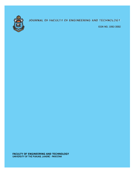
JFET 19 (2012) 109-124

Journal of Faculty of Engineering & Technology

journal homepage: www.pu.edu.pk/journals/index.php/jfet/index



MODELING AN AGENT FOR PAPER GENERATION SYSTEM USING UTILITY BASED APPROACH

M. Junaid Arshad[*], Mamoona Naz[1], Yasir Saleem[1], Amjad Farooq[1], K. H. Asif[1]
[1]Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan

Abstract

Test paper generation for examination in various level of education is one of the most preliminary requirements. Preparing and processing a test paper has vital place at every stage of E-education system. Most subsisting systems automate test paper generation by randomly selecting question items from knowledgebase/database. Usually these systems have no concern about difficulty level of question items. Some intelligent systems have been developed. But mostly it considers difficulty level at the time of assessment. Some other intelligent system gives approximate solutions for producing a test paper while considering difficulty level and type etc. Unlike the existing test paper generators, our utility based test paper agent chooses question items in such a way that the difficulty level of each question item takes part in computing exact difficulty level of test paper. Proposed Utility based test paper agent (UBTP Agent) picks the question items with its utility value. Hence provides exact difficulty level for whole paper as required by user. So test papers on the same level are different on basis of culled difficulty level. In design phase examiner creates a knowledgebase of questions for UBTP Agent by assigning some utility values with every question on which our selection algorithm operates. Whenever a test paper is required examiner provides difficulty level for test paper. And on the behalf of examiner UBTP Agent selects question with utility value in such a way the total of utility value is equal to required difficulty level. Later, a test paper is produced by test paper generator according to difficulty level specified by examiner. As percentage for any test paper is 1 to 100 percent for any test paper, therefore, 100 difficulty levels are available for any test paper. Here, selected difficulty level is for the entire paper. Finally, UBTP Agent model is proposed, implemented by providing algorithm, executed using case study, and tested to ensure the feasibility of this approach.

Keywords: Automation; Test Paper; Utility Based Agent; Knowledgebase

* Corresponding Author E-mail: junaidarshad@uet.edu.pk
Phone no.: 0092-42-99029260

1. Introduction

With the advent of computer based technology there is evolutionary change in many areas of our professional environments. Most peculiarly e-education is highly influenced. There is movement from manual to automated systems for different aspects of education system. Many information systems are being developed for many business and non business environment. These information systems provide great ease for professionals. The processes of education sector require evolutionary changes as this is major and very vast field. So there is great need from manual system education to e-education system.

These automated systems provide cost saving and time efficient solutions [1]. At every stage/level of education some examination process is required. The key to this examination process is a test paper. Testing a student at every stage i.e. from admission to examination is most inevitable part. Hence a test paper is always required. For example, General Aptitude Tests (GAT) is taken for admission in educational institutes at various levels of higher education. The patterns for test paper can be multiple choice questions, derivational, short or long questions etc. Furthermore its pattern can be related to subject for whom test paper generation is required. The questions are of logical, mathematical, and English comprehensive type. General admission or exam paper all are test paper with the difference in content. There is always a need for test paper in any area of education system.

Earlier an examiner is responsible for collecting question items, setting patterns for test paper, picking question items for paper, delivering papers, assessment of papers and delivering results. The patterns for these tests are usually same at different levels except difficulty level changes with the change in difficulty level. The most e-systems for generating test papers developed so far composes test papers using question items already entered in database/knowledgebase. With the arrival of e-education some software were developed to generate the test paper by randomly picking a question items from question items repository. It does not consider difficulty level of test paper. The idea of difficulty level is induced in some intelligent systems but it is considered at assessment time only [2]. Some multi-agent test paper generator considers difficulty level of question items in test paper generation but it does provide approximate solutions [1], [9]. Moreover, difference in test papers due to difference in difficulty level is not considered at same level. Subsequently test papers are not different on the basis of difficulty level. Some examples of simple and intelligent test paper generating applications are discussed under the heading of technical background. The proposed UBTP Agent provides absolute calculation of difficulty level for test paper.

Utility based agents are used when we are interested in how efficiently our goal is achieved. As discussed, our goal is to generate test with associated difficulty. Utility theory is used in assigning utility value to question items at the time of entry of question items and picking question items at the time of test paper generation. So this UBTP Agent is utility based test paper generator. The goal of UBTP Agent is generation of test paper with required difficulty level. The remaining material is organized as follows: Section 2 narrates technical background which includes utility based agents with description and names of some utility based agents and test paper generators and approaches. Section 3 proposes model for utility based test paper agent in terms of its structure and working. Section 4 describes implementation and testing in relation to our UBTP Agent. Section 5 demonstrates proposed approach using a case study. Section 6 compares different test paper generators including UBTP Agent. Finally, conclusions and future work are discussed in section 7.

2. Technical Background

2.1 Utility Based Agents

Utility based agents are autonomous, proactive and reactive in behavior in attaining its goal along with required efficiency [7]. The introduction of utility function in goal based agents helps in measuring the efficiency of way of attaining its goals [8]. More specifically utility based agents provide extra benefit of measure of efficiency in addition to usual benefits of goal based agents like usability and flexibility in terms of cost and time saving. The efficiency criteria can be any parameter i.e. minimization of time in covering distance by any agent. Utility based agents provide means to attain goals with efficiency measures. In some applications this efficiency measure is computation of total expected utility of goal.

Core interface agent architecture (CIaA) deals with the problem of usability in interfaces [3]. The usability is computed using the expected utility. Both environment and user intentions is unpredictable. This behavior is according to environmental events and intentions both. CIaA presents a new dynamic model to calculate expected utility for user intentions. The CIaA uses Bayesian network [5] based user model as its main component. Bayesian network for this dynamic interface architecture utilizes three random variables i.e. goals, actions and preconditions. Some utility functions are associated with every action variable and are used in computing total expected utility. Whenever, any observation is received from interface it is entered in observable history stack for later use. The CIaA agent can act in two ways, offering assistance and autonomously performing actions. The autonomous action and assistance against requested message by CIaA is furnished on the basis of comparison of expected utility and two limits in which first limit is initiated by user and second by use of help provided by CIaA.. This shows efficiency of CIaA in helping and fulfilling user intentions.

The agent for dynamic environments where plans for achieving goals are not predictable i.e. mobile agent tracking targets in dynamic environment deals in selection of plan with highest probability of attaining goals along with consideration of efficacy of executing this plan [4]. Mobile agent tracking targets requires highest possible plan for attaining goals along with consideration of efficiency of plan used. The efficient plan is that which has highest expected utility. Instead of computing actual, expected utility bounds for expected utility are considered. Bounds are recalculated iteratively for every state. A utility based optimal task scheduling problem (UOTSP) is proffered for a multi agent system in real time environment [20]. Tasks enter in the system having different importance and urgency and want to be accomplished by agents in emergencies. Some time there is a dependency relationship among tasks, so they need to be completed by agent in first. In fact, in emergency there is difference between tasks in terms of urgency and importance. Utility based optimal task scheduling problem overcomes the problems described above by defining the problem as by assigning utility values (which is any number) with completing each task and also with roles an agent can play. In other words agent gets some benefit by executing tasks i.e. utility value. In this approach an agent can play many roles and a role can be played by many agents. Moreover many tasks can be assigned to agents and vice versa. Now UOTSP gives efficient solution by maximizing the utility of overall system. The overall utility includes utility of tasks and agents both. UOTSP approach provides approximate solutions. Therefore, approximate algorithms like Greedy Task Scheduling (GTS) and restricted task ordering method (RTOM) are used to solve the problem. Many other utility based agents like Utility Based Multi-Agent System for Performing Repeated Navigation Tasks, Utility-based Agents for Hospital Scheduling [14], Utility of Mobile Agent Based E-Commerce Applications with Trust Enhanced Security [15], Wireless sensor networks and an energy-aware and utility-based BDI agent approach [16] are adapted.

2.2 Test Paper Generators and Approaches

Generating test paper is challenges, tedious and time consuming for the examiners. Usually the examiners keep their own question pool for test paper in any format to assist them to in preparing future exams. Existing technologies help the examiners to warehouse the questions in computer databases. The issue arises is how the existing technologies also helps the examiners to automatically generate the different types of test paper from time to time without considering about repetition and duplication from the growing question pool. For automation of generation for test papers, many applications are developed. These applications includes from simpler ones to base on artificial intelligence. Some of these simple test paper generators are Paper Builder and QGenie [17], [19]. The paper builder is the software for setting and generating examination question paper. The software has key features of complete and automatic paper generation, saving paper for future uses, very large question bank, making paper without repeating with previous paper's questions, printing answer keys.

Q Genie is a web based question paper generator solution for teachers, tutors, schools and coaching institutes for classes and subjects. The software provides facility of generating a test paper based on parameters like learning objectives, types of questions and competency level. In Q Genie, more consideration has been imparted to tag each question with its learning outcomes. There is an option to ignore the previous year questions into the current question paper.

The Intelligent system like IOAS is obtainable to automate the features like difficulty level, evaluation of students, and improve in learning by considering results [2]. The difficulty level is assigned at the time of assembling question pool but is used in assessment only. In this system, the preliminary level of students is assessed using some initial test. For initiation of software for particular groups of students some initial test is consorted. Results produced provides basis for next paper generation. The results for every student are refreshed according to new test results. The IOAS is intelligent online system comprising of three components that are Difficulty Assessment Algorithm, Automatic Question Generator, and Intelligent Question Module. The main component is intelligent question module. The question module makes use of difficulty assessment algorithm and automatic question generator.

In automatic question generator an open queue is used to represent test paper. Each node in queue is question item. The difficulty level in assessment is represented by considering number of edges traversed downward and upward in functional graph [6]. The type and the level of difficulty are taken in account of overall assessment of student's performance in particular test. A knowledge map represents student current level of understanding. Knowledge map is dynamic and is refreshed after each assessment. A multi-agent approach MASTG is used to deal with issues related to test paper generation in distributed environment [1]. The automatic test paper generation using multi-agent approach takes into account the difficulty level of questions and the type. Matrices are used to represent type and difficulty level of question items. In one matrix the percentage of each type of items in question paper is represented and in second matrix the difficulty level of each question item of particular type. MASTG calculates approximation of required difficulty level and then produces test paper. The multi-agent methodology is more like peer to peer approach in distributed environment.

There is also a unique solution namely GAMASTG which combines genetic algorithm for test paper generation and multi-agent systems for applications in distributed environment [9]. Genetic algorithm is based on survival of the fittest theory and has operations like cross over and mutation. The structure for test paper is defined using a chromosome having fixed length. A test paper is similar to single individual from population. It has some characteristics like type of questions, number of questions belonging to one type and the difficulty level for whole question paper. An integral weight for each type of question item is also ascribed as score for that type. A test paper is organized in portions of different type of question items. The score assigned to each type is used in selection and to compute the percentage of portion given. The difficulty level is divided in five portions of difficulty for whole test paper. The percentage of each portion varies for each test paper. This percentage is calculated using the score ascribed to each type. Additional features like knowledge points and teacher requirements are also considered. A teacher agent is responsible for providing all parameters. Test paper agent (TP Agent) is test paper generated. In start exam center agent (Exam Center Agent) initiates control agent (Ctrl Agent). A request for TP Agent is forwarded to Exam Center Agent which is real initiator for all agents like Ctrl Agent and TP Agent. Ctrl Agent get information from TP Agent, calculates fitness using fitness function, applies both genetic operators, and again calculates the fitness iteratively for many TP Agents. If TP Agent is not suitable then Ctrl Agent kills particular TP Agent. When the whole operation is over, the best solution is sent back. Same process is repeated to provide best solution. All non fit solutions are removed from population. This technique is based on approximation of fitness value and probabilities of survival of solution i.e. test paper. The multi-agent paradigm abates issues in conventional client server approach.

3. Proposed Model for Utility Based Test Paper Agent

Utility based agents augment the features of goal based agents by considering how much efficiency is attained when our goal is acquired. It is like in shortest path problem in which we want to find shortest path between source and destination in terms of less space. The main goal of utility based test paper agent (UBTP Agent) is to generate a test paper after selection of question items from knowledge base. But as it is utility based agent overall difficulty level of test paper is also considered. UBTP Agent considers the difficulty level of a test paper in selecting question items from knowledge base. Here the difficulty level of a test paper is its efficiency measure. Every question item in knowledge base has one extra attribute of utility value that is used in computing overall difficulty level.

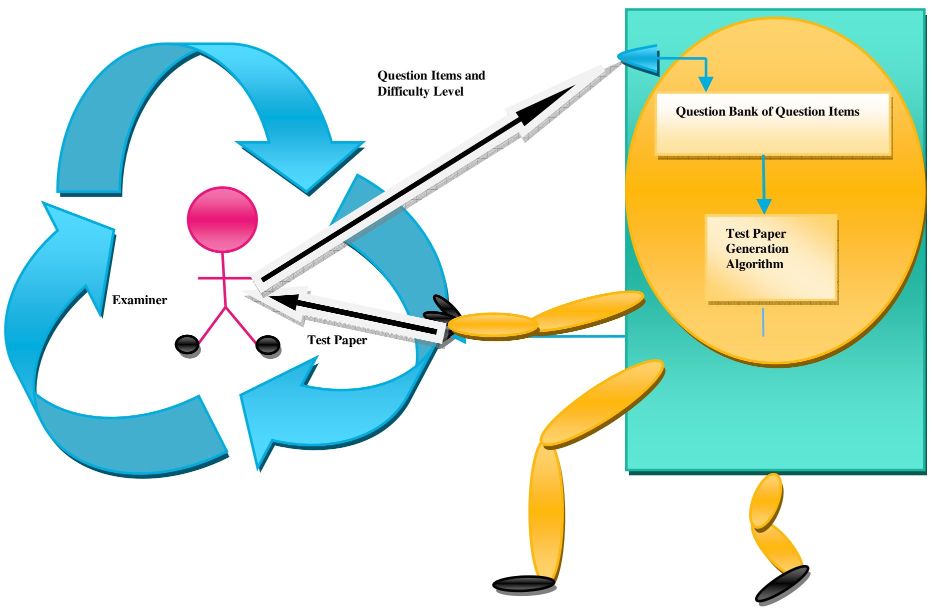

Question Items and Difficulty Level
Question Bank of Question Items
Test Paper Generation Algorithm
Examiner
Test Paper

Figure 1: Proposed Model for Utility Based Agent Test Paper Generator

UBTP Agent is intended to toil in e-education system especially for e-examination process. More precisely it toils for test paper generation component of e-examination process. It is initiated by examiners who previously comport test paper generation activities manually.

UBTP Agent is proposed to facilitate the examiner in test paper generation. Test paper generation is done on basis of considering certain parameters like difficulty level, subject and type.

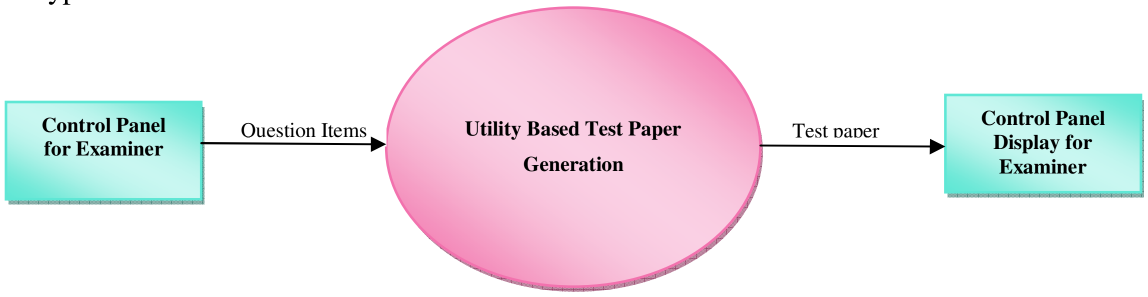

Control Panel for Examiner
Question Items
Utility Based Test Paper Generation
Test paper
Control Panel Display for Examiner

Figure 2: Context Level DFD for Utility Based Test Paper Generation

3.1 Structure

So by inferring structure of UBTP Agent the two main components of it are:

1) Knowledgebase Developer
2) Test Paper Generator

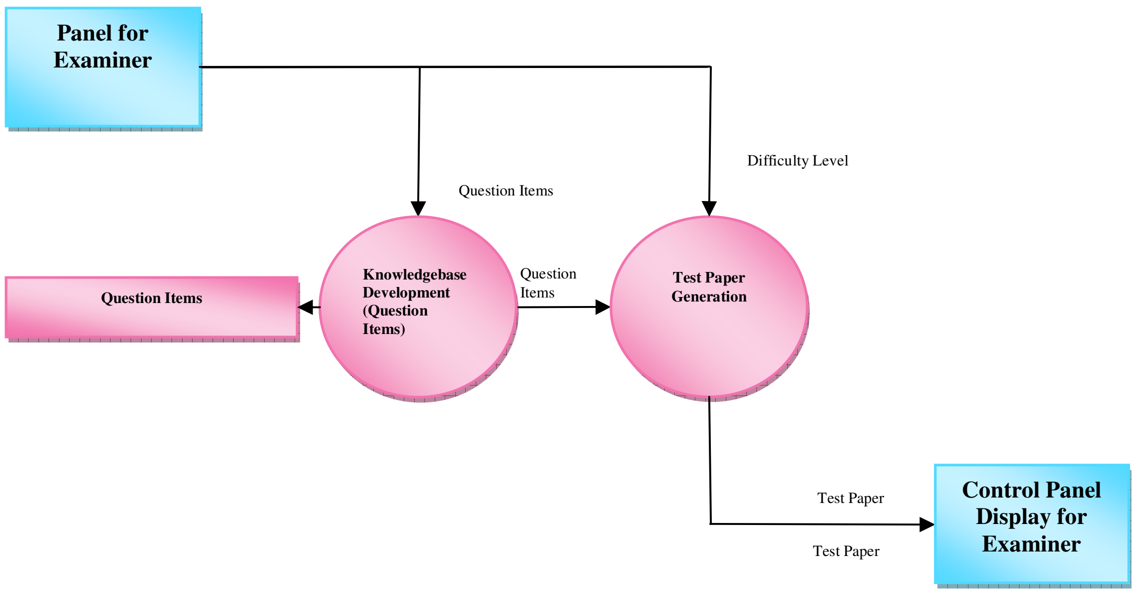


Figure 3: Level 1 DFD for Utility Based Test Paper Generation

3.1.1 Knowledgebase Developer

The preliminary requirement, in other words, very first input data for UBTP Agent is creation of knowledgebase. The knowledgebase for UBTP Agent is saved repository of question items. As a paper can be one to hundred percent difficult. Therefore 100 difficulty levels are associated with a single test paper. Each question item in knowledgebase is ascribed an integer value from 0 to 5 at the time of entry. A number in this range represents particular utility value assigned to question item. Higher the utility value of question item, the more it is difficult. For every utility value in this range some number of question items should be in knowledgebase. So that UBTP Agent can apply test paper generator algorithm.

The minimum total number of question items with every utility value in knowledge base or applying test paper generation algorithm is determined by using total difficulty levels and highest utility value that can be ascribed to a question item. For utility value what number of question items should be in knowledge base is calculated using by following formula:

$$\begin{aligned}&Minimum\ total\ number\ of\ question\ items\ with\ every\ utility\ value\\&\quad = total\ difficulty\ level/highest\ utility\ value\end{aligned}$$

Here total difficulty level is 100 and highest utility value is 5. So for every utility value number of question items is 20. After 20 the utility value of question items is decreased and until utility value will be 0. So 20 is minimum interval value after which utility value for question items can be changed.

Examiner enters the question items from most difficult to less difficult. The interval value provides ease in assigning utility value. When first slot of question items count is equal to interval value the utility value is automatically decreased and vice versa. Now from here minimum total number of question items in knowledge base can be calculated by:

$$\begin{aligned}&\mathit{Minimum\ total\ number\ of\ questions\ items}\\&\qquad = \mathit{Number\ of\ question\ items\ with\ every\ utility\ value} * (n+1)\end{aligned}$$

Where n is highest utility value that is 5 and 1 greater is because in range zero is involved. This formula is for lowest number of question items in knowledgebase. There is no upper limit on maximum number of question items. The number of question items a test paper can contain is provided by examiner. The lowest limit for a test paper to contain is the interval value which is 20. But there is no upper limit for any number of question items for test paper. When knowledge base is developed, UBTP Agent starts taking question items from knowledgebase to generate a test paper.

3.1.2 Test Paper Generator

For UBTP Agent a knowledgebase consisting of questions is created. All the questions with associated values are acting as states. These states are available for computing a path to reach our goal i.e. generation of test paper with required difficulty level. The test paper generation is based on culling the percentage of difficulty level ranging from one percent to 100 percent.

The percentage of anything varies from one to 100 percent. Therefore the range of difficulty level is one to 100 percent giving 100 difficulty levels for a single test paper. The selection of highest difficulty level yields difficult test paper and low difficulty level gives easy test paper. The examiner work is abated to only culling difficulty level for test paper. Now UBTP Agent is responsible for picking the questions from knowledge base. The required difficulty level for test paper is computed by culling appropriate questions with associated utility value from knowledge base. The highest utility value of any question for a test paper under consideration is calculated by formulae:

$$h\_uval = (d\_level / 20) \quad \text{if d\_level\%20==0}$$

$$h\_uval = (d\_level / 20) + 1 \quad \text{if d\_level\%20!=0}$$

Where 20 is the interval value for difficulty level after that the highest utility value is changed. The d_level is difficulty level for test paper given by examiner. The utility value of question items is used to compute the accumulative difficulty level of test paper. The algorithm for test paper generation is explained in implementation section.

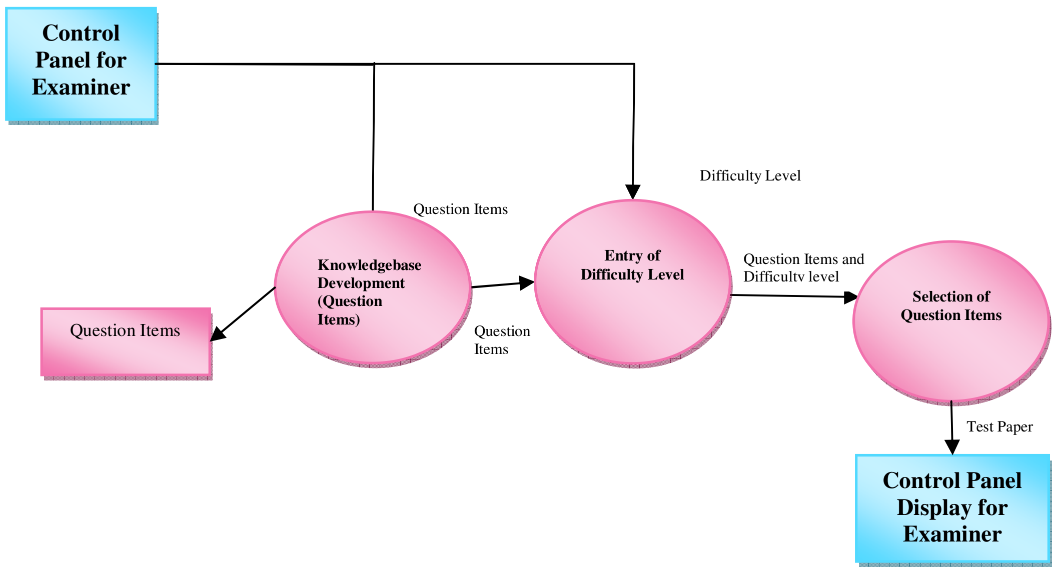

Control Panel for Examiner
Difficulty Level
Question Items
Knowledgebase Development (Question Items)
Entry of Difficulty Level
Question Items and Difficulty level
Selection of Question Items
Question Items
Question Items
Test Paper
Control Panel Display for Examiner

Figure 4: Level 2 DFD for Utility Based Test Paper Generation

The question items are selected in knowledge base randomly in order to ensure equal distribution of selection of every question in knowledge base. In order to avoid repeating question the selected question is marked unavailable after choosing. This prevents picking of any repeating question item. In this scenario UBTP Agent is not interested in any maximization or minimization. It gives a test paper with required difficulty level dynamically.

3.2 Working

The goal of our proposed utility based agent is to generate a test paper according to the level of difficulty chosen. The test paper is comprised of prescribed number of questions. These questions are selected with considering utility value and are used in computing path for reaching our goal i.e. generation of test paper.

For UBA to operate, the application primarily requires building a knowledgebase of question items. The examiner starts developing knowledgebase by entering number of question items appearing in the test paper. The number of question items actually entered is according to the methodology described for knowledgebase above. When this task is completed, we have knowledgebase on which our test paper generation algorithm for question items operates. The examiner provides the level of difficulty in a given range i.e., 1 to 100.

The selection of questions start and the test paper is generated dynamically. UBTP Agent selects given number of question from knowledge base. The selection takes in consideration of utility value of question items in order to meet difficulty level requirement entered by examiner. After selection the generated test paper is according to difficulty level specified by examiner.

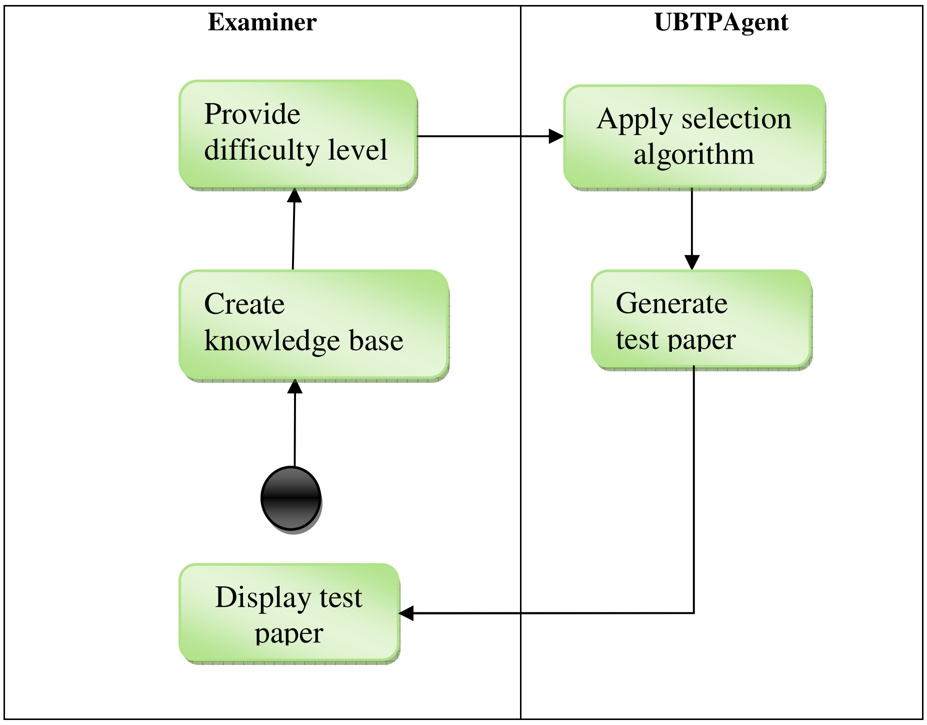


Figure 5: Agent UML Activity Diagram of UBTP Agent

4. Implementation of UBTP Agent

UBTP Agent is implemented in C++ and knowledge base is created in the form of file. Then This UBTP Agent is tested using system and integration test, load and stress test, acceptance test and usability test.

4.1 Implementation of UBTP Agent

UBTP Agent is designed for generating test paper with required difficulty level. So question items are needed. These question items can be of any type like long question, short question, derivational, multiple choice question etc. These question items are entered by examiner by simply using a loop. But with every question an extra attribute utility value is also stored. This utility value is assigned according to difficulty level of question. The higher the utility level, more difficult is question. These utility values have no concern in assessment of test paper. These utility values are used just in computing overall difficulty level of test paper. An algorithm can be written for creating knowledgebase for UBTPA gent by using number and type of question a test paper contain and ascribing utility values in the range 0 to 5. Examiner can use any programming paradigm for writing an algorithm for entering questions. So by using algorithm knowledgebase of UBTP Agent is created and saved for UBTP Agent. The examiner fills data for knowledgebase.

The Pseudo code is as follows:

```
1. READ noofquestions //number of questions
2. FOR j<6
3. FOR < noofquestions
4. SET Qitem // question items having u-value
5. ENDFOR
6. ENDFOR
```

Figure 6: Knowledgebase creation Pseudo code

An algorithm for creation of knowledgebase is given in Figure 6. After creation of knowledge base difficulty level for test paper is given UBTP Agent by examiner. The range for difficulty is from 1 to 100% giving 100 difficulty levels for a test paper, The UBTP Agent picks questions having utility value. Thus computing required difficulty level.

The Pseudo code is as follows:

```
1.  SET dlevel to 0,
    huval to 0,
    i to 0,
    j to 0,
    gcount to noofquestionss
2.  READ dlevel //difficult level of test paper
3.  CALCULATE
    if ((huval = dlevel%20)== 0)
    huval = dlevel/20
    else
    huval = dlevel/20 + 1
// highest uvalue for this paper by usng formula
4.  FOR j<20-(20-(dlevel%20))
5.  GET Qitem //question items having one highest u-value.
6.  ENDFOR
7.  FOR i<20-(dlevel%20)
8.  GET Qitem // question items having one less u-value
9.  ENDFOR
10. FOR j<qcount-20
11. GET Qitem //question items having 0 u-value.
12. ENDFOR
```

Figure 7: Test Paper Generator Pseudo code

An algorithm for generation of test paper containing question items is given in Figure 7. Three sets of question items are selected on the basis of utility value. One set of the question items has highest utility value calculated above. The second set of question items has one less than highest utility value calculated. The third set of question items comprised of questions with 0 utility values. A marked field associated with every question is set to 1 when it is selected. This ensures no repeating question item is selected. Every time a question item is selected, the associated utility value is used in computing overall difficulty for test paper. The final response of UBTP Agent is test paper with required difficulty level.

4.1.1 Development Tools for Implementing Utility Based Agents

Since utility based agents are goal based agents with efficiency measure. So, our main concern is on developing tools for achieving the task. Some existing tools that provide reusable components for developing agents are agent Tool Project, ZEUS, Comet Way JAK [18]. However, we adopt object oriented paradigm for implementing our UBTP Agent. This implementation is not specific to any development platform like java or .net. Moreover, the knowledgebase can be in form of file or relational database as we have used object for warehousing the information about objects i.e., question items in knowledge base. Here, we have used visual C++ for implementing our UBTP Agent and files for the permanent storage of knowledgebase.

5. Case Study

1). To initiate UBTP Agent needs a knowledgebase. Therefore first examiner develops knowledgebase. It is necessary to have some specified number of questions in knowledgebase.

2). In this case highest utility value is 5. So the 0 to 5 utility values can be ascribed to questions. For every utility value in this range some number of questions should be in knowledgebase. Here total difficulty level is 100 and highest utility value is 5. So for every utility value minimum number of question is 20.

3). In this case the minimum total number of questions is 120. (There is no upper limit on maximum number of questions.)

4). As an example, the user prompts for 120 or more questions to be entered according to formula given in knowledgebase development as shown in Figure 8.

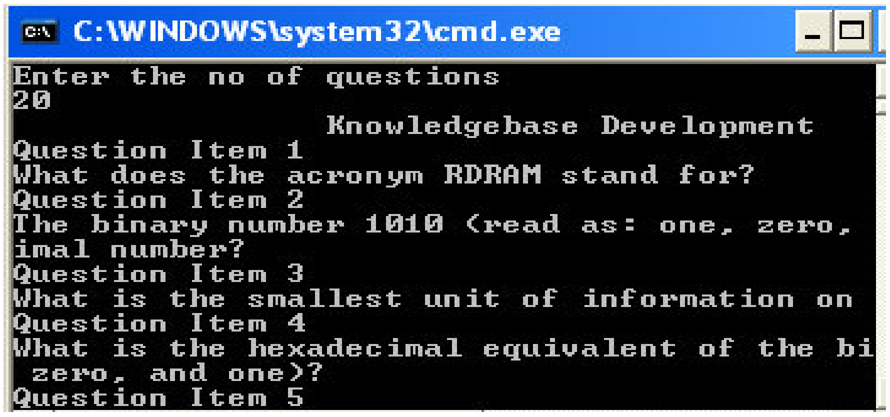
5). In this scenario the minimum number of questions for test paper can contain are 20. There is no upper limit for questions. Examiner provides number of question items and question items also.

6). Next the UBTP Agent prompts for the percentage of difficulty level for test paper. The examiner enters value a value 77 as shown in Figure 9.

7). In this case the highest utility value is 4. According to test generation algorithm the 17 questions in this test paper are value of 4 and 3 questions in this test paper are value of 3. The remaining questions are of utility value 0.

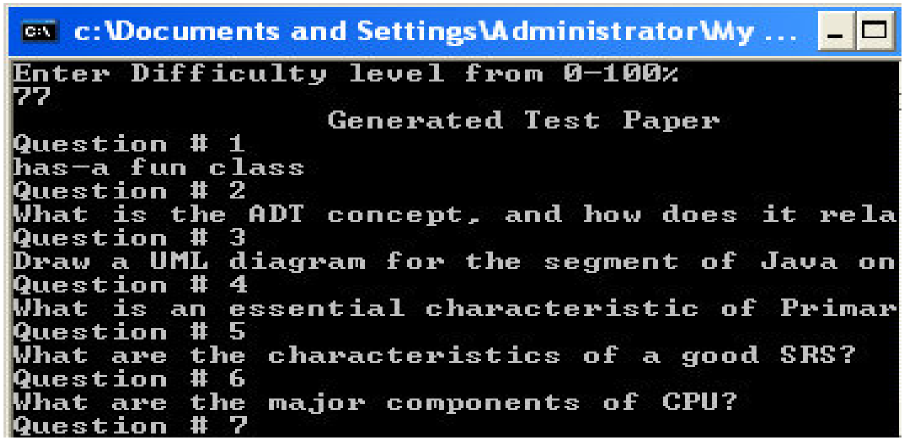
8). Computing these utility values for all questions confirm the overall difficulty level given for test paper.

9). Finally Questions are displayed on screen as shown in Figure 9.



```
Enter the no of questions
20
                Knowledgebase Development
Question Item 1
What does the acronym RDRAM stand for?
Question Item 2
The binary number 1010 (read as: one, zero,
imal number?
Question Item 3
What is the smallest unit of information on
Question Item 4
What is the hexadecimal equivalent of the bi
 zero, and one)?
Question Item 5
```

Figure 8: Knowledgebase Development



```
Enter Difficulty level from 0-100%
77
                Generated Test Paper
Question # 1
has-a fun class
Question # 2
What is the ADT concept, and how does it rela
Question # 3
Draw a UML diagram for the segment of Java on
Question # 4
What is an essential characteristic of Primar
Question # 5
What are the characteristics of a good SRS?
Question # 6
What are the major components of CPU?
Question # 7
```

Figure 9: Generation of Test Paper

6. Test Paper Generators' Comparison

The proposed UBTP Agent's main emphasis is on test paper generation. Before UBTP Agent many approaches are proffered to generate the test paper. Normally all test paper generator selects question from some stored question bank. The final output of these test generators is test paper containing questions. The number of questions in test paper is according to specified by user .All these specifications are also true for UBTP Agent. These test paper generation approaches including UBTP Agent are distinguishable on the basis of test paper generation algorithm. The test paper generation algorithms can be classified on parameters like type of question, subject score, difficulty level, solution type and development approach.

Solution / Parameter	Simple software based and Intelligence based	Agent based	Utility based agent Test Paper Generator
Type	Yes	Yes	Yes
Subject	Yes	Yes	Yes
Score	Yes	Yes	No
Difficulty Level	In some but not at the time of generation of paper, in assessment only	At the time of generation of paper	At the time of generation of paper
Solution type	-	Approximate	Absolute
Development Approach	Single with software solution, client server paradigm in distributed environment	Multi-Agent based in distributed environment	Single Agent

Table 1: Test Paper Generators' Comparison

7. Conclusion and Future Work

The UBTP Agent gives a novel approach for generating test paper using utility based agent. This agent provides us a great ease in following ways:

a) The development of knowledgebase consisting of question items either using file system or database system.

b) Generation of test paper using question items having utility value in knowledgebase according to difficulty level selected.

c) The final simulation results validated the feasibility of the proposed approach.

By using UBTP Agent examiner only has to provide difficulty level for whole test paper at time of test paper generation. Posed UBTP Agent considers only one part of e-admission system i.e. test paper generation. The examiner creates a knowledgebase according to methodology discussed in previous sections. UBTP Agent extracts the part of selecting question from examiner. UBTP Agent selects questions and generates test paper according to prescribed difficulty level and handover it to examiner. In this way it facilitates examiner who has now only work of entering questions and difficulty level.

Furthermore, this single agent system can be extended to multi-agent system by incorporating exam agents, distribution agents, and assessment agents. The exam agent is responsible for providing different types of question items. The distribution agent concerns with disseminating online test papers and submissions. The assessment agent can be built for mathematical or derivational type of question items.

References

[1] Wang Hairui, and Wang Hua, 2008. Research and Implementation of Multi-agent Based Test Paper Generation Algorithm. In International Conference on Computer Science and Software Engineering, IEEE Computer Society, Washington DC, USA, vol. 01, pp 493-496.

[2] Ang Tan Fong, Hu Heng Siew, Por Lip Yee, Liew Chee Sun, 2007. IOAS: An Intelligent Online Assessment System. WSEAE Transactions on Computers, issue 3, vol.6, pp 552-559.

[3] Brown, Scott M. and Santos, Eugene and Banks, Sheila B. 1998. Utility Theory-Based User Models for Intelligent Interface Agents. In 12th Biennial Conf. of the Canadian Society for Computational Studies of Intelligence.

[4] Jak Kirman, Ann Nicholson, Moises Lejter, Thomas Dean, and Eugene Santos Jr. 1993. Using goals to find plans with high expected utility. In Second European Workshop on Planning, Linkoping, Sweden, pp 158-170.

[5] Judea Pearl, 1996. Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference. Publisher Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[6] Tao Li and Sam Sambasivam, 2003. Automatically Generating Questions in Multiple Variables for Intelligent Tutoring. Proc ISECON, pp 472 – 478.

[7] Frederick Mills and Robert Stufflebeam, 2005. Introduction to Intelligent Agents. National Science Foundation.

[8] Ben Coppin. , 2004. Artificial intelligence illuminated, John and Bartlett Publishers, pp 544-551.

[9] Anbo Meng, Luqing Ye, Daniel Roy, Pierre Padilla., 2007. Genetic Algorithm Based Multi-agent System applied to Test Generation. Elsevier Science Ltd, pp 1205-1223.

[10] Chmiel, K. et al. (2004). Testing the efficiency of JADE agent platform. In Proceedings of the 3rd international symposium on parallel and distributed computing, Cork, Ireland (pp. 49–57). Los Alamitos, CA: IEEE Computer Society Press.

[11] FIPA00008, (2002). FIPA SL Content Language Specification. Foundation for intelligent physical agents. Available from http://www.fipa.org/specs/fipa00008/.

[12] FIPA00037, 2000. FIPA Communicative Act Library Specification. Foundation for intelligent physical agents. Available from http://www.fipa.org/specs/fipa00037/.

[13] Ram Meshulam, Ariel Felner, Sarit Kraus, 2005. Utility Based Multi-Agent System for Performing Repeated Navigation Tasks. In International Conference on Autonomous Agents. ACM publishers, New York, USA.

[14] Hans Czap, Marc Becker, Malte Poppensieker, Alexander Stotz, 2005. Utility-based Agents for Hospital Scheduling. In First International workshop on multi-agents systems for medicine, computational biology, and bioinformatics. The German National Research Foundation, Germany.

[15] Ching Lin, Vijay Varadharajan, Yan Wang, 2005. Utility of Mobile Agent Based E-Commerce Applications with Trust Enhanced Security. Trust, Privacy and Security in Digital Business, Heidelberg.

[16] Song Shen, Gregory M., P. O'Hare, 2007. Wireless sensor networks, an energy-aware and utility-based BDI agent approach. In International Journal of Sensor Networks. Inderscience Publishers, Switzerland.

[17] Fresh logics, 2009. Paper Builder. Foundation for Examination Question Paper generation. Available from http://www.freshlogics.com/paper_generator.php

[18] Agent land, 2001. Development Tools. Available from http://www.agentland.com/Download/8How_to_develop_an_agent/Development_tools/

[19] JILIT, 2008. QGenie. For manual test paper generation. Available from http://www.qgenie.com/whyqgenie.do

[20] Xiaowei Zhang, Bin Li, Junwu Zhu, Jun Wu, 2010. Utility Based Optimal Task Scheduling Problem in a Multi-agent System. In International Journal of Digital Content Technology and its Applications Volume 4, Number 9.