I.J. Modern Education and Computer Science, **2017, 5, 43-49**
Published Online May 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.05.06

Proposed Automated Framework to Select Suitable Design Pattern

M. Rizwan Jameel Qureshi and Waleed Al-Geshari
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: rmuhammd@kau.edu.sa, waleedalgeshary@gmail.com

Abstract—Many design patterns are available in the existing literature. Due to the availability of the enormous quantity of design patterns, it is extremely hard for a developer to find the suitable design pattern to address the problem. An experienced developer can even face problem to select the appropriate pattern for a specific problem and it is no man's land for junior developers. This paper proposes a novel framework that will generate problem-related questions to a developer to find suitable design pattern using a repository. The answers to these questions can guide developers to select the suitable design patterns. This paper uses the questionnaire as a data collection instrument to conclude the results. The results are found supportive indicating that the proposed framework will solve the problem in hand.

Index Terms—Design patterns, repository, programming, automated framework.

I. Introduction

Design patterns are reusable solutions to the occurring software design problems. It is a template or description to solve problems as per the requirements of projects and needs of companies. Many design patterns are introduced after the Gang of Four (GoF) publishes a book (Design Patterns: Elements of Reusable Object-Oriented Software) in 1994. The GOF defines 23 patterns allowing programmers to resolve their problems. The GOF defines name of each pattern to facilitate communication, specific problem in the context of application to apply, solution, and consequences of applying a pattern with respect to results and tradeoffs. The design patterns are divided into three types i.e., creational, structural and behavioral. It is recommended that quality of developed software is increased using design patterns like ease in maintenance, reuse and flexibility to update and upgrade [1]. Many studies are reported to reflect the importance of design pattern on software development using case study, survey and simulation research methods. A further investigation is required to estimate that how much cost, time, effort and resources consume to select a suitable pattern [2].

It merely depends on the experience of the developers to determine the suitable design pattern for a specific problem. Moreover, for the novice programmers, it is extremely hard to find the appropriate pattern as per the problem in hand. Many researchers are trying to find ways to select suitable design pattern for specific situations. Based on problem domain context, Intakosum and Muangon [3] propose a model for programmers to access the suitable design patterns to solve design problems. The proposed model [3] is composed of two parts, the analysis, and the calculation of index weight. The proposed model tested by creating document indexes from GOF design patterns descriptions. These indexes and their weights stored in the database and tested based on 105 queries. The result of this approach is limited to the matching between document, query indexes, and index weight. To improve this model, it needs to performing a query analysis that using techniques such as syntactic, semantic analysis, and case-based reasoning.

In this paper, an attempt is made to study the existing literature to identify the problems relevant to selecting suitable design patterns. The design pattern problem is addressed by proposing a novel framework to automate the process of pattern selection. The proposed framework will make it easy to software development teams to choose the most appropriate design patterns using a repository.

The paper is organized as follows. Section 2 outlines the related work. Section 3 defines the problem. Section 4 presents the details of the proposed solution. Section 5 validates the proposed solution.

II. Related Work

Design pattern definition language (DPDL) is proposed to help developers to implement and share design patterns [4]. DPDL is developed using XML and it can be utilized to define design patterns. By a combination of a natural language, design languages are introduced to define design patterns such as UML. DPDL is easy to use and easy to understand. It gives a clear description of the pattern and it can be extended. DPDL lacks in formalism, and it is difficult to recognize design patterns by using DPDL from source code. It requires translating the code into XML to parse and identify patterns. DPDL is validated through simulation using two open source tools [4]. DPDL also needs tool support to extend and enhance.

Smith and Plante [5] develop a tool so that a programmer can dynamically search for signs using a particular design pattern. The suitable recommendations are provided to a programmer during code development. A format is developed by representing both the structural and behavioral requirements of these patterns. This format is flexible but somehow it is limited in the types of behaviors and structures. In order to expand the patterns, it requires the determination and modulation of more anti-patterns to tool up the recommendations for more design patterns.

Nahar and Sakib [6] present anti-pattern based design pattern recommended (ADPR). ADRP discovers anti-patterns and recommends identical design patterns by using design diagrams. It is only designed to address abstract factory pattern and it also contrasts to the code-based recommender. A prototype of ADPR is implemented in Java to validate the proposed research [6]. The existing anti-patterns use the source code to perform comparative analysis. Presently, the proposed tool needs more work to be extended for other design patterns and generalize the process [6].

The intelligent solutions to a specific software problem are design patterns. The main problem with this approach is the indexing. Therefore, the searching tools for design patterns are needed to solve this problem. Muangon and Intakosum [7] propose a model to solve the indexing problem integrating case-based reasoning (CBR) and formal concept analysis (FCA). This model offers a technique that allows experts to regulate indexes. In addition, it provides a method that keeps the new experiences to solve similar problems those are to be searched in future. Muangon and Intakosum [7] evaluate their research using prototype technique.

Birukou et al. [8] present system for implicit culture support (SISC) to select the appropriate patterns for a specific design problem. It depends on the history decisions that are already taken by other programmers. SISC uses the implicit culture (IC) that supplies recommendations on design patterns. The proposed approach, by Birukou et al. [8], is validated by conducting experiments and it can be enhanced to provide support using advanced recommendation scenarios.

Hasheminejad and Jalili [9] propose a method to identify suitable design patterns. The proposed method is an input design patterns depend on the classification of the text. These inputs are the problem of several design patterns. This method starts in two steps i.e., start to learn the patterns and retrieving the appropriate pattern to a specific situation that is textually described.

Kampffmeyer and Zschaler [10] propose design pattern intent ontology (DPIO). The objective is to formulate and classify the GOF design patterns by their intents. The large number of the patterns are available that makes it hard for developers to find the useful patterns to solve certain design problems. Hence, it also requires tools support to search and find the appropriate design pattern to a certain problem. Kampffmeyer and Zschaler [10] develop Wizard that enables developers to find the applicable design patterns. The Wizard is a stand-alone application. It needs to be integrated with CASE tools to integrate the pattern.

According to the typical of helping the beginner designers, Diaz et al. [11] propose a recommendation system that integrates into a visually for the pattern named VEISIG. This module depends on the cooperative filtering. The information is formulated by expert users of both patterns language and the solutions. The goals organized in design views as assembly, routing, demonstration, personalization, and protection. The system begins rating the design patterns, which are not in the first selection. When all ratings are acquired, the algorithm selects the pattern that has the highest rating. Critically, the recommendation process in this approach is not transparent to the end users.

Cinn ́eide and Nixon [12] develop a new methodology to apply on a set of patterns (from GOF design patterns). This methodology elaborates creational patterns but it is not supporting structural and behavioral patterns. The proposed methodology is effective only if the programmer has a clear rational model.

Dong et al. [13] propose an approach to identify the design patterns based on the description of their structure including pattern matrix and weight. This approach consists of various phases: the first phase- the structural, second phase- the behavioral and third phase- the semantic analyses to minimize fake positives. Dong et al. [13] develop tools called DP-Miner to prove this research. DP-Miner calculates the weight of both matrix and classes to show the relation between them using Java packages. Further, Dong et al. [13] describe to select a pattern based on its behavioral, structural, and semantic analyses. It is a good idea to use data mining matrix algorithms to compute and store design pattern. The data mining algorithm matches a pattern based on the matrix and weight of the system.

Berghe et al. [14] introduce inductive logic programming techniques to choose the appropriate software patterns. These techniques depend on the concise relational models. The big challenge is to design a suitable formal language to represent requirements of the software in a relative learning framework. Issaoui et al. [15] present approach that helps designers during the design process. It helps the designer to select a suitable design pattern. The recommendation patterns guide a suitable design pattern by the number of semantic standard and retrieval of the design pattern intents. Alencar et al. [16] propose component based approach to describe the design patterns to reuse components. The research related to design patterns is supported by limited empirical evaluations [16].

Five patterns are described i.e., Decorator, Composite, Abstract Factory, Observer and Visitor [17]. Design patterns are effective when simple solution is preferred. It is extremely beneficial for the developers to adopt common sense approach to select suitable pattern. Prechelt and Liesenberg [18], Juristo and Vegas [19], Nanthaamornphong and Carver [20] and Krein et al. [21] repeat the same research setting that is conducted for experiment name PatMain in [17] to generalize the results but with a different tool. Thirteen students take part in the case studies those are conducted by Prechelt and Liesenberg [18]. Two case studies are performed by Juristo and Vegas [19] to develop two software systems. The results are replicated that are produced in PatMin [17] to confirm the validity of research. Juristo and Vegas [19] address Abstract Factory, Composite and Decorator patterns. Nanthaamornphong and Carver [20] address Observer, Visitor, Decorator and Composite. Nanthaamornphong and Carver [20] include eighteen students in the case study. Nanthaamornphong and Carver [20] results are dissimilar from the results of PatMin [17]. Krein et al. [21] study is based on Decorator, Composite and Abstract Factory patterns.

Table 1. Summary of the Existing Literature

Title	-	Limitation
Retrieving Model for Design Patterns [3]	-	It needs to be improved to perform a query analysis using techniques such as syntactic, semantic analysis, and case-based reasoning.
Towards design pattern definition language [4]	-	DPDL lacks formalism.
Dynamically recommending design patterns [5]	-	This paper limited in the types of behaviors and cannot deal with cyclical structure.
ACDPR: A Recommendation System for the Creational Design Patterns Using Anti-patterns [6]	-	ADPR is prepared for the recommendation of the Abstract Factory design pattern only.
Case-Based Reasoning for Design Patterns Searching System [7]	-	This model limited to indexing problem.
Choosing The Right Design Pattern: The Implicit Culture Approach [8]	-	The problem is with the implicit information of pattern.
Design patterns selection: An automatic two-phase method [9]	-	The proposed method relies on the number of inconsistencies in the classification of the design patterns group.
Finding the Pattern You Need: The Design Pattern Intent Ontology [10]	-	This ontology is not tested with other catalogs of design patterns.
Using recommendations to help novices to reuse design knowledge [11]	-	The validity of the experiment needs to be judged with an industrial case study.
Automated Software Evolution Towards Design Patterns [12]	-	The tool is beneficial only if a programmer has a clear rational model.
DP-Miner: Design Pattern Discovery Using Matrix [13]	-	All classes presented in a design pattern require being named with pattern-related information
Towards an Automated Pattern Selection Procedure in Software Models [14]	-	The big challenge of this paper is to design a suitable formal language.
A New Approach for Interactive Design Pattern Recommendation [15]	-	This paper does not examine how to offer composition among patterns.
A Pattern-Based Approach to Structural Design Composition [16]	-	It does not focus on behavioral properties of patterns.

The consequence of using design patterns on software development are discussed by performing experiments using JHotDraw software system [22]. It is concluded that design patterns have firm positive effect in improving the efficiency of software development and maintenance. A comprehensive literature review is completed to map and analyze the experimental data about GOF patterns [23]. The existing studies are inefficient to conclude the results that there is coherence between use of design patterns and the resultant software quality. A similar study is conducted to find the effect of GOF design patterns on four software quality attributes i.e., ability to maintain, flexibility, efficiency and correctness [24]. The outcome of the study infers that the effect of design patterns over ability to maintain, flexibility and correctness is adverse but efficiency is improved [24].

An empirical study is performed to examine the impact of selecting appropriate design patterns by developers on the functionality quality attribute [25]. The results are found encouraging and it show that though there is chance of misappropriation while selecting design patterns but use of design pattern have profound effect on software development and maintenance.

A study is presented that how to select a suitable design pattern using a collaborative tool to guide [2]. The collaborative tool contains a set of matching criteria and rules. The research is in its preliminary stage to drive ample results. The proposed tool needs to be tested in an industrial setting to judge its effectiveness. The impact of design patterns on quality attributes is illustrated using mathematical modeling and metrics [26]. Multiple case studies are conducted using open-source software to conclude the results. A decision support system (DSS) is developed to facilitate the developers to select the appropriate design pattern as per the requirements [26]. The proposed DSS needs to select pattern at issue, size of software and requirements regarding quality attributes. The results show increase in reusability and low maintenance if design patterns approach is followed while developing software in an organization. Alghamdi and Qureshi [27] measure the relative ratio between design patterns and software maintenance. A tool is also proposed to simulate the scenarios to evaluate the results.

The limitations of existing literature are illustrated in Table 1.

III. Problem Statement

The huge amount of the design pattern that available makes the developers confused for which design patterns should select to solve their problem. So, the question of this paper is as follows [13-14].

What are the methodologies that help developers to select the appropriate design pattern to solve their problem?

IV. The Proposed Solution

A developer faces serious issues while looking for a suitable design pattern to address requirement. In this research, we propose a novel framework to address this problem as shown in fig. 1. The proposed framework provides a way to select appropriate design pattern based on the attributes of the design pattern that are mentioned by Gang of Four authors (GOF) [1]. The attributes are pattern name, intents, descriptions, applicable, and structure. These elements help the developers to identify the suitable design pattern that can be the better solution of their problem. According to the GOF, the most attributes of the design pattern are shows in Table 2 [27].

Table 2. Design Pattern Attributes

Attributes	Description
Pattern Name	Describe the core of the pattern
Known as	List of the synonyms of the pattern
Intents	Explain the purpose and what the patterns do.
Motivation	This provide simple example of problem and solve this problem by the pattern
Applicable	List the applicability of the pattern
Structure	Diagram and objects
Participant	Describes the responsibilities of the classes and objects
Collaborations	This show how Participants Collaborations
Consequences	Describes the forces that exist with the pattern and the benefits, trade-offs, and the variable that is isolated by the pattern.

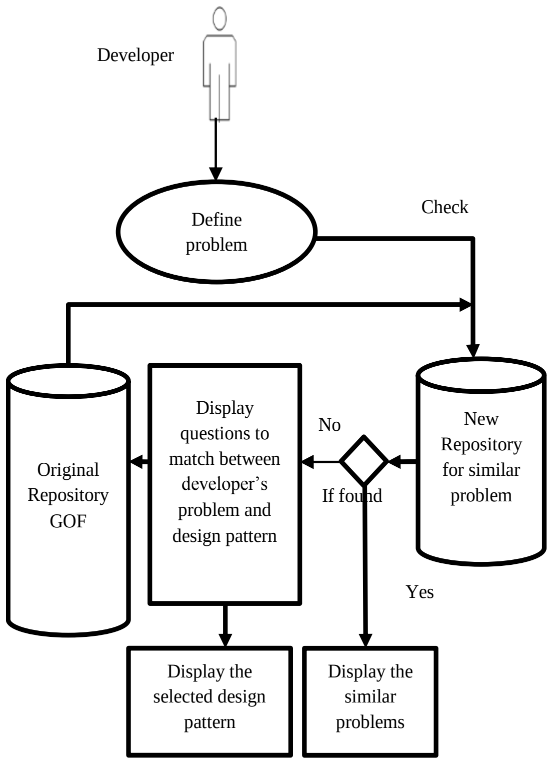
Developer
Define problem
Check
Original Repository GOF
Display questions to match between developer's problem and design pattern
No
If found
New Repository for similar problem
Yes
Display the selected design pattern
Display the similar problems

Fig.1. The proposed Framework.

The GOF Team divides the patterns into three types: A) Creational pattern, B) Structural patterns, and C) Behavioral patterns. This classification spends time for a developer who is looking for the design pattern. The proposed framework addresses the problem in hand by defining three goals.

Goal 1. Identify the relation between design patterns and the problem of a developer.

Goal 2. The experience of a developer.

Goal 3. The effectiveness of design patterns.

A. Goal 1- Identify the relation between the design patterns and the problem of a developer.

To find the relation between a design pattern and the developer problem, need to understand the problem and match with the design pattern based on Design pattern attributes. It will select mare than one design pattern at the beginning. The pattern with the highest matching will be selected from the GOF repository. To make it easier for the beginner developer, this is the core of our research creates the new repository. The new repository stores all problems and design patterns that solve those problems.

For the beginner developer, they need to check the new repository first, if found a similar problem then display its suitable design pattern, if not found a similar problem, in this case, needs to check the GOF repository (Original repository).The criteria to select design pattern from GOF repository depend on answering some questions. These questions build according to the properties of the design patterns. Based on the developer answer, the system decides which design pattern is suitable to the defined problem.

B. Goal 2- The experience of a developer.

Developers can be categorized based on their experiences.

1) Expert Developer.
2) Beginner Developer.

There is a relation to select appropriate design pattern and developer experience. It shows how long time they spent for searching to select the pattern. The expert developer can easily find a solution based on his experience but the beginner takes a long time to find a solution. To benefit from the expert developer for creating the new repository to store all problems and select design pattern. It will help the junior developer for selecting the similar problem.

C. Goal 3- The effectiveness of design patterns.

The effectiveness is the impact of the solution that provided by design pattern. Most of the developers using design patterns to solve problems. This shows how the patterns effectively in software developments. This methodology will formalize depend on the questionnaire that will spread to the developers and students in the college of computer science, and gather the answers to validates our goals.

V. VALIDATION

This research uses the survey methods to validate the proposed solution. The questionnaire consists of 13 questions covers three goals: the first goal is the relation between the design patterns and the developer problem, the second goal is developer experience, and the third goal is the effectiveness of the design patterns. The questions are answered using the likert scale that is ranging from 1 to 5.

- Very low effect indicating 1
- Low effect indicating 2
- Nominal/Average effect indicating 3
- High effect indicating 4
- Very high effect indicating 5

A. *Cumulative Analysis of Goal 1.*

Cumulative responses of the Goal 1, 'Identify the relation between design patterns and the problem of a developer', are shown in Table 3.

50% of the responses were in favor of Goal 1, of which 36% agreed while 13.9% strongly agreed to the effects of the proposed framework in helping to identify the relation between the design pattern and the problem of a developer. 34.4% of the cumulative responses were neutral while 15.5% of the participants were not in favor of it. 8.5% of the respondents disagreed and 6.9% of the software engineers strongly disagreed to Goal 1 as shown in fig. 2.

Table 3. Cumulative Frequency Analysis of goal 1

Q. No.	Very Low	Low	Nominal	High	Very High
Q1	0	2	8	15	6
Q2	1	1	5	12	12
Q3	0	0	19	10	2
Q4	3	2	10	15	1
Q5	0	8	10	13	0
Q6	9	3	12	2	5
Total	13	16	64	67	26
Avg.	6.9%	8.6%	34.4%	36.0%	13.9%

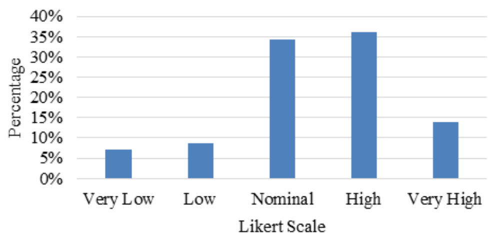

Fig.2. The Cumulative Analysis of Goal 1.

B. *Cumulative Analysis of Goal 2.*

Cumulative responses of the Goal 2 (The experience of a developer) were shown in Table 4. According to Table 4, 18.5% of the responses were in favor of Goal 2 of which 1.6% strongly agreed and 16.9% agreed to it. 51.6% of the respondents were not in favor of Goal 2. 26.6 of the software engineers were disagreed while 25% of the participants were strongly disagreed to Goal 2. Responses which remained neutral were 29.84% as shown in fig. 3.

Table 4. Cumulative analysis of goal 2

Q. No	Very Low	Low	Nominal	High	Very High
Q7	9	5	9	8	0
Q8	4	12	14	1	0
Q9	15	8	7	1	0
Q10	3	8	7	11	2
Total	31	33	37	21	2
Avg.	25%	26.6%	29.8%	16.9%	1.61%

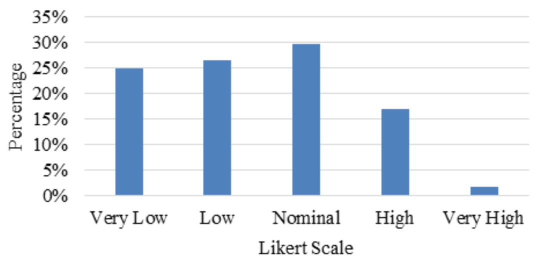

Fig.3. The Cumulative Analysis of Goal 2.

C. *Cumulative Analysis of Goal 3.*

Cumulative responses of the Goal 3 were shown in Table 5. Table 5 showed that 47.2% of the responses agreed with goal 3 in which 6.4% of the software engineers strongly agreed and 40.8% of the professionals agreed with it. 40% of the responses remained neutral for Goal 3. 11.8% of the responses were not in favor of the Goal 3 as shown in fig. 4.

Table 5. Cumulative Frequency Analysis of goal 3

Q. No.	Low	Nominal	High	Very High
Q11	3	11	17	0
Q12	1	20	7	3
Q13	7	7	14	3
Total	11	38	38	6
Avg.	11.8%	40.8%	40.8%	6.4%

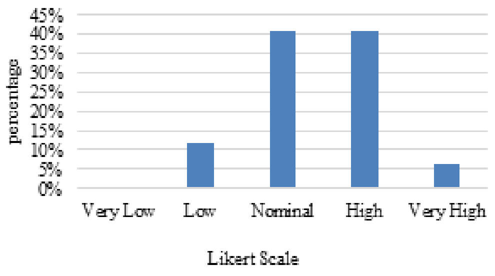

Fig.4. The Cumulative Analysis of Goal 3.

D. *Final Cumulative Analysis of Three Goals.*

Cumulative responses of the three goals were shown in Table 6. Table 6 showed that 38.5% of the responses agreed with 3 goals in which 7.3% of the software engineers strongly agreed and 31.2% of the professionals agreed to it. 35% of the responses remained neutral for 3 goals. A total of 26.2% responses disagreed with the three Goals, of which, 15.6% of the respondents disagreed while 10.6% of the software engineers strongly disagreed with the 3 Goals as shown in fig. 5.

Table 6. Cumulative Frequency Analysis of 3 Goals

Goal No.	Very Low	Low	Nominal	High	Very High
Goal 1	6.9	8.6	34.4	36.0	13.9
Goal 2	25	26.6	29.8	16.9	1.6
Goal 3	0	11.8	40.8	40.8	6.4
Total	31.9	47	105	93.7	21.9
Avg.	10.6%	15.6%	35.0%	31.2%	7.3%

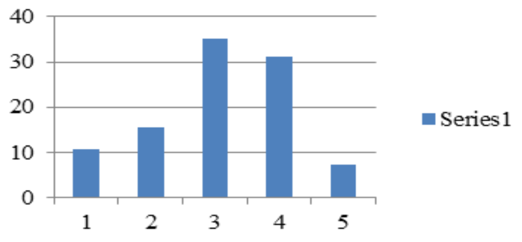

Fig.5. The Cumulative Analysis of 3 Goals.

VI. Conclusion

Design patterns are the repeatable solutions to the software design problems. The developers use the design pattern to solve their problem and to increase the efficiency of their project. The main problem that faces the experience and inexperience developers is how to select the appropriate pattern to a given problem. This paper proposed a novel framework that returns the suitable design pattern to the developer's problem. The result of this paper is validated by questionnaire cover three goals: the relation between the design patterns and the developer problem, the experience of a developer and the effectiveness of design patterns. The proposed framework is validated in using a survey, and it is supported by 40% of the respondents. The results indicate that the proposed framework is acceptable, practical and applicable for software development companies. It is anticipated that the illustrated work will encourage software companies to implement the proposed framework to select suitable design pattern to enhance the organizational productivity. As future work, we would like to build a project to test the proposed solution and prove it.

References

[1] Gamma, E., Helms, R., Johnson, R., Vlissides, J., ‘Design Patterns: Elements of Reusable Object-Oriented Software’ (Addison-Wesley Professional, Reading, MA, 1995).

[2] Nadia, B., Kouas, A. and Ben-Abdallah, H., ‘A design pattern recommendation approach’. Proc. of CORD Conference, 2011, pp. 590- 593.

[3] Intakosum, S., and Muangon, W., ‘Retrieving model for design patterns’. ECTI Transactions on Computer and Information Technology (ECTI-CIT), 2007, 3, (1), pp.51-55.

[4] Khwaja, S. and Alshayeb, M., ‘Towards design pattern definition language’. Software: Practice and Experience, 2013, 43, (7), pp. 747-757.

[5] Smith, S. and Plante, D. R., ‘Dynamically Recommending Design Patterns’. Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE), 2012, pp. 499–504.

[6] Nahar, N., and Sakib, K., ‘ACDPR: A Recommendation System for the Creational Design Patterns Using Anti-patterns’. Proc. of 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016, pp. 4-7.

[7] Muangon, W. and Intakosum, S., ‘Case-based Reasoning for Design Patterns Searching System’. International Journal of Computer Applications, 2013, 70, (26), pp. 16-23.

[8] Birukou, A., Blanzieri, E. and Giorgini, P., ‘Choosing the right design pattern: an implicit cultural approach’ (Technical Report, University of Trento, Italy, 2006).

[9] Hasheminejad, S.M.H. and Jalili, S., ‘Design patterns selection: An automatic two-phase method’. Journal of Systems and Software, 2012, 85, (2), pp.408-424.

[10] Kampffmeyer, H., and Zschaler, S., ‘Finding the pattern you need: The design pattern intent ontology’. Proc. of International Conference on Model Driven Engineering Languages and Systems, Springer Berlin Heidelberg, 2007, pp. 211-225.

[11] Díaz, P., Malizia, A., Navarro, I. and Aedo, I., ‘Using recommendations to help novices to reuse design knowledge’ Proc. Of International Symposium on End User Development, Springer Berlin Heidelberg, 2011, pp. 331-336.

[12] Ó Cinnéide, M., and Nixon, P., ‘Automated software evolution towards design patterns’. Proc. of the 4th International workshop on Principles of software evolution, 2001, pp. 162-165.

[13] Dong, J., Lad, D.S. and Zhao, Y., ‘DP-Miner: Design pattern discovery using matrix’. Proc. of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), 2007, pp. 371-380

[14] Berghe, A., Haaren, J. V., Baelen, S. V., Berbers, Y. and Joosen, W., ‘Towards an automated pattern selection procedure in software models’. Proc. of 22nd International conference on inductive logic programming (ILP 2012), 2012, pp. 68-73.

[15] Issaoui, I., Bouassida, N., and Ben-Abdallah, H., ‘A New Approach for Interactive Design Pattern Recommendation’. Lecture Notes on Software Engineering, 3, (3), 2015, p.173-178.

[16] Alencar, P., Cowan, D., Dong, J. and Lucena, C., ‘A pattern-based approach to structural design composition’. Proc. of 23rd International Annual Conference on Computer Software and Applications Conference, 1999. pp. 160-165.

[17] Prechelt, L., Unger, B., Tichy, W. F., Brossler, P. and Votta, L. G., ‘A controlled experiment in maintenance: comparing design patterns to simpler solutions’. IEEE Transactions on Software Engineering, 2001, 27, (12), pp. 1134-1144.

[18] Prechelt, L. and Liesenberg, M., ‘Design Patterns in Software Maintenance: An Experiment Replication at Freie University at Berlin’. Proc. of 2nd International Workshop on Replication in Empirical Software Engineering Research (RESER), Sept. 2011 pp.1-6.

[19] Juristo, N., Vegas, S., ‘Design Patterns in Software Maintenance: An Experiment Replication at UPM -

Experiences with the RESER'11 Joint Replication Project'. Proc. of 2nd International Workshop on Replication in Empirical Software Engineering Research (RESER), Sept. 2011, pp.7-14.

[20] Nanthaamornphong, A., and Carver, J. C., 'Design Patterns in Software Maintenance: An Experiment Replication at University of Alabama'. Proc. of 2nd International Workshop on Replication in Empirical Software Engineering Research (RESER), Sept. 2011, pp.15-24.

[21] Krein, J. L., Pratt, L. J., Swenson, A.B., MacLean, A. C., Knutson, C. D., and Eggett, D. L., 'Design Patterns in Software Maintenance: An Experiment Replication at Brigham Young University'. Proc. of 2nd International Workshop on Replication in Empirical Software Engineering Research (RESER), Sept. 2011, pp.25-34.

[22] Hegedűs, P., Dénes, B., Rudolf, F., and Tibor, G., 'Myth or Reality? Myth or Reality? Analyzing the Effect of Design Patterns on Software Maintainability' (Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity, Springer, 2012).

[23] Zhang, C. and Budgen, D., 'What Do We Know about the Effectiveness of Software Design Patterns?'. IEEE Transactions on Software Engineering, 2012, 38, (5), pp. 1213- 1231.

[24] Ali, M., and Elish, M. O., 'A Comparative Literature Survey of Design Patterns Impact on Software Quality'. Proc. of International Conference on Information Science and Applications (ICISA), 2013, pp.1-7.

[25] Hsueh, N. L., Wen, L.C., Ting, D. H., Chu, W., Chang, C. H., and Koong, C. S., 'An Approach for Evaluating the Effectiveness of Design Patterns in Software Evolution'. Proc. of 35th Annual Computer Software and Applications Conference Workshops (COMPSACW), July 2011, pp. 315–320.

[26] Ampatzoglou, A., Frantzeskou, G. and Stamelos, I., 'A methodology to assess the impact of design patterns on software quality'. Information and Software Technology, 2012, 54, (4), pp. 331–346.

[27] Alghamdi, F. M and Qureshi, M. R. J., 'Impact of Design Patterns on Software Maintainability'. International Journal of Intelligent Systems and Applications (IJISA), 2014; 6, (10), pp. 41-46.

[28] Suresh, S. S., Naidu, M. M., and Kiran, S. A., 'Design pattern recommendation system (methodology, data model, and algorithms)'. Proc. of International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2011), 2011, pp. 11-16.

Authors' Profiles

Dr. M. Rizwan Jameel Qureshi received his Ph.D. degree from National College of Business Administration & Economics, Pakistan 2009. He is currently working as an Associate Professor in the Department of IT, King Abdulaziz University, Jeddah, Saudi Arabia. This author is the best researcher awardees from the Department of Information Technology, King Abdulaziz University in 2013 and 2016. He is also honoured as a best researcher in Computer Science discipline from seven campuses of COMSATS Institute of Information Technology, Pakistan in 2008.

Waleed M. Al-Geshari received the Bachelor degree in Information Technology from King Abdulaziz University, Jeddah, Saudi Arabia in 2012 and currently, is a Master degree student in King Abdulaziz University, Jeddah, Saudi Arabia.

How to cite this paper: M. Rizwan Jameel Qureshi, Waleed Al-Geshari,"Proposed Automated Framework to Select Suitable Design Pattern", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.5, pp. 43-49, 2017.DOI: 10.5815/ijmecs.2017.05.06