ISSN 2278-3091
Volume 9, No.1.4, 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse8191.42020.pdf
https://doi.org/10.30534/ijatcse/2020/8191.42020

The Development of MCQ generating system based on Ontology Concepts

Noor Hasimah Ibrahim Teo[1], Nurul Diyana Mohd Nor[2], Nurul Hidayah Mat Zain[3], Nor Aiza Moketar[4]
[1]Universiti Teknologi MARA, Malaysia, shimateo@uitm.edu.my
[2]Universiti Teknologi MARA, Malaysia, nuruldiyanamn@gmail.com
[3]Universiti Teknologi MARA, Malaysia, nurul417@uitm.edu.my
[4]Universiti Teknologi MARA, Malaysia, noraiza1@uitm.edu.my

ABSTRACT

Education system needs an effective method to evaluate students in a competent way on their major concepts learned from their studies. Generating Multiple Choice Questions (MCQ) process is cumbersome and needs a lot of effort especially among novice instructors. The current MCQ generators do not automatically generate questions but only randomly display questions from the question bank. However, it is not easy to load a massive number of questions in the question banks where it is normally done in stages to have massive question collection in the question bank. Therefore, this paper proposes an automatic question generator specifically for MCQ to enable a massive number of questions generated in just a few seconds. In addition, it is useful for students who have to do self-learning for their tutorial exercises especially during the COVID -19 pandemic crisis where online learning is widely used to replace the traditional face-to-face sessions. The generator has adopted an ontological approach for question generation strategies, and it is implemented using rule-based reasoning. There are six modules proposed and discussed in this paper including the concept extraction from ontology, concept, and question stem mapping, generating appropriate answer options using ontology relation information and reordering of answer options. The functionality and validation test showed that Multiple Choice Question Generator (MCQ-G) can generate MCQ with appropriate answer options. This work will be extended in the future using the difference domain ontology and question types.

Key words: Automated Assessment, Covid-19, Multiple Choice Question, Ontology, Operating Systems, Question Generation.

1. INTRODUCTION

Question generation is a process that requires human's ideas from someone that has sufficient knowledge or expertise in that particular area of domain. Question generation is a task that affects many aspects of people's lives [1]. Asking questions can also help students to know their knowledge gaps and make them search for information to compensate for these gaps [2][3].

Question generation is designed by the institution's instructors who are experts in the specific field. Usually, an instructor creates the questions manually and sometimes he/she also depends on the question banks. The instructor must study all related information/contents through other sources such as books, research articles, internet videos, and some preliminary knowledge. The instructor should also know how to apply the knowledge to solve problems developed for these questions. Most of the college and university papers' levels are determined from students' past performance.

In the present, instructors do not need to generate questions manually since it will burden them and it is time-consuming [4]. Therefore, automated question generator is developed, which is more flexible for them to choose the subjects. The system also has difficulty levels for the question generation. This system is also more secure than generating questions manually and the question paper can be created faster even a few minutes before the examination which it can avoid from any paper leakage [3]. Question generation also involves the creation of reasonable questions from an input, which can be structured and unstructured [5]. The goal of generating questions is to create questions paper based on existing knowledge webs. However, there is still lack of MCQ generator for education subject especially in Computer Science. Most of the MCQ generator evaluates on general domain questions.

Thus, in this research MCQ-G is developed to generate MCQ that specific on computer science field, in which Operating System subject is used for experimental purposes. The proposed system will be able to create question based on predefined

question stem and using an ontological approach for selecting answer options. The aim of MCQ-G is to generate massive MCQ question repository to allow a question to be selected from a bigger range of questions that indirectly promote dynamic question generation. Dynamic, in this case, refers to massive choices of questions for selection at one time; thus, more different question set can be generated.

The automatic question generation features benefit online learning in twofold. First, as an instructor, it reduces the workload to create a quiz, tutorial, or exercise questions for students to work on. They can spend their time in preparing and updating their learning contents especially after the COVID-19 pandemic crisis, most of the classes are still going to be conducted via online mode. Second, as for the students, it will give an advantage in their self-learning by answering many questions.

There are different types of questions used for assessment purpose, this includes MCQ, true/false, short answers, matching, etc. In this paper, MCQ is chosen to be generated because MCQ is quite difficult to be created manually. The basic structure of MCQ includes question stems and answer options. There are two answer options which are correct and incorrect answers. In this work, the single correct answer type of MCQ is developed.

2. ONTOLOGY FOR EDUCATIONAL PURPOSES

Domain ontology has been widely used in the various domains including biomedical [6], cloud service platform [7] and web service [8] that are created using a difference ontology editor. The difference ontology editor is discussed in [9]. However, there is still lack of resources for domain ontology specifically to cater to educational assessments.

The ontologies in computer science that are publicly available is the Operating System (OS) ontology ontology which is used in this work. The ontology of Operating System [10], is an ontology for a standard undergraduate operating system course. The ontology is built using the semantic Web Ontology Language (OWL), and the concepts are extracted from four sources which include three textbooks [11] [12] [13] and lecture notes from one of the authors. It is reported that the ontology has more than one thousand concepts that spread into six parts; operating system overview, process management, storage management, I/O systems, Distributed Systems and Protection and Security.

In this work, difference operating system ontologies are used. This ontology proposed in [14] contains 97 concepts and are spread into nine parts; concurrency control, file systems, fundamental of an operating system, I/O system, Linux system, memory management, process and thread management, protection and security, and system software. Table 1 shows the concepts and its number of sub-concepts for ontology discussed in [14].

Table 1: Main Concepts and its number of sub-concepts in OS Ontology

No.	Concepts	No. of Sub Concepts
1	Concurrency control	11
2	File systems	7
3	Fundamental of operating system	6
4	I/O_ systems	19
5	Linux system	7
6	Memory management	11
7	Process and thread management	6
8	Protection and security	15
9	System software	6

3. PRIOR WORK ON ONTOLOGY-BASED MCQ GENERATION SYSTEM

Ontology elements such as classes, instance and properties are exploited to automatically generate multiple choice questions (MCQ), true/false(T/F), fill-in the blank (FIB), short answer questions, and long answer questions. However, substantial research efforts have been made in the generation of MCQ based on ontology domain knowledge (concept and relationship) as the source of knowledge.

The ontology-based MCQ has begun with the work proposed by [15] in 2008 which proposed three question generation strategies namely class-based strategy, terminology-based strategy, and property-based strategy. The question generation strategy focuses on the generation of appropriate answer options for MCQ that use “Choose the correct sentence” question stem. It has been reported that the class-based strategy generates the least amount of questions compared to property-based strategy for evaluation on five domain ontologies. MCQ generation has also been proposed in [16] by extending work from [15] to increase the difficulty of questions using annotation-based stem with the assumption that a greater similarity in answer options will increase difficulties of selecting a correct answer. The OntoQue using a semantic-based approach to generate MCQ from four domain ontologies is presented in [17]. The key answer for MCQ was chosen from the highest similarity index, while distractors are chosen from the less similarity index to control question difficulty. The result obtained showed the high precision of good question generated on three of the ontologies. The work exploiting concept and stem similarity to control question difficulty was also been presented in [18]. Two similarity measures SubSim() and GrammarSim() were proposed to control question difficulty. The result showed that out of 50 questions evaluated by three experts, 46 of the questions were reported as useful questions.

4. METHODOLOGY

This research conducted using standard System Development Life Cycle methodology. There are five phases involved in this research.

Phase 1
The first phase is data requirement gathering where suitable course ontology and suitable question templates are identified.
Phase 2
In the second phase, the analysis of concept in selected ontology is validated to ensure suitability of the concepts with keywords represented in Operating System Subject.
Phase 3
The third phase is the most critical phase for MCQ-G development, which is system design. The detailed system design is discussed in section 5.
Phase 4
The fourth phase involved MCQ-G development work on web-based platform using Java programming language.
Phase 5
The last phase is testing the functionality of feature in the system and validity of the questions output generated. The result is then reported based on features exist on the system as well as the different output cases.

5. MCQ-G SYSTEM DESIGN

This section discusses the system design for MCQ-G which includes the system architecture, OS ontology representation, algorithm design and interface design.

5.1 System Architecture

The development of MCQ-G is conducted in three phases as shown in Figure 1.

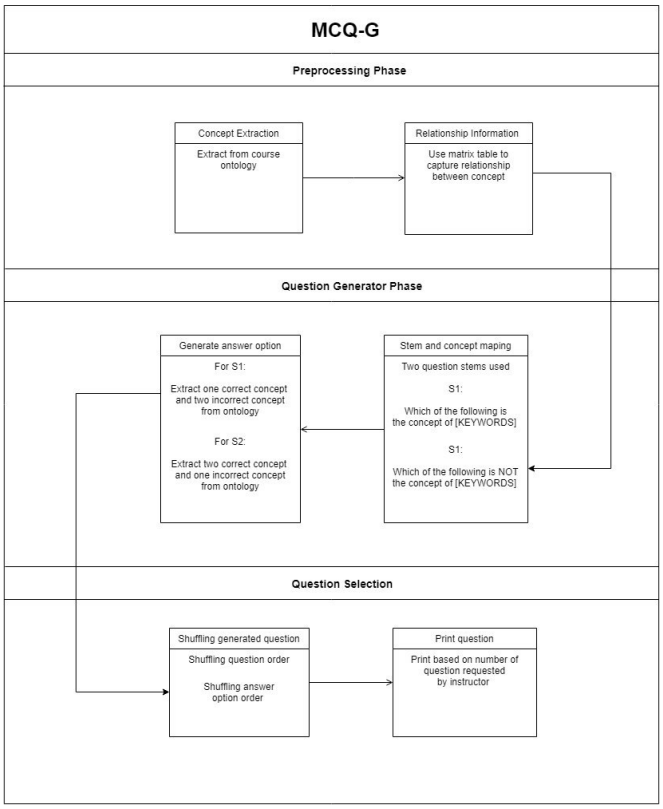

Figure 1: System architecture

The first phase is pre-processing of ontology elements to extract concepts that will be assigned as a [KEYWORD] for the question stems. The relationship between concepts in ontology is kept in the form of a matrix table. For example, in the OS ontology, the relationship between concept virtual memory and memory management as shown in Figure 2 is "Virtual memory is a sub-concept of Memory management". Thus, this information is kept in the matrix table with value 1 to indicate there is a relationship. The algorithm design for the extraction and keeping relationships in the matrix table is discussed in section 4.3.

The second phase is question generation where [KEYWORD] from question stems is instantiated with a concept from ontology. Information from the matrix table in the first phase is used to obtain correct and incorrect answer options. The strategy to select answer options follows the question generation strategy suggested in [15] where relationship information between concepts are used. For example, the question stem "Which of the following is the concept of Memory management?". By referring to Figure 2, the correct answer option is extracted from the sub-concept of *Memory management* and the two incorrect answer options are extracted from sub-concept of other concepts that are of the same level hierarchy with *Memory management* which Figure 2 refers to the *File system* or *Protection and security*. The choice of one correct answer will be randomly selected from the range of *Memory management* sub-concept (*virtual memory* or *paging*), while the two incorrect answer options are taken from sub-concept of the *File system* or *Protection and security*.

Subsequently, for the question stems, "Which of the following is the concept of Memory management?". By referring to Figure 2, the correct answer option is extracted from the sub-concept that is not under *Memory management* and the two incorrect answer options are extracted from the sub-concept of *Memory management*. The choice of one incorrect answer will be randomly selected from the range of other concepts.

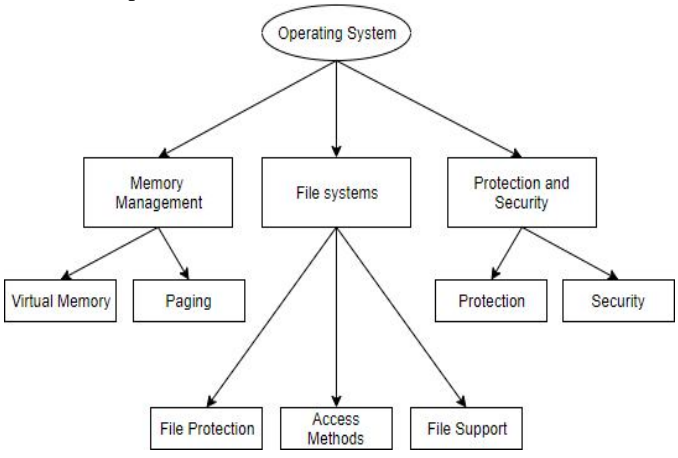

Figure 2: Snapshot of OS ontology concept relation

In the third phase, question selection will be implemented to generate questions randomly based on the number of questions requested by an instructor in MCQ-G. Before the question is generated, the order of answer options is shuffled, to display the difference order of correct and incorrect answer options. The position of answers options of the correct and incorrect answers will be changed each time the new request is made.

In the next section, the structure of the OS ontology used in the first and second phase is discussed.

5.2 OS Ontology Representation

The ontology representation is stored in the notepad in the form of "Every [concept 1] [relationship] [concept 2]". This representation is extracted from the OWL file using *FluentOntarion* editor. The example of the OS ontology representation is shown in Figure 3.

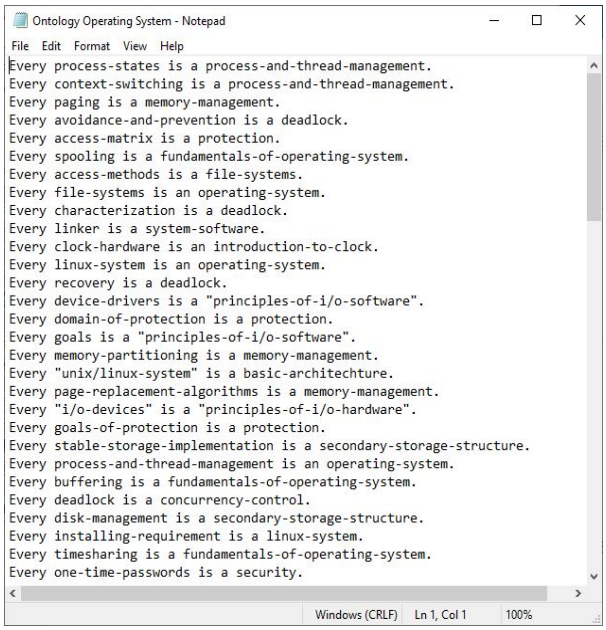

Figure 3: OS Ontology Representation

5.3 Algorithm Design

This section discusses the algorithm design for five functions created in MQQ-G. The algorithm is designed to automatically generate MCQ from the Operating System domain ontology. Each generated question consists of a question stem and three answer options (one correct answer and two distractors). Two question stems used in the experiment are as follow:

- S1: Which of the following is the concept of `[KEYWORD]?`
- S2: Which of the following is not the concept of `[KEYWORD]` ?

There are 6 modules created in MCQ-G as follows:

A. Extracting concepts from OS domain ontology.

First is to extract all concepts in the OS ontology into an array of concepts to enable pre-processing of the concepts to generate MCQ. The converted ontology file in .txt format is read into array `arrConc` and split into five tokens based on the ontology file structure in Figure 4. The split token for "Every Virtual machine is a Memory Management" in ontology .txt file is as follows:

Ontology (in .txt)	*Every*	*Virtual machine*	*is*	*a*	*Memory management*
token	token1	token2	token3	token4	token5

Figure 4: MCQ-G ontology file structure

The important token extracted here is from token1 and token5 which is the ontology concept used as KEYWORD in question stems S1 and S2 while token3 and token4 are the relation between token2 and token5. The concepts of token2 and token5 are then stored in different arrays `arr1` and `arr2` respectively. The pseudocode is as follows:

```
Start
while (read from a text file)
  add into arrConc
  for i less than arrConc size
    declare String Tokenizer
    while (String Tokenizer has more token)
      String T1 = token1;
      String T2 = token2;
      String T3 = token3;
      String T4 = token4;
      String T5 = token5;
      add T2 into arr1
      add T5 into arr2
    end while
  end for
end while
End
```

B. Creating a matrix table to keep a relationship value between two concepts.

Next, the matrix table is created to store the relationship information between concepts T2 and T5. Elements in arr1 and arr2 are combined in `arrComb` to create list data for row and column of the matrix table. There are 97 unique concepts in OS ontology that create matrix table `arr2D` of size 97 x 97. The *LinkedhashedSet* from Java function is used to create this matrix table. The relationship is then inserted into the

matrix table with value to indicate the relationship exists and 0 for no relation. The pseudocode is as follows:

```
Start
Declare int arr2D [97][97]
for i less than arrConc size
  Declare int loc1 and loc2 equals to 0
  for j less than arrComb size
    if (token2 equals (arrComb get j))
      loc1 equals j
    endif
    if (token5 equals (arrComb get j))
       loc2 equals j
    endif
    arr2D[loc1[loc2] equals 1
  end for
end for
End
```

C. Mapping ontology concept with question stem

Question is generated based on two question stems S1 and S2 as written at the beginning of this section. The [KEYWORD] in question stems is first instantiated with concepts from ontology.

D. Generate answer options

The answer options are attached to the question stem. This function is divided into two components: i. For correct answer and ii. For incorrect answers.

For S1, the correct answer option is taken from the sub-concept of [KEYWORD] and the incorrect answer options is a sub-concept that is not under [KEYWORD] in the OS ontology representation hierarchy. The `arr2D` is read to get information about relation. First, the index number for [KEYWORD] is read and this index number indicates a row index of `arr2D`. Then, the index for the column that has a relationship value of 1 is stored in `arrCA` (for the correct answers) and value 0 is stored in `arrIA` (for incorrect answers). Next is to run random function to select one concept from arrCA and two from `arrIA`. The following is the pseudocode to create answer options for S1.

```
for (i < arr2.getSize())
begin for loop
  if(arr2D[KEYWORD index][i] == 1)
    Adding concept arrCon[i] into arrCA;
  else
    Add concept arrCon[i] into arrIA;
end for loop

random function to choose 1 concept from arrCA;
random function to choose 2 concepts from arrIA;
```

For S2, the correct answer option is taken from the sub-concept other than [KEYWORD] and incorrect answer option is a sub-concept of [KEYWORD] in the OS ontology representation hierarchy. The information is also read from `arr2D`. The index number for [KEYWORD] is also read and indicated the row of `arr2D`. The difference for this question stem is the information taken for answer options. In contrast to S1, the index for the column that has a relationship value 0 is stored in `arrCA` for correct answer option, while value 1 is stored in `arrIA` for incorrect answers. The following is the pseudocode to create answer options S2.

```
for (i < arr2.getSize())
begin for loop
  if(arr2D[KEYWORD index][i] == 0)
    Adding concept arrConc[i] into arrCA;
  else
    Add concept arrConc[i] into arrIA;
end for loop
random function to choose 1 concept from arrCA;
random function to choose 2 concepts from arrIA;
```

E. Shuffling the order of question and answer options

Generated answer options for each question is stored in an array called *arrayOption*. This module implements the shuffling algorithm using the Collection class shuffle() from JAVA library. The purpose of this module is to allow the difference order of correct and incorrect answers for each question. The order of answer options can be one of the followings:

Order #1:
- A. Correct answer
- B. Incorrect answer
- C. Incorrect answer

Order #2:
- A. Incorrect answer
- B. Correct answer
- C. Incorrect answer

Order #3:
- A. Incorrect answer
- B. Incorrect answer
- C. Correct answer

Next is to rearrange the order of generated questions which is also using the Collection class shuffle() from JAVA library.

F. Print questions

This module is used to display questions based on the number of questions requested by the user. The random function is used to determine which concept is used to extract

the question generation. A random number generated indicates the index number of concepts stored in row exist of `array2D`. The concepts which the index number matched with the random number is used as a [KEYWORD] in question stems. Next, the answer option module is called to generate answer options based on the concepts stored in the matched index position. Then, the shuffled module in E is called to rearrange the order of question and answer options for each question.

5.4 Interface Design

A simple interface has been created to allow the user to enter the number of questions to be generated. This is to avoid over generation of questions from the system. The snapshot of the MCQ-G main page is shown in Figure 5.

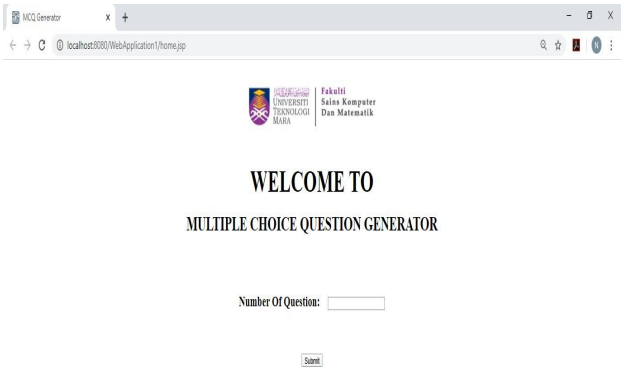

Figure 5: MCQ-G main page

The simple interface allows a user to enter the number of questions he/she plans to generate. Upon entering the submit button, the questions will be generated. Figure 6 and Figure 7 show the example questions generated for S1 and S2, respectively.

```
Which of the following is the concept of system-software.?
A. security
B. virtual-memory
C. loader,

Which of the following is the concept of process-and-thread-management.?
A. loader
B. mutual-exclusion-requirements
C. context-switching,

Which of the following is the concept of secondary-storage-structure.?
A. disk-scheduling
B. deadlock
C. protection.,

Which of the following is the concept of basic-architechture.?
A. interaction-between-processes-and-o-s
B. fundamentals-of-operating-system.
C. kernel,

Which of the following is the concept of security.?
A. the-security-problem
B. multitasking
C. loader,
```

Figure 6: Sample questions generated for S1

```
Which of the following is NOT the concept of system-software.?
A. compiler
B. "i/o-manager"
C. memory-management

Which of the following is NOT the concept of basic-architechture.?
A. goals-of-protection
B. "unix/linux-system"
C. kernel

Which of the following is NOT the concept of file-systems.?
A. file-support
B. access-methods
C. overlays

Which of the following is NOT the concept of virtual-memory.?
A. overlays
B. authentication
C. demand-paging

Which of the following is NOT the concept of concurrency-control.?
A. concurrency-and-race-conditions
B. mutual-exclusion-requirements
C. operating-system
```

Figure 7: Sample question generated for S2

6. FUNCTIONAL AND VALIDATION TEST RESULTS

There are two types of testing conducted for MCQ-G which are functional and validation test.

The functional test conducted indicated that all the buttons and text field worked perfectly. As for the text field, it could enter the number that the user wants where if the submit button was clicked, it would bring to the next page displaying the list of questions. When the user clicks on the back button, it will bring to the page where the user can insert the number of questions. The MCQ-G could also display the number of questions as requested. Table 2 shows the functions tested and its status.

Table 2: Functional test status

Function	Status
Text field to input the number of questions (input number only)	Pass
Submit Button	Pass
Save Button	Pass
Back Button	Pass
Generate several questions based on a request entered by the user	Pass
Random concept selection for [KEYWORD]	Pass
Questions displayed on the screen	Pass
Questions have different order each time submit button is click	Pass
Answer options change order	Pass

Validation test was conducted to check whether the selection of correct and incorrect answer options was accurate. The accuracy of the answer option was validated for both question stems S1 and S2. For S1, the correct answer option must be from the sub-concept of [KEYWORD] in its S1 and the two answer options must not be the sub-concept from [KEYWORD] of its S1. While for S2, the correct answer option must not from the sub-concept of [KEYWORD] in its

S2 and the two incorrect answer options are the sub-concept from [KEYWORD] of its S2.

The validation test was conducted manually by comparing the generated answer options with the ontology element in the ontology file. There were 20 test-runs conducted for each question stem, with each run generated 10 questions. The validation test conducted on all 20 test-run showed that all the generated answer options were generated accurately.

Table 3 shows the sample result obtained from the 1st run of the validation test for question stem S1. The test checked that the correct answer is the sub-concept of [KEYWORD] and the result status is recorded in "CorAns". The two incorrect answer options must be the sub-concept of the concept other than [KEYWORD] and the status is recorded in "IncAns".

Table 3: Sample result of validation test for S1

Questions generated	CorAns	IncAns
[KEYWORD]: **System Software** ***Correct answer:*** **C. Loader** ***Incorrect answers:*** **A. Security** **B. Virtual memory**	Yes	Yes
[KEYWORD]: **Secondary storage structure** ***Correct answer:*** **A. Disk scheduling** ***Incorrect answers:*** **B. Deadlock** **C. Protection**	Yes	Yes
[KEYWORD]: **Concurrency control** ***Correct answer:*** **A. Deadlock** ***Incorrect answers:*** **B. Buffering** **C. Swapping**	Yes	Yes
[KEYWORD]: **Virtual memory** ***Correct answer:*** **C. Overlay** ***Incorrect answers:*** **A. System software** **B. Assembler**	Yes	Yes
[KEYWORD]: **Clock** ***Correct answer:*** **C. Clock hardware** ***Incorrect answers:*** **A. Domain Protection** **B. I/O system**	Yes	Yes
[KEYWORD]: **Memory Management** ***Correct answer:*** **B. Virtual memory** ***Incorrect answers:*** **A. File system** **C. Mutual Exclusion**	Yes	Yes

Table 4 shows the sample result obtained from the 1st run of the validation test for question stem S2. The test checked that the correct answer is the sub-concept of the concept other than [KEYWORD] and the result status is recorded in "CorAns". The two incorrect answer options must be the sub-concept of [KEYWORD] and the status is recorded in "IncAns". The results of the experiments show that all questions have appropriate correct and incorrect answer options attached.

Table 4: Sample result of validation test for S2

Questions generated	CorAns	IncAns
[KEYWORD]: **File System** ***Correct answer:*** **C. Overlays** ***Incorrect answers:*** **A. File support** **B. Access method**	Yes	Yes
[KEYWORD]: **Virtual memory** ***Correct answer:*** **B. Authentication** ***Incorrect answers:*** **A. Overlays** **C. Demand paging**	Yes	Yes
[KEYWORD]: **Concurrency control** ***Correct answer:*** **C. Operating System** ***Incorrect answers:*** **A. Concurrency and race condition** **B. Mutual Exclusion**	Yes	Yes
[KEYWORD]: **I/O system** ***Correct answer:*** **B. System threats** ***Incorrect answers:*** **A. Principles of I/O hardware** **C. Principle of I/O software**	Yes	Yes
[KEYWORD]: **Memory management** ***Correct answer:*** **C. File systems** ***Incorrect answers:*** **A. Paging** **B. Virtual memory**	Yes	Yes

7. DISCUSSION AND FUTURE WORK

The algorithm design for generating tool that could automatically generate MCQ from the ontology is discussed. There are 6 modules proposed that are responsible for different functions. The main contribution of this paper is the development of a tool that could automatically generate answer options for MCQ using the relationship information in the ontology. The results for functionality and validation test showed that the MCQ-G was able to generate MCQ from the ontology knowledge representation. In addition, the relationship information of ontology representation could be used to select correct and incorrect answer options for MCQ. However, the result found in this paper only reported on the functionality and the accuracy of the algorithm design. Whether or not the generated questions had a meaningful or useful question, this will be explored in future studies. The MCQ-G will also be tested using different ontologies for other courses. Furthermore, the module for marking user's answers will be added.

8. CONCLUSION

The MCQ-G developed had shown a promising result for automatically generating MCQ with its appropriate answer options. This system had successfully deployed the ontology concept and relation to create MCQ with the related concept for answer options. The main contribution of this work is the design rule-based reasoning algorithm to generate MCQ automatically from ontology elements where the strategy for assigning answer options was adapted previous work. The limitation of the MCQ-G is it is only able to generate questions randomly from any concept and in the future, this function will be expanded to generate questions based on chapters in the Operating system subject.

ACKNOWLEDGEMENT

This work is supported by Universiti Teknologi MARA under the TEJA 2020 Internal Grant (GDT2020-41).

REFERENCES

1. S. Chen, B. Mulgrew, and P. M. Grant. **A clustering technique for digital communications channel equalization using radial basis function network**s, *IEEE Trans. on Neural Networks*, Vol. 4, pp. 570-578, July 1993.
2. A. C. Graesser and N. K. Person. **Question asking during tutoring**, *American Educational Research journal*, vol. 31, no. 1, pp. 104-137, 1994.
3. P. Gadge, V. Ravikant and G. Divya, **Advanced question paper generator using fuzzy logic,** *International Research Journal of Engineering and Technology*, vol. 4, no. 03, 2017.
4. F. K. Angar, H. G. Gori and A. Dalvi. **Automatic question paper generator system**, *International Journal of Computer Applications*, vol. 66, no. 10, pp. 42-47, 2017. https://doi.org/10.5120/ijca2017914138
5. X. Yao, G. Bouma and Y. Zhang. **Semantics-based question generation and implementation**, *Dialogue & Discourse*, vol. 3, no. 2, pp. 11-42, 2012.
6. M. A. Musen. **The protégé project: a look back and a look forward,** *AI matters*, vol. 1, no. 4, pp. 4-12, 2015.
7. M. G. Galety, S. B. Balaji and M. S. Basha. **OSSR-P: Ontological Service Searching and Ranking System for PaaS Services**, *International Journal of Advanced Trends in Computer Science and Engineering(IJATCSE)*, vol. 8, no. 2, pp. 271-276, 2019. https://doi.org/10.30534/ijatcse/2019/28822019
8. B. Saravana, R.S Rajkumar and B.F Ibrahim. **Service Profile-based Ontological System for Selection and Ranking of Business Process Web Services**, *International Journal of Advanced Trends in Computer Science and Engineering(IJATCSE)*, vol. 8, no. 1, pp. 18-22, 2019. https://doi.org/10.30534/ijatcse/2019/04812019
9. V. N. Dolzhenkov, I. D. Maltzagov, A. I. Makarova, N. S. Kamarova and P. V. Kukhtin. **Software Tools for Ontology Development**, *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, vol. 9, no. 2, pp. 935 -941, 2020. https://doi.org/10.30534/ijatcse/2020/05922020
10. Y. Ma, S. Gnaneswaran, M. Hardas and J. I. Khan. **Ontology of Operating System**, Networking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, Kent, UK, 2006.
11. A. Silberschatz, P. B. Galvin and G. Gagne. **Applied Operating System Concepts, Windows XP Update**, New Jersey, USA: Wiley, 2002.
12. A. Silberschatz, P. B. Galvin and G. Gagne. **Operating System Concepts**, Reading, Massachusetts, John Willy & Son., 2004.
13. A. S. Tanembaum. **Modern operating systems**, Upper Saddle River, New Jersey: Prentice-Hall, Inc, 2001.
14. K. Viljanen, J. Tuominen and E. Hyvönen. **Ontology libraries for production use: The Finnish ontology library service ONKI**, in *European Semantic Web Conference.*, Berlin, Heidelberg, 2019.
15. A. Papasolourus, K. Kanaris and K. Kotis. **Automatic generation of multiple-choice questions from Domain ontologies**, *In e-Learning*, pp. 427-434. 2008.
16. M. Cubric and M. Tosic. **Towards automatic generation of e-assessment using semantic web technologies,** *International Journal of e-Assessment. 2011*.
17. M. Al-Yahya. **OntoQue: a question generation engine for educational Assessment based on domain**

ontologies, *In 2011 IEEE 11th International Conference on Advanced Learning Technologies. 2011*.

18. T. Alsubait, B. Parsia and U. Sattler. **Generating Multiple Choice Questions From Ontologies: How Far Can We Go?,** *In International Conference on Knowledge Engineering and Knowledge Management. 2014.*