefact in the analysis phase as its *appropriateness* can have a significant impact on the design of the overall system.

4.1.1 Classes

Liu et al. (2003) observed that the identification of classes is difficult even for experienced analysts and OO developers, citing three contributory factors-

- complexity, vagueness and ambiguity of natural language;
- lack of the domain knowledge and OO experience; and
- the absence of effective OO methods and well-developed guidelines.

Expected Classes. Table 1 shows the number of *expected classes* that participants were able to identify in their class diagrams. Only 8% identified all eight expected candidate classes, 35% identified seven classes and 2% identify only one appropriate class.

Table 1. Analysis of class diagrams with expected classes

Classes[1]							
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2%	2%	8%	14%	10%	22%	35%	8%

4.1.3 Generalisation-Specialisation Hierarchies

The role of the generalisation-specialisation hierarchy is usually explained by showing examples of its use in models that feature common classification hierarchies e.g. Employee, PartTimeEmployee and FullTimeEmployee. While this approach seems simple, novices usually find defining hierarchies difficult. Detienne [Det97] finds that novices try to use inheritance as often as they can, but frequently use it incorrectly, especially when identifying the static characteristics (attributes) of the superclass and subclasses. She also finds that

[1] Total is not 100% because of rounding errors.

novices tend to use the generalisation-specialisation hierarchy to express a whole-part association. In our study we expected participant to identify a generalisation-specialisation hierarchy as exemplified in Fig. 1.

.

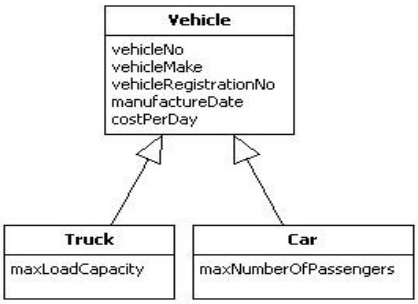

Figure 1. Appropriate generalisation-specialisation hierarchy

We found that participants' class diagrams did not include many generalisation-specialisation hierarchies (Table 2). Only 45% of participants defined appropriate inheritance hierarchies in their class diagrams; and 6% had a mixture of appropriate and inappropriate hierarchies.

Table 2. Analysis of class diagrams with generalisation-specialisation hierarchies

Inappropriate Use of Inheritance Hierarchy	Appropriate Use of Inheritance Hierarchy	Inheritance Hierarchies not Defined
12%	45%	49%

4.1.4 Object Interactions

Students generally have difficulty identifying messages to be sent in UML sequence diagrams. They do not know how to fulfil the responsibilities of the use case by getting objects to pass messages to each other. Students also have difficulty understanding that the interaction diagrams are dependent on the analysis class diagram in terms of its classes, associations and multiplicities.

A sequence diagram focuses on the time ordering in which messages are sent. It is useful for describing the order of invocation of methods. Only 40% of sequence diagrams displayed some evidence of responsibilities delegated to the appropriate objects. None of the student diagrams fulfilled all the responsibilities of the use case. 52% of sequence diagrams did not include any parameters in the messages – we do not however consider this a serious design fault as parameters can be optionally defined in UML.

4.2 Discussion

'Completeness of a design is concerned with the fact that the presence of information in some diagram requires the presence of other information in another part of the design. For instance, if there is a use-case that describes some system functionality, then there should also be a collection of classes that provides this functionality. If some information that we can deduce from available diagrams is not present in a design, then there is an incompleteness in the design' [LCM+03]. By description models produced by our study participants are incomplete

because there are missing elements in the class and sequence diagrams that have not been defined to represent the requirements of the system.

5 Evaluation: Changes in Levels of Understanding

5.1 Concept Mapping

Concept mapping, a tool for facilitating learning, was developed by Joseph Novak [NC06] at Cornell University in 1972, and is commonly used for visualising relationships between concepts. Within the context of OOAD we use concept maps to graphically present fundamental concepts and their inter-relationships within a problem domain.
Our study (Section 4) initially investigated the difficulties undergraduate students have when producing UML class and sequence diagrams. The results indicate the fundamental problems that learners have with OO modelling, and suggest that learners have difficulty with certain concepts which may represent thresholds in understanding. In order to address these threshold concepts, a concept-driven approach is developed to help novices produce more appropriate UML class and sequence diagrams. The effectiveness of this approach is evaluated by three different experiments.
A static concept map diagrammatically describes the structure of a system by illustrating its component concepts and their inter-relationships: The concepts thus defined model classes and attributes in an analysis class diagram. A static concept map is constructed by identifying concepts and their relationships from expanded use cases, and is built incrementally from the use cases. Rules for producing a static concept map and transforming it to a class diagram are defined in [SC07].
A dynamic concept map provides a dynamic view of the system behaviour by showing the key responsibilities that need to be fulfilled by specific concepts in order to fulfil a particular scenario of a use case. The concepts defined in the dynamic concept map model *objects* in the sequence diagram. For each use case, its key responsibilities are identified and added to the static concept map so as to produce a dynamic concept map. Rules for producing a static concept map and transforming it to a class diagram are defined in [SC08].

5.2 Helping Students Through Thresholds

Two studies [Sie10] have reported the effects of using concept mapping to assist learners in OOAD produce class and sequence diagrams. Some findings of the first study (‘Study 1’), including common faults found in UML diagrams, are described in Section 4; participants in this study were not taught any concept mapping techniques. In Study 2 [Sie10] twenty-one Year 2 IT undergraduate students were taught the concept mapping techniques and this study found a statistically significant reduction in the number of faults produced in UML class and sequence diagrams, particularly in terms of:

- identification of expected classes representing the key concepts in the problem domain;
- assignment of attributes to appropriate classes;
- identification of appropriate generalisation-specialisation hierarchies; and
- assignment of responsibilities to fulfil a particular scenario of a use case.

However the results achieved by Study 2 participants may not be attributed solely to the effect of concept mapping – other contributory factors to quality improvement include:

- As the students in Study 2 were currently enrolled in an OOAD course, their knowledge of OOAD concepts and experience in OO modelling was likely to be fresh in their minds.
- The students in Study 2 may have been given a better foundation on OOAD concepts.
- Students in Study 2 were given more time to produce the concept maps and UML models.

Participants responded positively to the use of concept maps: This is consistent with research [Roy08] reporting the successful use of concept maps in teaching. The results presented in [Sie10] lead us to conclude that concept mapping has a positive impact on class and sequence diagram, and suggest that concept mapping effectively helps learners understand OO modelling. In particular the use of specifically defined labelled links (in the static concept maps) helps learners distinguish between classes and attributes, and more easily identifies relationship types. Other particular benefits are that

- mapping is relatively easy to teach; the two types of notations solely used are nodes and links [NC06];
- maps help clarify the meaning of concepts using propositions [NC06];
- substantial guidelines for producing concept maps have been developed. These are defined in [SC07], [SC08].

6 Conclusion

In this paper we have proposed the following topics to be considered threshold concepts in OO modelling:

- Classes. In general, students find this topic to be troublesome (the students experience difficulties in identify appropriate classes from the problem domain; and assigning appropriate attributes to classes). The knowledge gained is irreversible (once understood, it cannot be easily forgotten) and transformative and integrative (knowledge gained can assist in developing the logical design of databases).
- Generalisation-specialisation hierarchies. Novices usually find defining hierarchies troublesome (they have problems grouping real-world objects in terms of classification because they are not used to grouping objects in hierarchies). This is not intuitive enough to be mastered without training [RA96]. The knowledge gained is irreversible (once the concept is understood, students will invariably find it easier to identify generalisation-hierarchies). This is transformative and integrative as this hierarchy can be used in use case diagrams and data base models.
- Object interactions. Svetinovic et al. [SBG06] found some common errors committed by students when producing interaction diagrams:
  - assignment of a large business activity to a single object whilst it should be fulfilled through the collaboration with other objects;
  - missing responsibilities that should be assigned to objects; and
  - missing objects that should participate in the overall responsibilities.

  This topic is known to be both difficult to teach and troublesome to learn. Again, once grasped the knowledge is irreversible, integrative and transformative.

Meyer and Land [ML03] suggest that once a student has been introduced to a threshold concept, he/she enters a state of '*liminality*'– a state associated with being 'stuck' and not possessing a mastery of the concept – until the necessary transformation of understanding has

taken place and the 'threshold' is crossed. Students who have problems producing analysis class diagrams will first need to cross the threshold in identifying appropriate classes (representing real-world objects) before they can assign attributes and relationships (e.g. associations, generalisation-specialisation hierarchies and whole-part hierarchies) to the classes. These thresholds have to be crossed before the students can successfully produce appropriate sequence diagrams.

Studies have been presented to support the adoption of concept maps as aids to novices for crossing the threshold of OO modeling. Participants in experiments conducted by the first author showed significant improvement in class diagram construction (after the participants have been taught the concept mapping techniques), in overcoming *troublesome* knowledge in the identification of appropriate classes, generalisation-specialisation hierarchies, and the appropriate assignment of responsibilities to objects. The knowledge gained is irreversible (once understood, it cannot be easily forgotten) and transformative and integrative (support and improve their skills of abstract thinking, and consequently improve their understanding of OOAD).

Bibliography

[ADS+00] R. Agarwal, P. De, A.P. Sinha, M. Tanniru. 'On the Usability of OO Representations. Communications of the ACM 43(10), pp. 83-89, 2000.

[BEM+07] J. Boustedt, A. Eckerdal, R. McCartney, J.E. Mostrom, M. Ratcliffe, K. Sanders, C. Zander. Threshold concepts in computer science: do they exist and are they useful? In *Proc. 38th SIGCSE Technical Syymposium on Computer Science Education*. 2007.

[BL06] N. Bolloju & F. Leung. Assisting Novice Analysts in Developing Quality Conceptual Models with UML. Communications of the ACM 49(7), pp. 108-112, 2006.

[Col94] E. Colbert. Abstract Better and Enjoy Life. Journal of Object-Oriented Programming (JOOP), 7(1), 1994.

[CB06] C.J. Cope & G. Byrne. Improving Teaching and Learning about Threshold Concepts: The example of Confidence Intervals. In *Proc. Threshold Concepts within the Disciplines Symposium.* 2006.

[DE05] D. Hounsell & N. Entwistle. Enhancing Teaching-Learning Environments in Undergraduate Courses. http://www.etl.tla.ed.ac.uk/docs/ETLfinalreport.pdf

[Det97] F. Détienne. Assessing the Cognitive Consequences of the Object-Oriented Approach: A Survey of Empirical Research on Object-Oriented Design by Individuals and Teams. Interacting with Computers, 9(1), pp. 47-72, 1997.

[EMM+06a] A. Eckerdal, R. McCartney, J.E. Moström, M. Ratcliffe, C. Zander. Can Graduating Students Design Software Systems? In *Proc. 37th SIGCSE Technical Symposium on Computer Science Education*. 2006.

[EMM+06b] A. Eckerdal, R. McCartney, J.E. Moström, M. Ratcliffe, K. Sanders, C. Zander. Putting Threshold Concepts into Context in Computer Science Education. In *Proc. 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 2006.

[ET05] A. Eckerdal, M. Thuné . Novice Java Programmers' Conceptions of "object" and "class", and Variation Theory. In *Proc. 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 2005.

[LCM+03] C. Lange, M.R.V. Chaudron, J. Muskens, L.J. Somers, H.M. Dortmans. An Empirical Investigation in Quantifying Inconsistency and Incompleteness of UML Designs. In *Proc. 2nd Workshop on Consistency Problems in UML-Based Software Development*. 2003.

[Mar93] J. Martin. *Principles of Object-Oriented Analysis and Design.* Prentice-Hall International, Inc., Englewood Cliffs, New Jersey.

[MBE+08] J.E. Moström, J. Boustedt, A. Eckerdal, R. McCartney, K. Sanders, L. Thomas, C. Zander. Concrete Examples of Abstraction as Manifested in Students' Transformative Experiences. In Proc. *4th International Workshop on Computing Education Research*. 2008.

[ML03] J.H.F. Meyer, R. Land. Threshold Concepts and Troublesome Knowledge (1): Linkages to Thinking and Practising within the Disciplines. Improving Student Learning: Improving Student Learning Theory and Practice - Ten Years On. Oxford: OCSLD, pp. 412-424, 2003.

[NC06] J.D. Novak, A.J. Cañas. The Origins of the Concept Mapping Tool and the Continuing Evolution of the Tool. Information Visualization 5(3), pp.175-184, 2006.

[NS02] B.J. Neubauer & D.D. Strong. The Object-Oriented Paradigm: More Natural or Less Familiar? J. Comput. Small Coll., 18 (1), pp. 280-289, 2002.

[RA90] M.B. Rosson & S.R. Alpert. The Cognitive Consequences of Object-Oriented Design. Human-Computer Interaction, vol. 5, 1998.

[RA96] C. Ryan & G. Al-Qaimari. A Cognitive Perspective on Teaching Object Oriented Analysis and Design. http://citeseer.ist.psu.edu/cache/papers/cs/3179/http:zSzzSzgoanna.cs.rmit.edu.auzSz~ghassanzSzrmit.pdf/a-cognitive-perspective-on.pdf

[RR09] J. Rountree, N. Rountree. Issues Regarding Threshold Concepts in Computer Science. In *Proc. 11th Australasian Conference on Computing Education.* 2009.

[Roy08] D. Roy. Using Concept Maps for Information Conceptualization and Schematization in Technical Reading and Writing Courses: A Case Study for Computer Science Majors in Japan. In *Proc. IEEE International Professional Communication Conference*, 2008.

[SBG06] D. Svetinovic, D.M. Berry, MW Godfrey. Increasing Quality of Conceptual Models: Is Object-Oriented Analysis that Simple? In *Proc. 2006 International Workshop on Role of Abstraction in Software Engineering*. 2006.

[SC07] V.Y. Sien, D. Carrington. A Concepts-First Approach to Object-Oriented Modelling. In *Proc. Third IASTED International Conf on Advances in Computer Science and Technology*. 2007.

[SC08] V.Y. Sien, D. Carrington. Using Concept Maps to Produce Sequence Diagrams. In *Proc. IASTED International Conference on Software Engineering*. 2008.

[Sie10] V.Y. Sien, Teaching Object-Oriented Modelling using Concept Maps. In *Proc. 6th Educators' Symposium: Software Modeling in Education at MODELS 2010*.

[TC07] C. Taylor & C.J. Cope. Are There Educationally Critical Aspects in the Concept of Evolution? In *Proc. of UniServe*, 2007.

[Wir06] R.J. Wirfs-Brock. Looking for Powerful Abstractions [Object Oriented Technology]. Software, IEEE, 23(1), pp. 13-15, 2006.