2013

A Model for Measuring Knowledge Constructions of Students in Online Discussions

Pornpat Sirithumgul
Claremont Graduate University, sirithumgul.p@gmail.com

Lorne Olfman
Claremont Graduate University, lorne.olfman@cgu.edu

Information Systems: Transforming the Future

24th Australasian Conference on Information Systems, 4-6 December 2013, Melbourne

Proudly sponsored by

ACIS 2013 Principal Sponsor

A Model for Measuring Knowledge Constructions of Students in Online Discussions

Pornpat Sirithumgul, Lorne Olfman
Social Learning Software Laboratory
Center for Information Systems and Technology
Claremont Graduate University
{pornpat.sirithumgul, lorne.olfman}@cgu.edu

Abstract

In this paper, we propose a conceptual idea for measuring knowledge in online discussions. The measurement we propose is based on technical terms of any ontology that students use for describing concepts related to each other. We conducted a simulation experiment, an experiment examining online documents that discuss our example concepts. The experiment demonstrated that the relationships linking concepts affect the relationships of concepts' technical terms, which we can use to evaluate knowledge and understanding of students.

Keywords

Knowledge measurement, online discussions, knowledge construction, technical terms, word relationships

INTRODUCTION

Online discussion technologies have been increasing in importance recently partly because they can support social learning. There are several studies focusing on improving the techniques for promoting productive learning of students through online discussions (Choi et al. 2005; Hu and Yang 2005; MacKnight 2000; Mandernach 2006; Pena-Shaff and Nicholls 2004; Schellens et al. 2007; Veerman and Veldhuis-Diermanse 2001; Veldhuis-Diermanse 2002; Weinberger and Fischer 2005; Yang et al. 2005 and Zhu 1996). Most research (Choi et al. 2005; Hu and Yang 2005; Schellens et al. 2007 and Yang et al. 2005) suggests techniques, especially Socratic techniques, in encouraging students to have productive discussions leading to knowledge construction. Some research (Gunawardena et al. 1997; Pena-Shaff and Nicholls 2004; Veerman and Veldhuis-Diermanse 2001; Veldhuis-Diermanse 2002; Weinberger and Fischer 2005 and Zhu 1996) focuses on building measures for assessing productive discussions of students from their online posts.

Content analysis is a popular technique which is applied to measure how productive students' discussions are in three aspects – cognitive presence, critical thinking and knowledge construction. The content analysis technique has been enhanced in detail over time in order to reflect performance of students and effectiveness of instructional factors in online discussions. However, content analysis is a time-consuming method, which requires knowledge and experience of instructors in the evaluation process.

In this research, we propose a conceptual idea for building a quantitative measure used as an indicator of knowledge construction of students in online discussions. Our knowledge measurement concept is proposed to be less complex but more objective than the evaluation methods based on content analysis, which will help reduce biases generated from different evaluators.

RELATED WORK

This work is developed from the research in the area of content analysis aiming to measure knowledge construction of students in online discussions. Some of the important references in this domain include: Gunawardena et al. (1997), Pena-Shaff and Nicholls (2004), Veerman and Veldhuis-Diermanse (2001), Veldhuis-Diermanse (2002), Weinberger et al. (2005) and Zhu (1996).

Gunawardena et al. (1997) proposed a content analysis model, which was used for measuring knowledge construction via the process of social negotiation. The researchers suggested that in online discussions, five phases of interaction may occur – phase I (sharing/comparing of information), phase II (the discovery and exploration of dissonance or inconsistency among ideas, concepts, or statements), phase III (negotiation of meaning/co-construction of knowledge), phase IV (testing and modification of proposed synthesis or co-construction), and phase V (agreement statements/applications of newly constructed meaning). The researchers also proposed 21 indicators for evaluating knowledge occurring in those phases.

Pena-Shaff and Nicholls (2004) proposed a method for analyzing student interactions and knowledge construction in computer bulletin board discussions. For knowledge construction measurement, the researchers

proposed the categories of online discussions that students may have. Indicators were also proposed for each category for evaluating knowledge that students earned while discussing. The discussion categories include questions, reply, clarification, interpretation, conflict, assertion, consensus building, judgment, reflection, support and other (mixed messages difficult to categorize).

Veerman and Veldhuis-Diermanse (2001) aimed to measure knowledge construction generated from collaborative learning of students via computer-mediated communication in academic courses. Four considerations were employed to analyze messages posted by students – new idea; a relevant message that was not mentioned before; explanation, a message used for refining or elaborating a prior message; and evaluation, a message discussing strength/weakness of a prior message or justifying a prior message.

Veldhuis-Diermanse (2002) developed a method used for analyzing learning processes of students in computer supported collaborative learning (CSCL) and also developed a coding scheme for measuring the quality of constructed knowledge based on the structure of the observed learning outcome (SOLO) taxonomy of Biggs and Collis (1982). There are four levels of knowledge construction proposed in this method – level D (students can identify and define a new point that has not been defined yet), level C (students can list, organize or classify things that should be in the same group), level B (students can provide reasons for a choice they make, elaborate ideas/concepts, including link, apply and compare/contrast ideas/concepts/theories they learn), and level A (students can critique/justify and generalize ideas).

Weinberger and Fischer (2005) proposed a framework for measuring knowledge construction from argumentative discussions of students via the text-based online discussion boards. There are four dimensions for this framework – the participation dimension, the epistemic dimension, the argument dimension, and the dimension of social modes of co-construction. In the process of knowledge evaluation, two levels of measurement are applied – micro-level and macro-level. Messages produced by students were divided into micro-segments, segments of concepts which could be related to each other; and macro-segments, segments grouping subsequent micro-segments together.

Zhu (1996) proposed a framework for evaluating knowledge of students from the discussion messages students sent via e-mail and Vaxnotes, electronic conferencing software. The coding scheme of this framework consists of three parts – participant categories, types of interaction and note categories. There were seven classes of notes defined in this work – meaning as reflective notes, comments, discussions, answers, information sharing notes, scaffolding notes and questions. Zhu (1996) divided the question type into two sub-types – information-seeking questions and discussing questions. The researcher assumed that the answer to the first-type question could be key quotes, basic facts, procedural knowledge, and stories from one's readings; and the answer to the second-type question could be elaboration on discussion topics, exchanges of thoughts or ideas on related concepts and issues, personal understandings, and topic-related discussion questions.

Although content analysis is a widely-used method in this research area, we found that reliability of content coding is a crucial issue for this method. Only some previous studies (Veerman and Veldhuis-Diermanse 2001; Veldhuis-Diermanse 2002; Pena-Shaff and Nicholls 2004 and Weinberger and Fischer 2005) reported how the researchers handled the issue. In this study, we avoid the difficulty of content analysis by proposing an objective method which can measure concept clarification, an objective indicator reflecting knowledge construction of students. Concept clarification is used as an indicator of knowledge construction in the current study because we observed the previous studies and found that although the knowledge construction measures vary by measurement criteria, concept clarification is the aspect the studies shared in common. We, therefore, assume that concept clarification is a foundation of knowledge construction. The lexical cohesion model (Halliday and Hasan, 1976) is applied together with an object-oriented model in the current study to measure concept clarification.

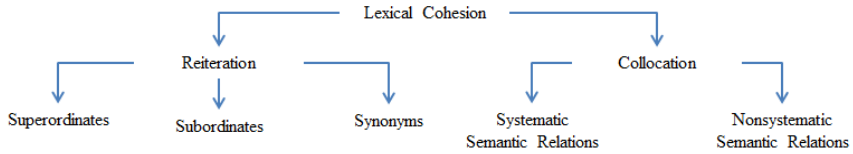


Figure 1: Lexical cohesion model

According to Halliday and Hasan (1976), lexical cohesion means sentences, topics or phrases, each of which can be categorized by meanings or the relationship of meanings. There are two main types of lexical cohesion proposed by Halliday and Hasan (1976) − reiteration and collocation. Reiteration and collocation cohesion types can be classified into sub-cohesion types as shown in Figure 1.

Reiteration is a type of word relationship consisting of (1) the relationship of sentences sharing the same word(s), including sentences using identity of references, e.g., pronouns; and (2) the relationship of sentences related to

each other in forms of superordinates, subordinates or synonyms. Words having superordinates and subordinates relationships are the words linking together with the concepts of generalization and specialization. For example, the word fruit is the general idea of the specific words such as peach and apple. Collocation is the relationship of co-occurring words. There are two types of collocation – systematic semantic and nonsystematic semantic. Systematic semantics include ordered sets, e.g., the set of {one, two, three}; unordered sets, e.g., the set of {red, blue, green}; and words having a part-and-whole relationship, e.g., the set of {face, eyes, nose, mouth}. A nonsystematic semantic relationship includes words having high potential of co-occurrence in the same context. However, these words co-occur because they share the same conceptual idea of the real world, rather than a semantic relationship such as the co-occurrence of the words garden and digging. In the present work, we modify the lexical cohesion model to describe concept relationships. The detail of model modification is described in the Research Methodology section.

RESEARCH METHODOLOGY

The current study adapts the lexical cohesion model to describe concept relationships. Although the previous model (see Figure 1) is suitable for presenting word relationships, it needs to be adapted to serve the variety of relationships among concepts. Figure 2 shows a concept of software testing called black-box testing and its related concepts in the lexical cohesion model. The relationships of all concepts in Figure 2 are further expanded in our model as shown in Figure 3.

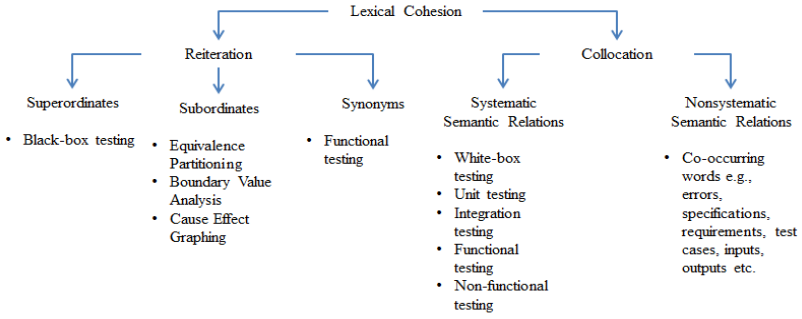


Figure 2: An example of applying the lexical cohesion model

Figure 2 shows an application of the lexical cohesion model to black-box testing, a concept we use as an example in our study, and other relevant testing types of black-box testing; *equivalence partitioning, boundary value analysis,* and *cause effect graphing* are specific types (subordinates) of black-box testing. *Functional testing* is an interchangeable name for black-box testing (we call this term an interchangeable term rather than a synonym of black-box testing because it is a term that software testers use when they apply black-box testing to testing functions of software, but some software testers use this term only to represent black-box testing). Other relevant testing concepts in systematic semantic relations can be used to clarify the focal concept, e.g., *white-box testing*, *unit testing*, *integration testing*, *non-functional testing*, and other technical terms in the group of nonsystematic semantic relations, which are used to describe the concept of black-box testing, e.g., *errors*, *specifications*, *requirements*, *test cases*, *inputs*, *outputs*, etc. Although the lexical cohesion model can be used to describe a concept, it cannot be used to specify some concepts having different relationships to the focal concept, especially the relevant concepts in systematic semantic relationships. For example, white-box testing should be described in the model as an opposite concept of black-box testing, or functional testing and non-functional testing should be described in the model as composite concepts of black-box testing as well as unit testing, and integration testing should be described in the model as a concept that inherits part of its characteristics from black-box testing (the other part is inherited from white-box testing). In addition, the previous model does not suggest how co-occurring words, the words in nonsystematic semantic relationships, are defined, especially when several concepts are related to each other.

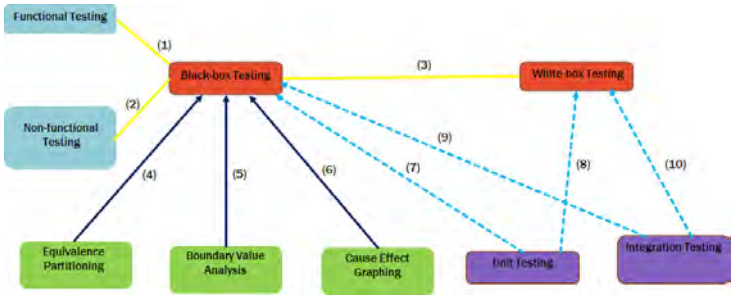


Figure 3: The proposed model

To fill the gaps we found, the lexical cohesion model is applied together with an object-oriented model as shown in Figure 3. The object-oriented model is applied because object-oriented relationships can represent most relationships of real-world concepts. In this study, three object-oriented relationships are included – association relationship or mutual relationship (relationships (1), (2) and (3) in Figure 3), *IS-A* relationship (relationships (4), (5) and (6) in Figure 3) and *HAS-A* relationship or *PART-OF* relationship (relationships (7), (8), (9) and (10) in Figure 3).

The association relationship represents the relationship of two types of concepts – opposite concepts and composite concepts. Thus for black-box testing, the opposite concept is white-box testing, and the composite concepts are functional testing and non-functional testing. Functional testing and non-functional testing are considered composite concepts of black-box testing because they are two different applications to which black-box testing can be applied; especially for functional testing, the term may be used as an interchangeable term of black-box testing or used to represent that black-box testing is used to test software functions. Non-functional testing is another composite concept of black-box testing because black-box testing can be applied to serve non-functional requirements of software.

The *IS-A* relationship is used to describe the relationship between a general concept and specific concept in our model. Black-box testing, for example, is a general concept of its specific testing concepts – equivalence partitioning, boundary analysis and cause-effect graphing. The relationships between black-box testing and its specific testing concepts also show that each specific type of testing inherits some characteristics from black-box testing, but every specific testing type has its own testing technique.

A *HAS-A* relationship is an object-oriented relationship that represents an object containing another object. From our example, unit testing and integration testing have multiple *HAS-A* relationships with black-box testing and white-box testing; from the other view, black-box testing and white-box testing have multiple *PART-OF* relationships with unit testing and integration testing. The relationships also show that unit testing and integration testing inherit the techniques of black-box testing and white-box testing while they have their own characteristics of testing.

From the three relationships, three assumptions regarding the technical terms used to describe all concepts and their relationships are derived as follows:

Assumption 1: A concept in an association relationship may share some technical terms with another concept in the same relationship. White-box testing, an opposite concept of black-box testing, may share some technical terms with black-box testing because those technical terms may be used to compare or distinguish the two concepts from each other. Functional and non-functional testing, the composite concepts of black-box testing, may share some technical terms with black-box testing because functional and non-functional testing are composite methods of black-box testing. The technical terms that are used to describe black-box testing may be also used to describe the composite methods.

Assumption 2: A specific concept in a *IS-A* relationship may inherit its technical terms from a general concept of the same relationship. The three specific types of black-box testing – equivalence partitioning, boundary analysis, and cause-effect graphing may inherit not only the technique of black-box testing, but also the technical terms used in black-box testing. However, each specific testing type should have its own technical terms reflecting its special testing technique.

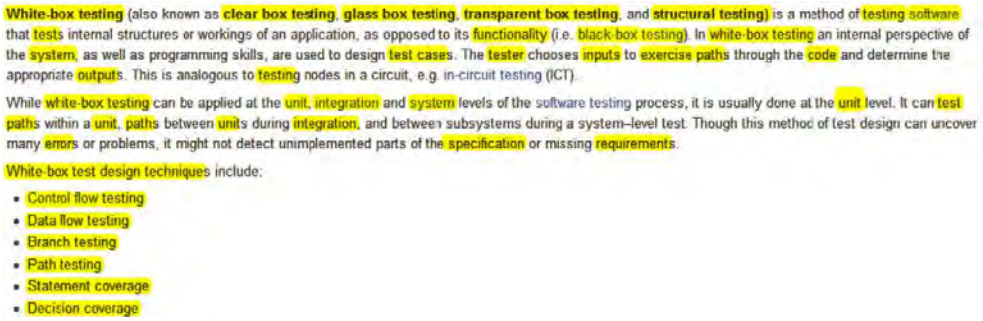
Assumption 3: A specific concept having multiple relationships with other general concepts may inherit its technical terms from the related general concepts. Unit testing and integration testing, for example, have *HAS-A* relationships with black-box testing and white-box testing. It may be possible that some technical terms of unit testing and integration testing are inherited from black-box testing and white-box testing. However, unit testing and integration testing should have their own technical terms showing their testing techniques.

This study aims to learn how knowledge is constructed by proving the three assumptions. The knowledge measure we propose in this work is derived from the result of the experiment we conduct to prove the assumptions. If the experimental result proves that *assumption 1* is correct, we will have a principle for measuring knowledge in the discussions of composite and opposite concepts. If the experimental result proves that *assumption 2* is correct, we will have a principle for measuring knowledge in the discussions of the concepts having general-and-specific relationships. If the experimental result proves that *assumption 3* is correct, we will have a principle for measuring knowledge in the discussions of the concepts, which inherit their characteristics from multiple concepts.

EXPERIMENT

We manually collected documents discussing the nine types of software testing through the Google search engine. The nine concepts of software testing were used as keywords in searching – black-box testing, white-box testing, functional testing, non-functional testing, equivalence partitioning, boundary value analysis, cause-effect

graphing, unit testing and integration testing. For the search results of each keyword, ten documents were collected and technical terms in the documents were extracted and examined by a computer program designed and coded by the first author. The documents we collected are the web pages of Wikipedia, blogs and software testing tutorial pages. Some documents were not included in our experimental data such as presentation files because we wanted to prove our assumptions from the documents containing complete sentences for describing concepts.



White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

Figure 4: An excerpted section of a search result document

Figure 4 shows an excerpted section of a Wikipedia page (http://en.wikipedia.org/wiki/White-box_testing -- last accessed 07/24/2013) describing the concept of white-box testing. The highlighted words are the technical terms (the terms used in software testing) that we extracted from the document to examine in our experiment.

To prove *assumption 1* and *assumption 2*, the technical terms of six testing concepts – white-box testing, functional testing, non-functional testing, equivalence partitioning, boundary value analysis and cause-effect graphing were examined. The probability of technical terms used to describe each concept occurring in their respective documents are calculated and compared with the probability of the same technical terms occurring in black-box testing documents. To prove *assumption 3*, the technical terms used in the documents of unit testing and integration testing were examined by comparing the probability of the technical terms occurring in each testing type with the probability of the same technical terms used in black-box testing and white-box testing documents.

From Figure 5, functional testing is used as an example for describing the algorithm we use in proving *assumption 1* and *assumption 2*. In our experiment, six testing concepts – white-box testing, functional testing, non-functional testing, equivalence partitioning testing, boundary value analysis and cause effect graphing are examined by the algorithm shown in Figure 5 where the detail of each step is clarified.

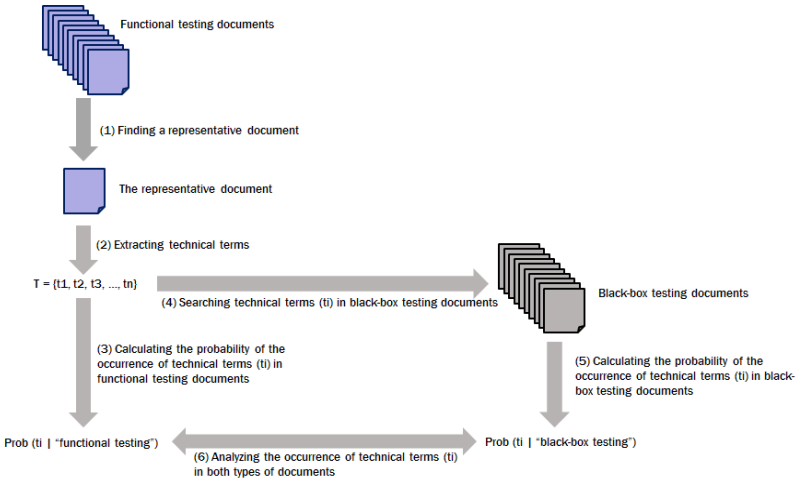


Figure 5: Research algorithm for assumptions 1 & 2

***Step 1:* Finding a representative document of a given testing concept**

One out of ten of search result documents is selected to be a representative document at this step. The representative document is selected as the document that is similar to the most documents. In this work, *Cosine similarity* shown in equation (1) is used to find similarity between two documents. The variables – x and y represent two different document vectors in the equation.

$$COS(x, y) = \frac{x, y}{\| x \| \| y \|} \quad (1)$$

A representative document is the document having the maximum average similarity. The average cosine similarity (AVG_k) of any document k is shown in equation (2) and the maximum average similarity is shown in equation (3); the average similarities of all documents (document x_k where k = 1 to 10) are compared to find the maximum average similarity of all documents.

$$AVG_k = \frac{\sum_{i=1}^{10}(COS\ (x_k, y_i))}{10} \quad (2)$$

$$\text{The maximum average similarity} = \arg\max_k\ (AVG_k) \quad (3)$$

$$\text{The maximum average similarity} = \arg\max_k\ (AVG_k) \quad (3)$$

Step 2: **Technical terms extraction**

The technical terms in the representative document are extracted by using the software testing glossary of the International Software Testing Qualifications Board (ISTQB, 2012).

Step 3: **Calculating the probability of occurrence of technical terms in the documents of the given concept**

From Figure 5, the set of technical terms T is $\{t_1, t_2, ..., t_n\}$ where n is the number of different technical terms occurring in the representative document of the given concept, and N is the total frequency of all technical terms occurring in all documents of the given concept. The probability of the occurrence of a technical term t_i (i = 1 to n) equals to the frequency of the technical term t_i divided by N. The probability is shown in equation (4).

$$Prob\ (ti\ |"functional\ testing") = \frac{Count\ (ti)}{N} \quad (4)$$

$$Prob\ (ti\ |"functional\ testing") = \frac{Count\ (ti)}{N} \quad (4)$$

Step 4: **Searching the occurrence of the technical terms in black-box testing documents**

The technical terms that are extracted in *step 2* are used to search their occurrence in the black-box testing documents. The frequency of each technical term t_i is collected and used to calculate the probability of the occurrence in *Step 5*.

Step 5: **Calculating the probability of the occurrence of the technical terms in black-box testing documents**

At this stage, the probability of technical terms t_i occurring in black-box testing documents is calculated, and the value from the calculation will be analyzed in *Step 6*. The probability of the occurrence of a technical term t_i (i = 1 to n where n is the number of different technical terms occurring in *Step 3*) equals to the frequency of the technical term t_i occurring in black-box testing documents divided by N_{black}, the total frequency of all technical terms occurring in black-box testing documents. The probability is shown in equation (5).

$$Prob\ (ti\ |"black - box\ testing") = \frac{Count\ (ti)}{Nblack} \quad (5)$$

Step 6: **Analyzing the occurrence of the technical terms in both types of documents**

At this stage, the probability of the occurrence of the technical terms we calculate in *Step 3* and *Step 5* is compared and analyzed. The probability of the technical terms occurring in the documents of the concepts – white-box testing, functional testing, non-functional testing, equivalence testing, boundary value analysis and cause-effect graphing is compared against the probability of the same technical terms occurring in black-box testing documents. The result of this step is reported in the Experimental Results section, and an analysis of this step is provided in the Discussion section.

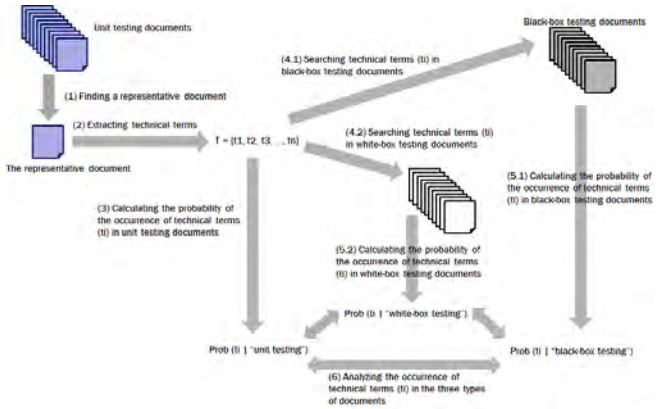


Figure 6: Research algorithm for assumption 3

To prove *assumption 3*, the technical terms of two testing concepts – unit testing and integration testing are examined. *Step 1* through *Step 6* in our algorithm are repeated to prove the assumption, but an additional process is added in *Step 4*, *Step 5* and *Step 6* as shown in Figure 6. In *Step 4*, the technical terms are searched in black-box testing documents and the additional search for the same technical terms is also done in white-box testing documents. In *Step* 5, the probability of the technical term occurrence is calculated for black-box testing and white-box testing documents, and in *Step 6*, the probability of the technical terms occurring in unit testing/integration testing documents is analyzed and compared against the probability of the technical terms occurring in black-box testing and white-box testing documents.

EXPERIMENTAL RESULTS

The results of *Step 6* in our experimental plan are reported in this section. In Table 1, the list of the specific testing types is shown in the column *Title*. The number of different technical terms occurring in the representative documents of the specific testing types is shown in the column # *of terms*. The three columns – # *of frequent terms*, # *of common terms* and # *of unique terms* show the results of comparing the probability of the technical terms occurring in the specific testing documents with that occurring in the black-box testing documents. The column # *of frequent terms* shows the number of the technical terms of which occurrence probability in the specific testing documents is 100 percent or greater than the occurrence probability in the black-box testing documents. The column # *of common terms* shows the number of the technical terms of which the difference between the occurrence probability in the specific testing documents and in the black-box testing documents is less than 100 percent. The column # *of unique terms* shows the technical terms occurring only in the specific testing documents, not in the black-box testing documents.

Table 1. Comparison of the technical terms occurring in specific testing vs. black-box testing

Title	# of terms	# of frequent terms (%)	# of common terms (%)	# of unique terms (%)
White-box testing	40	11(27.50%)	19 (47.50%)	10 (25%)
Functional testing	17	6 (35.30%)	9 (52.94%)	2 (11.76%)
Non-functional testing	35	14 (40%)	6 (17.14%)	15 (42.86%)
Equivalence partitioning testing	6	2 (33.33%)	2 (33.33%)	2 (33.33%)
Cause-effect graphing testing	17	6 (35.29%)	8 (47.06%)	3 (17.65%)
Boundary value analysis testing	12	5 (41.67%)	6 (50%)	1 (8.33%)

Table 2 is similar to Table 1, but the columns # *of frequent terms*, # *of common terms* and # *of unique terms* show the results of comparing the probability of the technical terms occurring in the specific testing documents – unit testing and integration testing – with that occurring in the documents of black-box testing and white-box testing. The column # *of frequent terms* shows the number of the technical terms of which occurrence probability in the specific testing documents is 100 percent or greater than the occurrence probability in the documents of black-box testing and white-box testing. The column # *of common terms* shows the number of technical terms of which the difference between the occurrence probability in the specific testing documents and in black-box and white-box testing documents is less than 100 percent. The column # *of unique terms* shows the technical terms occurring only in the specific testing documents, not in black-box testing and white-box testing documents.

Table 2. Comparison of technical terms occurring in specific testing, and black-box and white-box testing

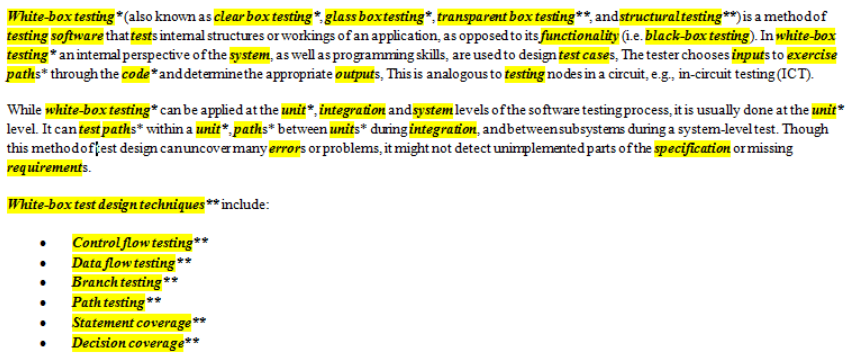
Title	# of terms	# of frequent terms (%)	# of common terms (%)	# of unique terms (%)

Unit testing	42	8 (19.05%)	33 (78.57%)	1 (2.38%)
Integration testing	47	11 (23.40%)	30 (63.83%)	6 (12.77%)

From Table 1 and Table 2, we can see that the shared terms (frequent terms and common terms) occur more than fifty percent of all technical terms. Especially for unit testing and integration testing, the shared terms occur in both documents much more than the unique terms. Among the shared terms of unit testing, 35 out of 41 are shared between with black-box testing and white-box testing, one is shared only with black-box testing, and five are shared only with white-box testing. Among the shared terms of integration testing, 34 out of 41 are shared with black-box testing and white-box testing, and seven are shared only with white-box testing. From all technical terms occurring in unit testing, we found that 27 (about 64%) are shared with integration testing, and from all technical terms occurring in integration testing, we found that 26 terms (about 55%) are shared with unit testing. That unit testing and integration testing share more than half of all technical terms validates *assumption 1* in that the relationship of opposite concepts can be observed from technical terms that the concepts share. The detail of shared terms of opposite concepts is discussed in the Discussion section.

The difference between frequent terms and common terms we found in this study is that most frequent terms can be used to distinguish a specific testing type from its referred testing (black-box testing in Table 1, and black-box testing and white-box testing in Table 2) while we can rarely do that by using common terms. For example, the frequent terms of white-box testing – *code*, *unit*, *path*, and *coverage*, can be used to distinguish white-box testing from black-box testing better than its common terms – *functionality*, *input*, *output*, *result*, *system*, *test*, *tester*, *testing*, *error*, *software*, and *test case*.

We also found that most frequent terms and unique terms of all types of the specific testing reflect the specific characteristics of the testing. Some of those terms are synonyms or the terms that reflect the testing name itself, e.g., *white-box testing*, *structural testing*, and *glass-box testing* (for white-box testing), *functional testing* and *functionality* (for functional testing). Most of the frequent and unique terms we found reflect specific techniques/processes of the testing, e.g., *control flow testing*, *data flow testing*, *path testing*, *decision coverage*, *statement coverage*, *code*, *unit*, *path*, *coverage*, and *complexity* (in white-box testing), *load testing*, *performance testing*, *security testing*, *efficiency*, and *robustness* (in non-functional testing), *input domain* and *input value* (in equivalence partitioning testing), *cause-effect diagram*, *decision table*, and *state transition* (in cause-effect graphing testing), *negative testing*, and *boundary value* (in boundary value analysis testing), *bug*, *unit*, and *test suite* (in unit testing) and *branch*, *integration*, *module*, *driver*, *stub*, *entry criteria*, *exit criteria*, *bottom-up testing* and *top-down testing* (in integration testing). Based on the example in Figure 4, the technical (highlighted) terms we extracted and used are classified to the groups of frequent terms (marked by *), common terms (not marked) and unique terms (marked by **) as shown in Figure 7.



White-box testing* (also known as ***clear box testing****, ***glass box testing****, ***transparent box testing****** , and ***structural testing******) is a method of ***testing software*** that ***tests*** internal structures or workings of an application, as opposed to its ***functionality*** (i.e. ***black-box testing***). In ***white-box testing**** an internal perspective of the ***system***, as well as programming skills, are used to design ***test cases***, The tester chooses ***inputs*** to ***exercise paths**** through the ***code**** and determine the appropriate ***outputs***, This is analogous to ***testing*** nodes in a circuit, e.g., in-circuit testing (ICT).

While ***white-box testing**** can be applied at the ***unit****, ***integration*** and ***system*** levels of the software testing process, it is usually done at the ***unit**** level. It can ***test paths**** within a ***unit****, ***paths**** between ***units**** during ***integration***, and between subsystems during a system-level test. Though this method of test design can uncover many ***errors*** or problems, it might not detect unimplemented parts of the ***specification*** or missing ***requirements***.

White-box test design techniques*** include:

- ***Control flow testing******
- ***Data flow testing******
- ***Branch testing******
- ***Path testing******
- ***Statement coverage******
- ***Decision coverage******

Figure 7: An example of frequent terms, common terms, and unique terms in an excerpted document

DISCUSSION

In this section, the experimental results are discussed in three aspects – (1) the analysis of shared terms (common and frequent), (2) the validity of the three assumptions we proposed, and (3) the conceptual idea we propose for measuring knowledge in online discussions.

From the experimental results in Table 1 and Table 2, we found two types of shared terms – common terms and frequent terms. A common term occurring in a document can be in one of the following groups:

(i) A technical term that relevant concepts share and use frequently. The difference of occurrence of this group of common terms among relevant concepts is so small that we cannot use them to distinguish relevant concepts from each other. For example, the common terms – *test*, *tester*, *testing*, *system*, *result*, *software* are the common terms that black-box testing shares with its related concepts.

(ii) A common term is a technical term used to describe an opposite concept of a focused concept. The common terms in this group are used to describe a focused concept indirectly. For example, *black-box testing* and *functionality* are common terms in the white-box testing documents because they are used to compare the concept of black-box testing and white-box testing, not to describe the concept of white-box testing directly.

A frequent term is a technical term that relevant concepts share, but it occurs with a specific concept more often than other relevant concepts. A frequent term occurring in a document can be in one of the following groups:

(i) A technical term that relevant concepts share and use in general. However, it can occur more often in a specific concept than its relevant concept because it can be used to reflect the characteristic/technique/process of the specific concept. For example, *path*, *unit*, *code* occur with the concepts of white-box and black-box testing, but the technical terms are used in white-box testing more often than in black-box testing because they are related to the technique of white-box testing more than the technique of black-box testing.

(ii) A frequent term is a technical term linking a general concept and a specific concept together. A frequent term in this group occurs when a technical term is used in a general concept and a specific concept. However, the frequent term occurs in the specific concept documents more often than in the general concept documents. For example, the technical terms *equivalence partitioning*, *cause-effect graph* and *boundary value analysis* are frequent terms that equivalence partitioning testing, cause-effect graphing testing and boundary value analysis testing share with black-box testing. The frequent terms in this case occur because black-box testing mentions the specific testing concept in its document, but the frequent terms occur in the documents of specific testing methods where the frequent terms are discussed more often than in the document of black-box testing.

From the experimental results and the analysis of shared terms, the three assumptions can be clarified. For *assumption 1*, we assume that concepts related to each other by an association relationship may share some technical terms. From the experimental results and the analysis of shared terms, we found that opposite concepts, e.g., white-box and black-box testing, and unit testing and integration testing share common terms in groups (i) and (ii) and frequent terms in group (i). For composite concepts, e.g., functional testing and non-functional testing, we found that the composite concepts share common terms in group (i) and frequent terms in group (i).

For *assumption 2*, we assume that two concepts related to each other by a *IS-A* relationship (the relationship of general and specific concepts) may share some technical terms and each specific concept should have its own technical terms reflecting its specific technique. From the experimental results of equivalence partitioning testing, cause-effect graphing testing and boundary value analysis testing in Table 1, and the analysis of shared terms, we found that the general and specific concepts share the common terms in group (i) and the frequent terms in groups (i) and (ii). The technical terms reflecting specific techniques of a specific concept can be observed from frequent terms and unique terms in this case.

For *assumption 3*, we assume that a specific concept having multiple relationships with several general concepts may share technical terms with those general concepts and the specific concept should have its own technical terms reflecting its specific technique. From the experimental results in Table 2, we found that unit testing and integration testing share common terms and frequent terms with black-box testing and white-box testing. From the analysis of shared terms, we found that general and specific concepts share common terms in group (i) and frequent terms in groups (i) and (ii). The technical terms reflecting specific techniques of a specific concept can be observed from frequent terms and unique terms in this case.

In all types of concept relationships, frequent and unique terms can be used to distinguish any two concepts from each other. From our experiment, we found that most frequent terms and unique terms include terms that reflect specific characteristics/techniques/processes of concepts. In measuring knowledge construction of students in online discussions, common terms can tell us that students are discussing in a scope of a given concept. However, the common term itself cannot tell how much students understand the given concept. Unique terms and frequent terms can be used to prove whether or not students have the depth of understanding in this case.

From the other view, common terms, frequent terms, and unique terms can be used to measure knowledge in two dimensions – breadth and depth. The breadth dimension occurs in the case of composite concepts – the relationships of black-box testing to functional testing and non-functional testing, for example. We can check how much students understand the concept of black-box testing from the technical terms they use. If students use the common terms that functional testing and non-functional testing share with black-box testing, e.g., *test*, *tester*, *testing*, *system*, *result*, *software*, we may conclude that students have a general idea about black-box testing, but if students include frequent terms and unique terms of functional testing, e.g., *specification*, *functional testing*, *test data* and *functionality*, and non-functional testing, e.g., *load testing*, *non-functional testing*, *reliability* and

performance in their discussions, we may conclude that students understand the application area of black-box testing, which means the students understand the concept more than the general idea.

The depth dimension occurs in the case of general-and-specific relationships, e.g., the relationships between black-box testing and the specific testing types of black-box testing (equivalence partitioning testing, boundary value analysis testing and cause-effect graphing testing). If students use the common terms that the specific testing concepts share with black-box testing, e.g., *test*, *tester*, *testing*, *system*, *result*, *software*, *test case*, *functionality*, we may conclude that students have a general idea about black-box testing, but if students include frequent terms and unique terms of the specific testing concepts, e.g., *input domain*, *input value*, *equivalence partitioning*, *decision table*, *cause-effect graph*, *boundary value* in their discussions, we may conclude that students understand the specific techniques of black-box testing, which means the students understand the concept of black-box testing deeper than the general idea.

A multiple-inheritance relationship is a complicated view of the depth dimension. From the example in this study, if students can discuss the technical terms of black-box testing and white-box testing, we may conclude that students understand the general concepts of black-box testing and white-box testing. However, if students include the frequent terms and the unique terms of unit testing, e.g., *unit testing*, *bug*, *test suite* and integration testing, e.g., *branch*, *module*, *integration*, *driver*, *stub*, we may imply that students understand the level of testing to which black-box testing and white-box testing can be applied.

CONCLUSION

In this paper, we propose a conceptual idea for measuring knowledge constructions of students in online discussions. The result of our study shows that concepts are related to each other by the relationships of their technical terms, which we can apply to measure knowledge in breadth and depth dimensions. Our measuring method can be further developed to be an automatic tool used for evaluating the quality of students' discussions. This tool will complement the previous methods, which require human efforts and time cost. Finally, our approach can be applied to any ontology for which standard technical terms are defined. More than using it for evaluating knowledge constructions, our approach can be applied to find the relationships of concepts and technical terms in the glossary of any ontology as well.

REFERENCES

Choi, I., Land, S. M., and Turgeon, A. J. 2005. "Scaffolding peer-questioning strategies to facilitate metacognition during online small group discussion," *Instructional Science*, 33, 483-511.

Gunawardena, C.N., Lowe, C.A. and Anderson, T. 1997. "Analysis of a global online debate and the development of an interaction analysis model for examining social construction of knowledge in computer conferencing," *Journal of Educational Computing Research*, 17 (1997), pp. 397–431

Halliday, M. and Hasan, R. 1976. *Cohesion in English.* Longman Group.

Hu, B. and Yang, J. 2005. "Analyzing critical thinking and factors influencing interactions in online discussion forum," In The fifth IEEE international conference on Advanced Learning Technologies (ICALT' 05).

International Software Testing Qualifications Board (ISTQB). 2012. Retrieved 22 July, 2013, from http://www.istqb.org/

MacKnight, C. 2000. "Teaching critical thinking through online discussions," *Educause Quarterly*, 4, 38-41. Retrieved October 23, 2012 from http://eac595b.pbworks.com/f/macknight+2000+questions[1].pdf

Mandernach, B. 2006. "Thinking critically about critical thinking: Integrating online tools to promote critical thinking," *InSight: A Collection of Faculty Scholarship*, 1, 41-50. Retrieved 23 October, 2012 from http://www.insightjournal.net/Volume4/IntersectionScholarshipTeachingLearningOnlineCourseDesignTeacherEducation.pdf

Pena-Shaff, J. B., and Nicholls, C. 2004. "Analyzing student interactions and meaning construction in computer bulletin board discussions," *Computers & Education*, 42, 243–265.

Schellens, T., Keer, H., Wever, B. and Valcke, M. 2007. "Scripting by assigning roles: Does it improve knowledge construction in asynchronous discussion groups?" In *Computer-Supported Collaborative Learning. Springer.* 2, 225-246. DOI = 10.1007/s11412-007-9016-2

Standard glossary of terms used in Software Testing. 2012. Retrieved 15 July, 2013, from http://www.istqb.org/downloads/finish/20/101.html

Veerman, A., and Veldhuis-Diermanse, E. 2001. "Collaborative learning through computer-mediated communication in academic education," In Euro CSCL 2001 (pp. 625–632). Maastricht: McLuhan institute, University of Maastricht.

Veldhuis-Diermanse, A. E. 2002. "CSCLearning? Participation, learning activities and knowledge construction in computer-supported collaborative learning in higher education." Unpublished doctoral dissertation. Wageningen Universiteit, Nederland. Retrieved 15 August, 2004, from http://www.gcw.nl/dissertations/3187/dis3187.pdf.

Weinberger, A., and Fischer, F. 2005. "A framework to analyze argumentative knowledge construction in computer supported collaborative learning," *Computers & Education*, 46(1), 71–95.

Yang, Y. C., Newby, T. J., and Bill, R. L. 2005. "Using Socratic questioning to promote critical thinking skills through asynchronous discussion forums in distance learning environments," *The American Journal of Distance Education* 19(3), 163-181.

Zhu, E. (1996). "Meaning negotiation, knowledge construction, and mentoring in a distance learning course." In: Proceedings of selected research and development presentations at the 1996 national convention of the association for educational communications and technology. Indianapolis: Available from ERIC documents: ED 397 849.

COPYRIGHT