Automatic Recommendation of Software Design Patterns: Text Retrieval Approach

Abeer Hamdy[1, 2*] , Mohamed Elsayed[1]
[1]Faculty of Informatics and Computer Science, British University in Egypt, Elshorouk city, Egypt
[2]Computers and Systems Departments, Electronics Research Institute, Cairo, Egypt

[*]Corresponding Author: Abeer.hamdy@bue.edu.eg
Manuscript submitted February 2018; accepted April 23, 2018.
doi: 10.17706/jsw.13.4.260-268

Abstract: Design pattern is a reusable solution to a commonly occurring design problem in certain context. Using design patterns in software development improves the product's quality, understandability and productivity. However, it is a challenging task for novice developers to select the right design pattern to solve a design problem. The paper proposes a methodology for the automatic selection of the fit design pattern from a list of patterns. The proposed methodology is based on Text retrieval approach where the design problem scenarios are described in natural language. A vector space model (VSM) was created for the catalogue of design patterns. A vector of features consists of unigrams and bigrams is generated for the given design problem scenario. The recommended design pattern is the closest to the problem scenario. The proposed mechanism was evaluated using the Gang of four design patterns and the experimental results showed the effectiveness of the proposed methodology.

Keywords: Design pattern selection, Gang of four, Information retrieval, Recommendation, Vector space model.

1. Introduction

In software engineering, a software design pattern (DP) is a general well-proven reusable solution to a recurring problem within a given context in software design. Design patterns represent a standardized and well documented best practices used by experienced software developers. Using design patterns in software development results in increasing the software reusability, quality and maintainability, in addition to reducing the technical risk to the project by not having to develop and test a new design. Furthermore, design patterns consider a communication language that facilitates the communication among the development team members [1], [2].

However, The existing of a large number of design patterns makes the selection of a fit design pattern for a given design problem a difficult task to the experienced developer, and makes it a challenging task for the inexperienced one who is not familiar with design patterns. To overcome this difficulty a supporting tool that automatically suggest to the developer a right design pattern for a given design problem during the design phase becomes a necessity.

Recently, a number of research studies was conducted on the automatic selection of the fit design pattern. Some of these studies developed techniques for suggesting the suitable pattern based on the UML design diagrams [3], [4]. Other techniques are based on question-answer [5], [6]. Some studies used text

classification and text retrieval techniques [7]-[10]. Others recommended design patterns based on anti-patterns detected in the design documents or the code [11], [12]. Some studies used Case Based reasoning CBR technique. Where, the fit design pattern is selected according to the previous experiences of pattern usages stored in a knowledge base in the form of cases [13], [14].

This paper proposes an approach based on text retrieval for automating the process of software design pattern recommendation to solve a given design problem. The motivation for this approach is the following:

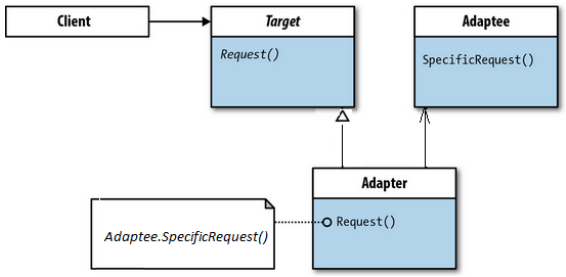
1. It allows the developers to describe their design problems in natural language.
2. The task of design pattern recommendation is analog to the text retrieval task.

The structure of the paper is as follows: Section 2 Explains design patterns. Section 3 presents the literature survey in the field of design pattern selection. Section 4 discusses the proposed approach and section 5 discusses the experiments and the results. Finally, section 6 concludes the paper.

2. Design Pattern Illustration

The concept of design pattern was initiated in software development in 1994 when four software engineers (Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides) published their book titled "Design patterns: Elements of reusable object oriented software" [1]. These authors are together popular with the title Gang of Four (GOF). GOF patterns are 23 patterns and categorized into three categories namely Creational, Structural and Behavioral patterns. GOF defined a template to describe the patterns. This template has two counterparts which are the pattern's problem domain and the solution domain. The problem domain counterpart includes the intent of the pattern and the context where the pattern can be applied. While the solution domain counterpart includes the UML diagrams that describes the static structure of the pattern and its dynamic behavior. In addition to, a description to the constituent components, their responsibilities and the ways in which they collaborate. The tradeoffs (consequences) when applying the pattern and the anti-patterns are also discussed in this counterpart. Table 1 shows part of the description of a structural pattern called Class Adapter design pattern.

Table1. Adapter design pattern description

Category	Structural
Problem Domain:	
Intent	Change the interface of a class into another interface. It let the classes work together without modifying their source code.
Applicability	The Class Adapter pattern is used when: - You want to reuse an existing class but its interface is not compatible with the interface you need. - You have a class hierarchy and you need to use one or more subclasses. But you need to change their interfaces. It is impractical to subclass the subclasses to change their interface. - You need to have classes with incompatible interfaces work together.
Solution Domain:	
Structure	
Participants	-Target: Defines the interface that the Client uses. -Adaptee: The existing class with the interface that needs to be adapted. ·Adapter: Changes the interface of Adaptee class to the Target class interface.
Collaborations	Client class invoke methods of an adapter object. In turn, Adapter invokes corresponding methods in the Adaptee class to execute the request.

3. Literature Review

The following subsections summarize the literature review in the field of design pattern selection.

3.1. UML Based Approach

Kim and Khwand [3], Kim and Shen [4] use class diagrams and collaboration diagrams to generate a meta-model for each design pattern. However, this approach has two limitations which are: 1) the meta-models of some patterns will be similar as some patterns are similar in their structure but they have different intent for example State, Strategy patterns and Façade, Adapter patterns. 2) This approach is not scalable due to the overhead resulting from generating the meta-model; in addition to the more the number of patterns increases the more the similarity between the meta-models increases.

3.2. Question-Answer Based Approach

In question-answer based approach the software developer is provided with some questions about the design problem. The most suitable patterns for this problem are recommended based on the designer answers [5], [6]. Palma et al. [5] constructed a Goal-Question-Metric model (GQM) from the question-answers to recommend patterns. In this GQM model, the defined goal is a pattern name. The system consists of two layers, the first layer has the conditions, where the second layer has the sub-conditions. The model evaluation has been done by a total of six graduate students along with two information technology professionals. The outcome of the evaluation resulted in a success ratio which reached 50%. While, Pavlie et al. [6] used the question-answers to build an ontology-based model for design patterns recommendations. However, constructing the questions in this approach is a challenge task especially with the large number of patterns. Furthermore, the set of questions are usually biased towards the specifics of the design patterns themselves rather than the software design problem.

3.3. Case Based Reasoning (CBR) Approach

In CBR approach the fit design pattern is recommended based on previous experiences (cases) stored in a repository. Each case comprises two main parts which are: A description to the problem and the solution (fit design pattern). Gomes et sl. [13] built a repository of cases and retrieve the closest case from the repository for a user provided class diagram. While, Muangon and Intakosum [14] proposed a solution where both Case Based Reasoning (CBR) and Formal Concept Analysis (FCA) are integrated together forming a cohesive technique. This integration enabled the organization of indices to construct a complete design problem description which is used as aid to find more suitable design patterns. Both of the indices and case similarity are calculated using FCA. As argued by Gomes et al. The core shortcoming of CBR based approach is the fact that its accuracy relies on both of the quality and diversity of the case repository.

3.4. Anti-Patterns Based Approach

Nahar [11] identifies the anti-patterns in the design diagrams then recommend the suitable design pattern. Smith and Smith and Plante [12] recommend patterns at the code-level, where patterns are recommended dynamically during the code development phase. They identify anti-patterns using structural and behavioral matching in the code, and then suitable design patterns are recommended to overcome the identified anti-patterns. However, design pattern recommendation in the code development phase is too late as the software has already been designed and should be changed.

3.5. Text Classification and Retrival Based Approach

This approach is based on matching the design problem textual description against DP textual descriptions [7]-[10]. Sanyawong *et al*. [7] developed classifiers to determine the design pattern category for a given design problem. They used popular classification techniques: Naive Bayes, J48, k-NN, and SVM.

They used 26 case studies for evaluation. Suresh [8] proposed a framework for design pattern recommendation that depends on two approaches which are: text retrieval, question-answer. In this framework the design problem is represented as a collection of words. Initially, the system operates on this collection of query words with the goal of finding a suitable design pattern whose intent is similar to the query words. In next step, the intent of top candidate design patterns are displayed for the designer to select the most suitable pattern intent. The system provides further support to the designer through providing a set of questions. Then the system scores the recommended patterns according to the designer answers. Ultimately, if no suitable candidate pattern is found, the system searches for the most similar query in its history database. Then the corresponding pattern is allocated to the problem. However they have partially implemented and tested their framework. In addition to using the pattern intent only in the first step will not produce a good performance and requires the involvement of the user in the selection process.

4. Methodology

This approach is based on building a Vector Space Model (VSM) for the GOF design patterns. VSM represents each design pattern as a vector in a vector space. Vectors that are close together in this space are similar and vectors that are far apart are distant. The design problem will be represented as a vector in the same space. The closest design pattern vector to the query vector is the recommended pattern for this problem. Fig. 1 illustrates the steps of the proposed approach which starts with the textual preprocessing then indexing and feature selection which is followed by applying a similarity measure function to select the closest Design Pattern.

4.1. Text Preprocessing

Each design pattern description and each design problem scenario is processed through three activities: Tokenization and Normalization then Stop-Word Removal then stemming. The goal of the preprocessing is to reduce the feature set size and data the sparsity.

- Tokenization and Normalization: The token is a sequence of characters and does not include delimiters such as punctuation marks and spaces. Tokenization is the process of splitting the text at the delimiters into tokens (words). Then all the words are normalized by transferring them into lowercase.
- Stop-Word Removal: It is the process of elimination of non-descriptive words like linking verbs and pronouns. Stop words are considered noise, they increase the size of the Vector Space Model and do not contribute to the retrieval process.
- Stemming: It is a process for normalizing words to their root forms. For example a stemmer can reduce each of the words “instantiating” and “instantiated” to the word “instantiate”. Also, verbs like “am”, “is” and “are” are transferred to the to the verb “be”. Porter stemming algorithm [15] was used in this work.

4.2. Indexing and Feature Selection

VSM is used to represent the collection design patterns as it is common and effective Statistical representation of a corpus of documents in many text based applications like text categorization [16] and text summarization [13]. VSM model represents each design pattern as a vector of features. In this work features are unigrams (words) and bigrams (two words that appear consecutively in the text). Bigram features are important to distinguish between the different patterns especially the patterns of the creational category. For example, all the Creational category patterns include the word “create” but the Singleton

pattern is the only pattern that has the bigrams “create one” and “one instance”; the Factory pattern has the bigram “create object”. Also, many patterns include the words “object” and/0r “class” ; while for example State pattern includes the bigram “object state” , Adapter pattern has the unique bigrams “change interface” and “incompatible interface”.

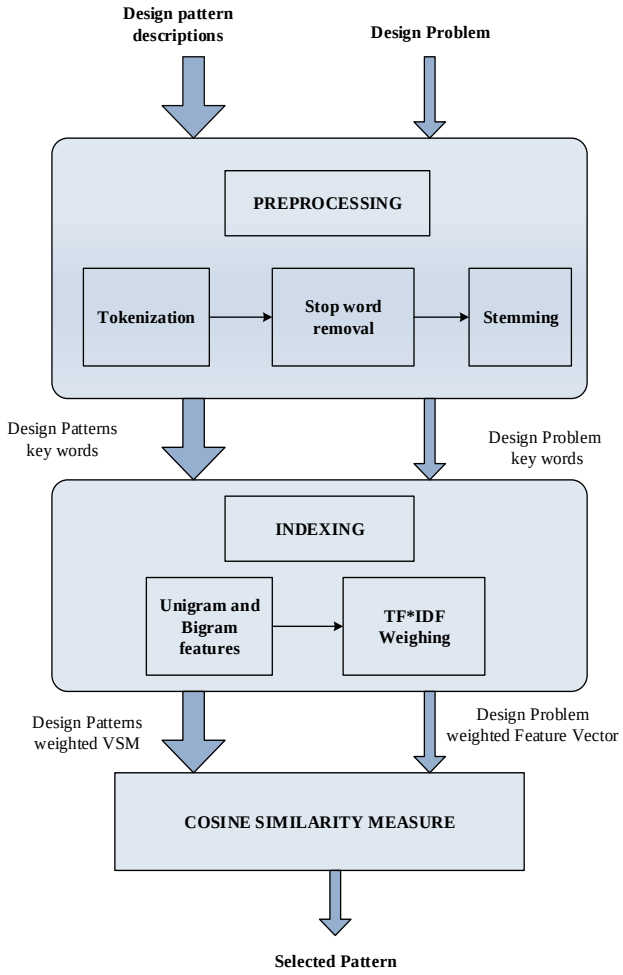


Fig. 1. Proposed framework for automatic selection of a design pattern.

In order to enhance the performance of the text retrieval process and clean the noises of the documents; term weighting scheme should be used [17], where the weight of a term in a VSM reflects the relative importance of this term for a specific DP description within a collection of DP descriptions. TF*IDF [17] weighing mechanism was used in this work as it is a popular weighing functions and used in many text mining applications. TF*IDF stands for Term Frequency- Inverse Document Frequency. Term frequency TF (t,d) measures how many times a term (t) occurs in a document (d). While Document frequency DF (t,D) measures how many documents in a collection (D) the term (t) appears in. Inverse document frequency IDF (t,D) equals to the inverse of DF(t,D). Classical TF*IDF is computed by equation (1) as follows:

$$TF * IDF(t,d,D) = TF(t,d) \times IDF(t,D) \quad (1)$$

where, $IDF(t,D) = \frac{1}{\mathrm{DF(t,D)}}$

TF*IDF value copes with the fact that the repetitive words in a document usually carry a high level of information to that document, and that the less frequent a term is mentioned in a corpus the higher its importance to the document in which it appears. It should be noted that TF*IDF computed using equation

(1) does not take the document length into consideration. Also, TF value indicates that a term occurs five times in a document is five times valuable than if it occurs once in the same document, which is not true. So, other forms to compute the TF*IDF were recommended in the literature to make the TF*IDF values correspond to user intuitions of the relevance of each term. In this work equation (2) is used to compute TF*IDF.

$$TF * IDF(t,d,D) = Sqrt\big(TF(t,d)\big) * IDF(t,D) * 1/Sqrt(length) \quad (2)$$

where, $IDF(t,D) = \log(N/(DF(t,D)+1))$

After applying the TF*IDF each design pattern description and each problem scenario will be represented by a vector of feature $V(d) = (v(t_1, d), v(t_2, d), \ldots\ldots, v(t_n, d))$, where, d is the document of a pattern description or a problem scenario, $v(t_i, d)$ is the TF*IDF of each term in d, n is the size of the VSM which is the number of the key unigrams and bigrams in the collection of the design pattern descriptions after the preprocessing stage. Each feature vector will have a number of terms with Zero $v(t_i, d)$.

4.3. Similarity Measure

A similarity measure is used to retrieve the suitable design pattern, for a given design problem, from a collection of patterns. In this work Cosine Similarity (CS) is adopted. CS is one of the popular measures in the field of information retrieval. As the value of CS between two vectors is determined by the angle between the vectors, while Euclidian distance is based on the lengths of the vectors. Cosine similarity $CS_k(V(S), V(P_k))$ between the feature vector of a design problem ($V(S)$) and the feature vector of the Kth design pattern description ($V(P_k)$) is given by Equations 3 and 4 as follows:

$$CS_k(V(S), V(P_k)) = \frac{V(S)\cdot V(P_k)}{\|V(S)\|\|V(P_k)\|} \quad (3)$$

$$CS_k(V(S), V(P_k)) = \frac{\sum_{i=1}^{n} v(t_i,S)\, v(t_i,P_k)}{\sqrt{\sum_{i=1}^{n} v^2(t_i,S)}\sqrt{\sum_{i=1}^{n} v^2(t_i,P_k)}} \quad (4)$$

The most suitable design pattern for a given problem scenario is selected based on one of the following three cases:

Case #1: The Kth design pattern with the highest value of $CS_k(V(S), V(P_k))$ is selected.

Case #2: The design patterns which satisfy equation 5 are candidates to solve the given problem:

$$|CS_k(V(S), V(P_k))| > \theta \quad (5)$$

Where, θ is a threshold value for the similarity. In this case may be no pattern is selected.

Case # 3: The design patterns which satisfy equation 6 are suggested.

$$|CS_{max} - CS_k(V(S), V(P_k))| \leq \theta \quad (6)$$

Where, CS_{max} is the maximum value of similarity between any design pattern and the given problem scenario, and θ is a threshold. Case #1 was adopted in this work.

4.4. Evaluation Metric

Precision metric was used to assess the proposed approach. Precision is defined by equation (18) as follows:

$$\text{Precision} = \frac{\text{Total number of correctly recommeded design patterns}}{\text{Total number or recommended patterns}} \quad (18)$$

5. Experiments and Results

To evaluate the effectiveness of our approach, two corpus were created one of them includes the textual descriptions of 14 pattern from the catalog of GoF design patterns. Each pattern document includes: the intent, applicability, participants, and collaborators. GOF book in addition to Wikipedia.com were used to prepare a rich description document to each pattern includes the pattern distinctive words. While the other corpus includes 32 real design problem scenarios collected from various sources including various design patterns books, Wikipedia.com and Sourceforge.com. We label each design problem with the fit pattern manually. We meant to have some design problems written briefly or poorly to test the robustness of our approach. Five samples of these design problems are defined as follows:

Design Problem #1: A menu consists of a set of choices and a mechanism for a user to specify which choice they want. There are a variety of styles of menus. One menu style is to print a number in front of each string (e.g., 1, 2, and so on) and let the user enter a number to make a choice. In general, all of the menus must provide a way to add entries to the menu, delete entries, display the menu, and obtain the user's choice.

Design Problem #2: The Company class is the central class that encapsulates several important features related to the system as a whole. It is required to make sure that only one instance of this important class can exist.

Design Problem #3: The system has an interface named "MediaPlayer". This interface is implemented by a concrete class AudioPlayer. AudioPlayer has methods that play mp3 format audio files. There is another interface AdvancedMediaPlayer which is implemented by a concrete class AdvancedAudioPlayer to play vlc amd mp4 format files. It is required to have AudioPlayer class to use AdvancedaudioPlayer class to be able to play other formats.

Design Problem #4: The designer of an adventure game wants a player to be able to take and drop various items found in the rooms of the game. Two of the items found in the game are bags and boxes. Both bags and boxes can contain individual items as well as other bags and boxes. Bags and boxes can be opened and closed and items can be added to or taken from a bag or box

Design Problem #5: The system approves purchasing requests. There are four approval authority. The selection of the approval authority depends on the purchase amount. If the amount of the purchase is higher than 1 million dollar, the owner who approves. If it ranges from 500k to less than 1 million the CEO who approves, if it ranges from 25k to less than 500k the head of department approves, if less than 25k the vice who approves. The approval authority for a given dollar amount could change at any time and the system should be flexible enough to handle this situation.

Design Problem #6: The system should have only one printer spooler although the system can identify many printers.

Pre-processing was performed using the natural language toolkit NLTK [15]. Table 2 shows both of the correct and the first three recommended design patterns (using our approach) for each of the design problems listed above. It was found that the precision of our approach is equal to **65.5%.** Design Problem definitions of failed cases were reviewed and it was noted that these cases do not include descriptive words of the pattern or they are not well written for example design problem#6. It is one of the failed scenarios, it is not well written. Table 3 shows the three highest cosine similarity values for this problem. It could be observed that the highest value tends to be zero which means that the proposed approach was not able to provide a recommendation for this problem. It was observed during experimentation that the precision of the proposed approach could be enhanced through enhancing the description of both of the problem scenario and adding more information to the description of the design patterns themselves.

Table 2. Correct and Recommended Pattern for Sample Design Problems

Problem ID	Correct Pattern	1st recommended Pattern	2nd recommended pattern	3rd Recommended pattern
1	Strategy	Strategy	Singleton	Visitor
2	Singleton	Singleton	Adapter	Prototype
3	Adapter	Adapter	Bridge	Visitor
4	Composite	Composite	Decorator	Bridge
5	Chain of Resp.	Chain of Resp.	Command	State
Precision = 65.5%				

Table 3. Cosine Similarity (CS) Results for Problem#6

Design Pattern	CS
Strategy	0.037
Singleton	0.015
Visitor	0.015

6. Conclusion and Future Work

This paper proposed an approach based on text retrieval for the automatic selection of a suitable design pattern to solve a specific design problem scenario. The experimental results illustrated that the proposed approach is promising; however, the accuracy of the proposed approach is influenced by two main factors. Firstly, the existence of an efficient dataset to the descriptions of the design patterns. This dataset should include as much information as possible. Secondly, the quality of the design problem scenarios. The more the problem scenario includes words from the design pattern descriptions, the higher the probability of selecting the right design pattern. Using a lexical database like WorldNet [18] may alleviate the influence of this factor on the results.

We experimented our approach using the catalog of GOF patterns only but we are currently working on extending our dataset to include more catalogues of patterns like patterns of concurrency, security and real time systems. In addition to considering more features in the vector space model.

References

[1] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design pattern: Elements of reusable object-oriented software. Addison-Wesley.

[2] Hsueh, N. L., Kuo, J.-Y., & Lin, C. C. (2007). Object-oriented design: A goal-driven and pattern-based approach. *J. Softw. Syst. Model*. *8(1)*, 1–18.

[3] Kim, D. K., & Khawand, C. E. (2007). An approach to precisely specifying the problem domain of design patterns. *Journal of Visual Languages and Computing*, *18*, 560–591.

[4] Kim, D. K., & Shen, W. (2008). Evaluating pattern conformance of UML models: A divide and conquer approach and case studies. *Softw. Q. J.*, *16(3)*, 329–359.

[5] Palma, F., Farzin, H., Gu´eh, Y.-G., & Moha, N. (2012). Recommendation system for design patterns in software development: An DPR overview. *Proceedings of the 3rd International Workshop on Recommendation Systems for Software Engineering*.

[6] Pavlic, L., Podgorelec, V., & Hericko, M. (2014). A question-based design pattern advisement approach. *Computer Science and Information Systems*, *11(2)*, 645–664.

[7] Sanyawong, N., & Nantajeewarawat, E. (2015). Design pattern recommendation: A text classification approach. *Proceedings of the 6th International Conference on Information and Communication Technology for Embedded Systems*.

[8] Suresh, S., Naidu, M., Kiran, S. A., & Tathawade, P. (2011). Design pattern recommendation system: A methodology, data model and algorithms. *Proceedings of the International Conference on Computational Techniques and Artificial Intelligence*.

[9] Hasheminejad, S. M. H., & Jalili, S. (2012). Design patterns selection: An automatic two-phase method. *Journal of Systems and Software*.

[10] Hussain, S., Keung, J., & Khan, A. A. (2017). Software design patterns classification and selection using text categorization approach. *Applied Soft Computing*.

[11] Nahar, N., & K. Sakib. (2016). ACDPR: A recommendation system for the creational design patterns using anti-patterns. *Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*.

[12] Smith, S., & Plante, D. R. (2012). Dynamically recommending design patterns. *Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering*.

[13] Gomes, P., Pereira, F. C., Paiva, P., Seco, N., Carreiro, P., Ferreira, J. L., & Bento, C. (2002). Using CBR for automation of software design patterns. *Advances in Case-Based Reasoning, Springer Berlin Heidelberg*.

[14] Muangon, W., & Intakosum, S. (2013). Case-based reasoning for design patterns searching system. *International Journal of Computer Applications*.

[15] NTLK. Retrieved from http://www.nltk.org

[16] Castells, P., Fernandez, M., & Vallet, D. (2007). An adaptation of the vector-space model for ontology-based information retrieval. *IEEE Trans. Knowl. Data Eng*., *19(2)*, 261–272.

[17] Tonella, P., & Antoniol, G. (1999). Object oriented design pattern inference. *Proceedings of the IEEE International Conference on Software Maintenance*.

[18] Wordnet. Retrieved from http://www.nltk.org/howto/wordnet.html

Abeer Hamdy is a Lecturer in the Faculty of Informatics and Computer Science at The British University in Egypt since 2009.

She received her B.Sc. degree with honors, M.Sc. and Ph.D. degrees in Electronics and Electrical communications from the Faculty of Engineering, Cairo University in 1992, 1998, 2003 respectively.

She has been awarded two fellowships from Academy of Scientific Research in Egypt to conduct post doctoral research at University of Connecticut (2005-2006) and University of Central Florida (2007-2008) at the United States.

Abeer Hamdy's research interest includes software engineering , Robotics, Fuzzy systems, Parallel Computing and BitTorrent systems.

Mohamed Elsayed is an assistant lecturer at The British University in Egypt. He received his English BA at the Faculty of Al Alsun, Ain Shams University, and Master of Arts in international education at the Graduate School of Education, The American University in Cairo.

Mohamed Elsayed's research interest includes Educational issues, Teaching Reading, Social work, Practicing sports mainly football and Professional development.