ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/261335857

Design pattern selection: A solution strategy method

Conference Paper · December 2012

DOI: 10.1109/ICCSII.2012.6454337

CITATION
1

READS
60

2 authors:

Eiman Sahly
University of Benghazi
4 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Omar M. Sallabi
University of Benghazi
17 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Multi-Agent System to support design patterns selection View project

A Survey of Reverse Engineering Tools: Towards assisting developers to select the suitable tool View project

All content following this page was uploaded by Eiman Sahly on 28 April 2015.

The user has requested enhancement of the downloaded file.

Design Pattern Selection: A Solution Strategy Method

Eiman M. Sahly
Faculty of Information Technology
University of Benghazi
Benghazi, Libya
eiman.sahly@benghazi.edu.ly

Omar M. Sallabi
Faculty of Information Technology
University of Benghazi
Benghazi, Libya
Omar.sallabi@benghazi.edu.ly

***Abstract*—Design patterns represent a highly effective way to improve the quality of software system, due to its ability to capture the best practices and design knowledge; this enable software engineer to develop their applications. Unfortunately, choosing suitable patterns to specific design problems are difficult task for novice. Over the past years of research in the field of software engineering several conferences and workshops are focused on writing and publishing patterns; this increases the degree of challenges that face the developer in determining the suitable design patterns. This paper presents method to facilitate and assist the designers in selecting the suitable patterns for their problems. We propose a strategy method to obtain the appropriate recommendation.**

Keywords—Design Pattern Selection; Information Retrieval Techniques; Case-Based Reasoning; Recommender Systems.

I. INTRODUCTION

Over the last few decades, witnessed the field of software engineering for industrial applications growth significantly. Recent studies show that software evolution for software systems are increasing in complexity every day [1]. Using of design pattern can build higher software quality, a reduced development cost and ease of maintenance. Design pattern [2] is a generic proven solution to a common recurring software design problem. Moreover, represent a highly effective way to improve the quality of software engineering. Due to its ability to capture the best practices and design knowledge based on real experience in field of software engineering. Basically, only selection suitable patterns achieved design quality. Therefore, experienced software engineering who have a deep knowledge of patterns can a select suitable pattern for a given problem effectively, while this task is very hard for inexperienced one[3].

In fact, novice development has difficult to find the pattern that addresses a particular design problem for several reasons [4]:

- They don't have concrete definition or a clear understanding of what the problem domain is.
- The novice not familiar with deign pattern and not having enough knowledgeable about it for decide whether reuse patterns or develop a special-purpose solution.

Almost fifteen years ago, the Gang of Four (GoF) book [2]; stated the problem of selecting patterns to find the suitable from more than 20 existing pattern. Since then, the problem of selecting patterns still exists. Currently, there are growing number of proposed design patterns is continuously introduced in several researches, conferences and workshops (such as Euro-PLoP1). With this rapid growing it's extremely difficult for designers to choosing patterns suitable for their problems.

This paper addressed the problem of pattern selection. The proposed method is based on several algorithms depending on the scenarios situation. In this work, several techniques are used such as Case-Based Reasoning (CBR) and Formal Concept Analysis (FCA), Information Retrieval (IR) Techniques, Implicit Culture (IC).

Our aim, in this paper, is to provide assistance to both novice and expert designers in the following:

- Assisting novice designers by giving them the necessary recommendations and suggestions for a given problem.
- Present guidelines to support how to implement pattern.
- Expand the knowledge of novice about pattern and how to solve design problems, through some recommendations given to them.
- Increase their ability to learn new design patterns through navigations from one pattern to another.
- A collaborative environment facilitating experience sharing among developers, as well as support knowledge transfer to the novice.

This paper is organized as follows: In section 2, gives some related work of our research. Present our method solution for patterns selection in section 3. Finally, section 4 summarizes the conclusions of this paper and future work.

II. RELATED WORK

There are several methods and approaches attempted to define a way to select a suitable design pattern. From the available literature these can be classified as: pattern detection (e.g. [5],[6]), patterns recommendation based on queries (e.g.

[1] EuroPLoP™ is the premier European conference on patterns.

978-1-4673-5157-7/13/$31.00 ©2013IEEE

[8],[9],[10], [12]), Mixture between detection and select pattern (e.g. [7]) , and automatic pattern selection (e.g. [11]).

Shi et al [5], presented PINOT tool; fully automated pattern detection from existing source code; based on reclassification of the GoF patterns from catalog classifies for forward-engineering to catalog classifies for reverse-engineering by their pattern intent.

Zhu et al [6] presented LAMBDES-DP tool. Its theoretical foundation is a descriptive semantics of UML in first order logic, and the used repository of GoF design patterns. The tool uses the LAMBDES system to translate the proved UML into their descriptive semantics and invokes the theorem prove SPASS to decide whether the design conforms to a pattern.

Nadia et al [7] presented approach assists the designers choosing their appropriate design patterns. Their approach was supported by an interactive tool. The tool give a bitty to draw a design fragment, present the problem, re-phrases the problem in order to obtain the intention of a certain pattern and use of WordNet[2] which lexical dictionaries. Then, it explores the candidate solutions by filtering patterns that meet the intentions through the use of recommendation rules.

Birukou et al [8] presented an approach use of a multi-agent system (MAS) based on the Implicit Culture (IC) framework to help designer in selecting patterns by getting suggestions from the group. Repository built only on 23 design patterns from GoF book. Depending on the match between problems of design pattern selection with problem of selecting web links relevant to keyword. Therefore, they used the simulator developed for the application of the IC Multi-Agent Platform to web search. The results have shown that an increase in the number of users causes an increase in the recall of the suggestions produced by the system.

Guéhéneuc et al [9] presented a first prototype as a simple recommender system for design patterns. This approach is based on extract important words by analyzing the textual GoF patterns. In their application they didn't use collaborative filtering and feedback from the users, rather they just choose important words.

Díaz et al [10] proposed a module for a recommendation tool embedded in a visual environment, according to a preliminary study that recommends pattern based on the selection designers, used collaborative filtering techniques. With a view to supporting designers in making the correct design decision. It supports a novice user to understand a design patterns language.

Hasheminejad et al [11] proposed a method based on the text classification approach. It has simple automation of pattern selection process, for suggest the suitable design patterns according to the degree extent of similarity between the problem definitions of the retrieved design pattern. The proposed method has four phases, preprocessing, learning design patterns classifiers, determination of a design pattern classes, and suggestion of design pattern(s).

[2] http://wordnet.princeton.edu/

Suresh et al [12] proposes a prototype methodology to find a suitable pattern to the user. The methodology has two search scenarios and the three algorithms for finding the suitable pattern. They used information retrieval techniques and social recommendations to recommends pattern. This paper is the closest to our work.

III. METHOD OF SOLUTION

Before describing the proposed method, let us firstly classify user's levels. In Dreyfus model [13] provide a definition of levels from novice to expert (Novice, Advanced beginner, Competent, Proficient, and Expert). At present, we are using three levels (Novice, Advanced beginner and Expert). In order to determine the user's level, three questions will help us to determine his/her level. Table I summarizes these questions, and the list of answers that used to identify user's level.

TABLE I. QUESTIONS ABOUT DETERMINE THE LEVEL OF USER

Q1	How many design patterns do you know?
	None – less 3 - 23GoF – all 23GoF and more.
Q2	Do you work with design pattern?
	Not at all - Sometimes – Always
Q3	Are you write and publish your design patterns?
	No – yes

From the previous table, Q1 focuses on identifying the user's level (i.e. user is a novice user if the choice none or less than 3, then there is no need to answer the further questions Q2 and Q3). If a user selects (23GoF or all 23GoF and more), then the user is not a novice user but, answer of Q2 is required to identify his/her other level. Furthermore, Q2 is used to distinguish between advanced beginner and expert. The (not at all) option means novice group and the option (Sometimes) is advanced beginner group but the option (Always) is not certain that an expert group. The last question, Q3 confirm that the user in the expert group.

In the rest of this section, we describe our method to solve the problem of selecting appropriate design pattern. In essence, our proposed method consists of four iterative steps; Fig. 1 illustrates a conceptual overview of it.

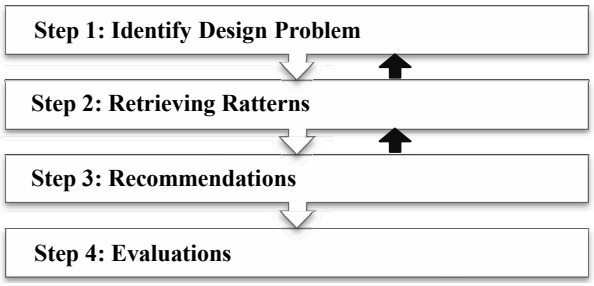

Figure 1. The steps of solution strategy method.

A. Step 1: Identify design problem

The users identify the descriptions of their problem through submit queries to obtain the suitable patterns. The proposed method depends on the user's level that has registered on as explained above.

B. *Step 2: Retrieving patterns*

In our current work, finding suitable pattern is based on many strategies, and it depends on four scenarios situations. Fig. 2 illustrates situation four possible scenarios for finding the suitable pattern.

i. **If** user (Novice or Advanced beginner or Expert) is submits Query **then** executes **Scenario1**: apply ***QMP*** algorithms (ii).

ii. **If** ***QMP*** not found appropriate pattern **then** executes **Scenario2**: apply ***QSPQ*** algorithms (iii).

iii. **If** ***QSPQ*** not found appropriate pattern then

Check the user **if** ***Expert*** user **then** executes **Scenario4** **and** apply ***CIK*** algorithms.

Else executes **Scenario3** and apply ***QAS*** algorithms (iv).

iv. **If** ***QAS*** not found appropriate pattern then executes **Scenario4** and apply ***CIK*** algorithms

Figure 2. Situation four possible scenarios for finding the suitable pattern.

From the previous figure, four algorithms have been used (QMP, QSPQ, QAS and CIK). These algorithms will be explained in next subsection. Basically, the system starts scenario1 when submitting a query via (novice or advanced beginner or expert) user to search pattern needed by matching query intent with pattern intent. However, sometimes users' queries fail to describe completely their requirements clearly. Hence, continue scenario2 to search similar query with previous user's queries, if similar query is found and satisfies minimum thresholds of confidence then suggest the same pattern, if not found satisfactory patterns from similar query then checking in user level; if the user is an expert then continue with scenario4. Else, continue with scenario3; question and answer session would begin. May not find the appropriate pattern for several reasons, some of patterns maybe don't have the knowledge base in repository or maybe the users do not specify correct domain of the problem. Therefore, continue with scenario4 with interaction collaborative which present the problems on the group of experts to find a solution for design problem.

1) *Algorithms*

The purpose of following algorithms is to find the suitable pattern based on the user's query to solve a specific design problem.

a) *Algorithm 1:* Query-Matching-Pattern (***QMP***)

Purpose: The purpose of this algorithm is to parse and analyze the text user's query to find matching between patterns intents and the given query.

Basically, the analysis process of a query text is like removing stop words (i.e. stop words are common words like a, an, the, and, etc.) and stemming removes words suffixes like: (a) "s, es" from nouns, (b) the 'ing' from verbs, and etc. As a result, the algorithm returns a list of patterns. Fig. 3 shows the pattern of recovery processes using the QMP algorithm. We propose the use of Information Retrieval (IR) techniques that allows users to search for information based on their needs [14]. This allows the retrieval of design patterns from the specific design problems. The most common model and classic is Vector Space Model (VSM). This model is an algebraic model for representing text documents and user queries as vectors concept [15]. All documents and queries are represented as m-dimensional vector spaces. Through compute the cosine value of the angle between document (pattern) and query vectors to determine the similarity between them. In practice, it is helpful to use retrieval techniques from open source for IR such as Dragon Toolkit[3] and Lecene[4].

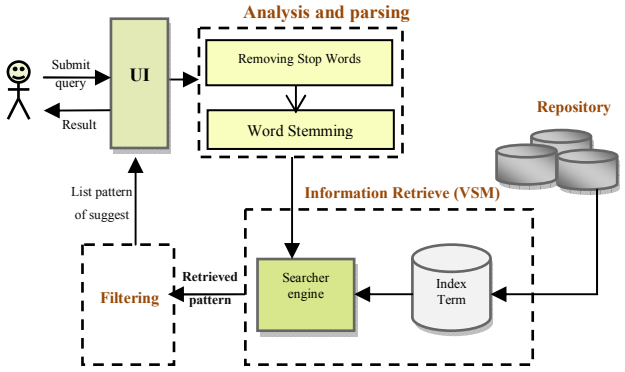

Figure 3. Illustrates the conception of the work of the QMP algorithm

In fact, the success of suitable pattern retrieval depends on several factors such as search and index. Searching by keywords, it is the possibility to specify where the keyword must be present in the pattern description. Searching uses additional data such as requirements, properties and quality goals addressed by patterns. Unfortunately, inexperienced cannot define their problem clearly, that causes keyword-search problem [16]. On the other hand, indexes affect the quality of retrieval maybe low in the right documents when the index appears in many documents. Hence, in this work the ideas which inspired by [15] are proposed to categorize the indexes to the domain, functions and use situations the solution offered by the design pattern. The elements indexes are based on intent pattern, problem statement and applicability. Fig. 4 shows the restructure a design pattern description document.

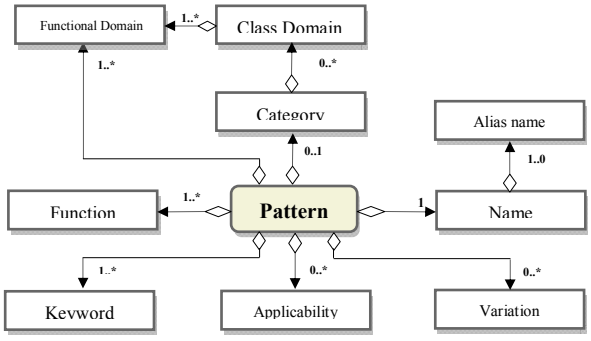

Figure 4. Illustrates proposed meta-data of pattern description

[3] http://dragon.ischool.drexel.edu
The Dragon Toolkit is a Java-based development package API .

[4] http://lucene.apache.org
High-performance, full-featured text search engine library. Based on the Vector-Space Model and the Boolean-Model in IR.

From the previous figure, each element can be explained as follows:

- *Name*: an index that contains the representation of a design pattern name.
- *Alias name*: other well-known names of the pattern.
- *Category:* an index that contains each design pattern in one category.
- *Class Domain:* contains indexes that represent the subset of patterns such as HCI, software, security, business, analysis and architecture.
- *Functional Domain:* contains indexes that can refer to major function of a design patterns class domain. The Patterns can be applied in different functional domains.
- *Function:* contains indexes that refer to major functions or actions stated in pattern intent.
- *Keyword:* contains indexes that illustrate specific features of each pattern.
- *Applicability:* contains indexes that refer to various domains in various situations. The applicability ensures whether pattern is right for a specific problem or not.
- *Variation*: indexes that contains refer to other variations of pattern for e.g. proxy pattern has many variations such as virtual proxy, remote proxy.

This information will be used as one criterion to specify priority level of indexes. The priority level shows quality of indexes to identify proper design patterns [15].

b) Algorithm2: Query-Similarity-PreviousQuery(***QSPQ***)

Purpose: This algorithm works on search similarity between the requested query and previous users' queries to find recommends pattern.

This algorithm is useful and can be applicable to solve new problem similar to previous problems. One way to solve new problem by use the past knowledge is Case Based Reasoning (CBR) techniques. CBR [16][17][18] is a powerful problem-solving approach that uses past experiences to be knowledge for solving similar problems. CBR cycle [16] process for to solve a new problem by retrieve the most similar cases from knowledge-base (case base, case memory), and reusing it in the new problem situation; and revising the solution and store new experience in base case to use in the future problem solving (learning).

However, regarding the keyword-search problem is may can appear when you applying the case base reasoning to solve new problem. Therefore, we propose applies approach of Formal Concept Analysis (FCA) support Case-Based Reasoning (CBR) in [19][20][21]. They use FCA as the way of inductive for extracting embedded knowledge in case base and allows the flexibility to maintain index. FCA is useful for learning index by used case organization that analyze case base for obtain knowledge embedded through suggest suitable indexes to use for new problem.

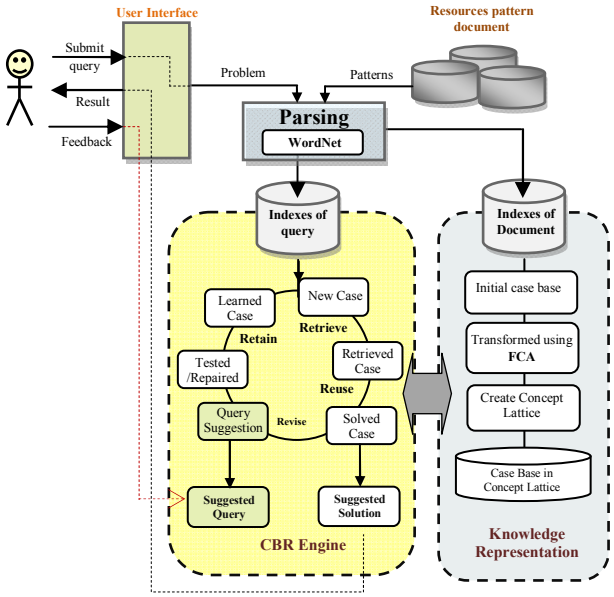

Figure 5. Illustrates the conception of the work of the QSPQ algorithm

Traditionally, the user submits a query which includes a description of the problem in full-text form. It will be parsed into index of problem same method of pattern document processing as shown in Fig. 5. Hence, the mechanism that allows a reformulation of the ad hoc problem of the user, thanks to the use of large lexical database of English (such as WordNet). Subsequently, these indexes are used to retrieve case in the knowledge base with similarity function. Then, suggest the solution of the problem through reuse of retrieved case similar to the problem. In addition, use the obtained knowledge base to revise the problem, can be suggested to fulfill the problem by using user's feedback. Hence, the modified problem will be used to compare with previous cases again. Finally, the new problem and its solution are decided to retain in case base.

c) Algorithm 3: Question-Answer-Session (***QAS***)

Purpose: This algorithm focuses on finding the pattern based on question-answer session that helps narrow down the selection process, questions will vary according to the previous answers of the user and calculates weights obtained for to recommendations suitable as shown in Fig. 6.

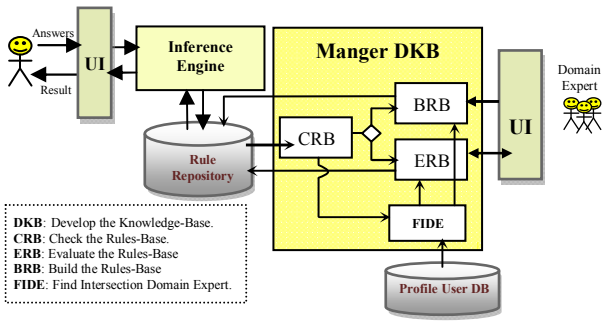

Figure 6. Illustrates the conception of the work of the QAS algorithm

However, we propose applies approach partition the questions into different levels in [22]. In order to reduce of the

search space for select a design pattern. The questions are divided among the following four levels:

Pattern-domain questions ⟶ Level 1
Pattern-category questions ⟶ Level 2
Pattern-intent questions ⟶ Level 3
Pattern-specific questions ⟶ Level 4

In particular, we propose to develop the knowledge-base by experts' users which enrolled in the system and not just rely on the system administrator. More specifically, the development process rule-bases are done by a group of experts who intersect experience in the same domain as shown in the Fig. 7, for develop the rule-base of pattern.

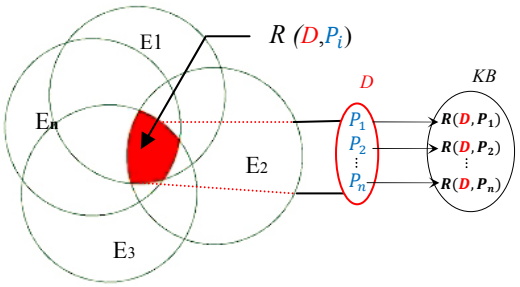

Figure 7. Illustrates process to evaluate and develop the knowledge-Base

There is a relation between design pattern and application domains. That can be defined in formal logic as (∃P)(P ∈ D) where P is a design pattern, and D is application domains in different fields such as (software architecture, communications, user interface, mobile application, real-time system, etc.). Also, we can describe relation between expert users E with D as (∃D)E(D). Consequently, each pattern has rules (R) in the knowledge-base (KB). Also, E intersects experience in same D, which can be defined as $E_1(D, P_i) \cap E_2(D, P_i) .. \cap E_n(D, P_i)$. Therefore, it can be written as:

$$\cap_{k=1..n} E(D, P_i) \quad (1)$$

Where,

$$P_i; i = 1,2,3..m , i \in D.$$

Therefore, a simple measures rules are defined that can drive development or evaluate of the knowledge-base automatically. The rule can be in one of the following two forms.

$$R(D, P_i) \notin KB \rightarrow \text{Build}[\cap_{k=1..n} E(D, P_i)]: i \in D \quad (2)$$

$$R(D, P_i) \in KB \rightarrow \text{Evaluation}[\cap_{k=1..n} E(D, P)]: i \in D \quad (3)$$

Note that the value of $R(D, P_i)$ is building R of P_i by users defined in the form (1), if P_i is don't have rule in KB as in the form (2). Alternatively, in the form (3), $R(D, P_i)$ is evaluated by users defined in the form (1), when P_i have R in KB.

In the above, we use mathematical notations [23], such as set membership (∈), existential (∃), not set membership(∉), implication (→), intersect (∩), and definition (:=).

d) Algorithm 4: Collaborative-Implicit-knowledge (**CIK**)

Purpose: The purpose of this algorithm is the transfer of implicit knowledge through collaboration with communities of users to find the pattern as shown in Fig. 8. Receives a request from the user includes a description of the problem and which domain. The problem is displayed on the blackboard and domain sends to the observer, the observer sends queries to the database for users interested in the same domain. Subsequently, it sends a list of those users to the user sponsor for interact with users and send all suggestions solution of the problem presented in the blackboard to observer.

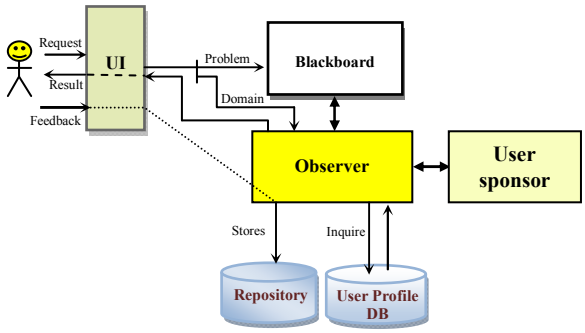

Figure 8. Illustrates the conception of the work of the CIK algorithm

The observer used rules to store documentation of new knowledge. It can be represented in pseudocode as follows:

1. **If** *request* (U_x; problem _*a*; domain _*b*) **then**
   [*Suggest* (U_y; $n \times$ pattern); *] // $n = 1|more$.
   // meaning (*), any suggestions from other users.
2. Compute a score of weighted pattern for each suggestion submitted.
3. **If** *Accept* (U_x; *Suggest* (U_y; $n \times$pattern)) **then**
   Strong (U_x; problem _*a*; domain _*b*; [*Suggest* (U_y; $n \times$ pattern); *; PWS]) // PWS = pattern weighted score.

Figure 9. pseudocode of observer rules

It means that if user _x is requesting for the problem _a specified with domain _b, then the observer returned suggested of one or more from user (_y is user who is interested domain _b). Afterwards, from this suggestion is calculated the pattern weighted score. From higher score, the user _x sends to the observer to inform accept suggest user _y for the problem _a. accordingly; the observer stores this data in the repository.

C. Step 3: Recommendations

The recommendations contribute significantly to raising the efficiency of users through influencing their selection. There are three types of recommendations supported by the proposed method:

- Recommending patterns: provide suitable patterns for solving a particular problem.

- Recommending how apply patterns: Provide recommendations how to implement the pattern (s) (i.e. by giving guidance how to implement pattern through suggesting a list of users who have used the pattern before).
- Recommending pattern sequences: Showing the recommendations consist of sequences of patterns usually apply together that depending on relationship between patterns (patterns language).

D. Step 4: Evaluation

In the last step, we evaluate process the effectiveness of this method. The main purpose of the evaluation is to assist of identifying necessary improvements for the proposed method. Using a questionnaire to ask questions to the users to find out whether the system helps choose the appropriate patterns as well as suggestions for improvement. The following table provides a few these questions:

TABLE II. QUESTIONS ABOUT EVALUATE THE RECOMMENDATIONS

No.	Questions of the evaluation
Q1	Does the recommendation help to solve the problem?
Q2	The use of our system is easy?
Q3	Are you satisfied with the proposed solutions?
Q4	What is your assessment of the recommendation?
Q5	How useful was the system to understand the patterns?
Q6	How would you improve the quality of the solutions?

IV. CONCLUSION

In this paper we discussed about our solution method for facilitating the selection design pattern and assists inexperienced to understand a design patterns and experienced learn about new pattern that it might produces produce better solutions or more complete solutions. The proposed method can be serving as a guide for new developers/designers to find their needed design pattern according to recommendations given to them. As a future work, we propose an implementation of our solution method based on Multi-Agent System (MAS) that aid to enhance recommendations for selecting design pattern effectively.

ACKNOWLEDGMET

We convey our sincere thanks to Mohamed Hagal for his many insightful comments. We would also are grateful to Prof. H. Alzubaidy for his valuable discussions in mathematics and logic. We would also like thank to Rabeia N. Abdleati for her reviewers, feedback and suggestions to improvement.

REFERENCES

[1] M. Bhatti, N. Anquetil, S. Ducasse, "An Environment for dedicated Software Analysis tools", ERCIM News 88, p. 12, January 2012.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Boston, MA, USA: Addison-Wesley, 1995.

[3] I. Sommerville, Software Engineering: Addison-Wesley, Boston, MA, USA, 7 th ed, 2004.

[4] D.-K. Kim and C. El Khawand, "An approach to precisely specifying the problem domain of design patterns," in Journal of Visual Languages and Computing 18 (6), 2007, pp. 560-591.

[5] N. Shi and R. Olsson, "Reverse Engineering of Design Patterns from Java Source Code," in 21st IEEEACM International Conference on Automated Software Engineering ASE06, Washington, DC, USA, 2006, pp. 123-134..

[6] H. Zhu, I. Bayley, L. Shan, and R. Amphlett, "Tool Support for Design Pattern Recognition at Model Level," in Annual IEEE International Computer Software and Applications Conference, Oxford, 2009, pp. 228 - 233

[7] B. Nadia, A. Kouas, H. Ben-Abdallah, "A design pattern recommendation approach", CORD Conference Proceedings, pp. 590-593, 2011.

[8] A. Birukou and M. Weiss, "Patterns 2.0: a Service for Searching Patterns," EuroPLoP 2009, Irsee Monastery, Bavaria, Germany, July 8-12, 2009.

[9] Y. Guéhéneuc and R. Mustapha, "A simple Recommender System for Design patterns," Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories, July 2007.

[10] P. Díaz, A. Malizia, I. Navarro, I. Aedo, "Using Recommendations to Help Novices to Reuse Design Knowledge," IS-EUD, vol. 6654, pp. 331-336, 2011

[11] S. Hasheminejad and S. Jalili, "Design patterns selection: An automatic two-phase method", Journal of Systems and Software, vol. 85, pp. 408–424, February 2012.

[12] S. S. Suresh, M. M. Naidu, S. Asha Kiran et al., "Design Pattern Recommendation System: A Methodology, Data Model and Algorithms," presented at the ICCTAI'2011, 2011.

[13] P. Benner, "From Novice To Expert," AJN, American Journal of Nursing, vol. 82, pp. 402-407, March 1982.

[14] R. Baeza-Yates and B. Ribiero-Neton, Modern Information Retrieval: Addison Wesley, 1999.

[15] I. Sarun and M. Weenawadee, "Retrieving Model for Design Patterns," ECTI Transactions on computer and information technology, Vol. 3, No.1, pp.51-55, May 2007.

[16] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," AI Communications. IOS Press, vol. 7, pp. 39- 59, March 1994.

[17] B. Ralph, J. Kolodner and E. Plaza, "Representation in case-based reasoning," Knowledge Engineering Review, vol. 20, pp. 209-213, 2005.

[18] A. Aamodt, "Case-based reasoning - an introduction," A newforce in advanced systems development, Unicom Seminars, pp. 9-23, April 1995

[19] I. Sarun and M. Weenawadee, "Adaptation of Design Pattern Retrieval Using CBR and FCA," Computer Sciences and Convergence Information Technology, ICCIT '09. Fourth International Conference on, pp. 1196 - 1200 Nov. 2009.

[20] I. Sarun. and M. Weenawadee. "Retrieving Design Patterns by Case-Based Reasoning and Formal Concept Analysis," ICCSIT International Conference on Computer Science and Information Technology, Vol. 4, August 2009, pp. 424-428.

[21] B. Díaz-Agudo, M. A. Gómez-Martín, P.P. Gómez-Martín and A. P.A. González-Calero, "Formal Concept Analysis for Knowledge Refinement in Case Based Reasoning," in *AI-2005, the 25th International Conference on nnovative Techniques and Applications of Artificial Intelligence*, 2005, pp. 233-245.

[22] D. C. Kung, H. Bhambhani, R. Shah, G Pancholi, "An Expert System for Suggesting Design Patterns - A Methodology and a Prototype," In T. M. Khoshgoftaar, Software Engineering With Computational Intelligence, Kluwer International, 2003.

[23] D. C. Goldrei, Classic Set Theory: For Guided Independent Study. London: Chapman and Hall/CRC 1996.

View publication stats