2018

TOWARDS MORE ACCURATE AUTOMATIC RECOMMENDATION OF SOFTWARE DESIGN PATTERNS

Abeer Hamdy
The British University in Egypt, abeer.hamdy@bue.edu.eg

MOHAMED ELSAYED
The British University in Egypt

Journal of Theoretical and Applied Information Technology
15th August 2018. Vol.96. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TOWARDS MORE ACCURATE AUTOMATIC RECOMMENDATION OF SOFTWARE DESIGN PATTERNS

ABEER HAMDY[1,2, a] , **MOHAMED ELSAYED** [1, b]
[1]Faculty of Informatics and Computer Science, British University in Egypt, Elshorouk city, Egypt
[2]Computers and Systems Departments, Electronics Research Institute, Cairo, Egypt
Email: [a]Abeer.hamdy@bue.edu.eg , [b]Mohamed.sayd@bue.edu.eg

ABSTRACT

Design pattern is a high-quality reusable solution to a commonly occurring design problem in certain context. Using design patterns in software development improves some of the quality attributes of the system including productivity, understandability and maintainability. However, it is hard for novice developers to select a fit design pattern to solve a design problem. The paper proposes a text retrieval based approach for the automatic selection of the fit design pattern. This approach is based on generating a vector space model (VSM) of unigrams and topics to the catalogue of patterns. The topic is a set of words that often appear together. Latent Dirichlet Allocation topic model is adopted to analyze the textual descriptions of the patterns to extract the key topics and discover the hidden semantic. The similarity between the target problem scenario and the collection of patterns is measured using an improved version of the popular Cosine similarity measure. The proposed approach was assessed using Gang of four design patterns catalog and a collection of real design problems. The experimental results showed the effectiveness of the proposed approach which achieved 72 % precision.

Keywords: *Design Pattern Selection; DP Recommendation; Gang of Four, Text Mining, Information Retrieval, Topic Modelling, Vector Space Model.*

1. INTRODUCTION

Software design is the most challenging activity in the software development life cycle. Design patterns are standardized and well documented best practices used by experienced software developers. Using patterns in software development leads to an increase in software reusability, quality and maintainability, in addition to reducing the technical risk of the project by not having to develop and test a new design [1], [2].

However, the existence 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 suggests to the developer the right design pattern for a given design problem during the design phase becomes a necessity.

Recently, a number of research studies were conducted to address this issue. Some of these studies developed techniques for suggesting the suitable pattern based on analyzing the UML design diagrams [3], [4]. Other techniques are interactive or semi-automatic techniques where the user is provided with a set of questions formulated from the design pattern descriptions after which, the fitting design pattern is determined according to the answers provided by the user [5], [6]. Some studies used text classification and text retrieval techniques [7], [8], [9], [10] while 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 usage stored in a knowledge base in the form of cases [13], [14].

This paper is contributing in the proposal of a novel approach for automating the process of selecting the fit design pattern (from a repository of patterns) to solve a given design problem. The proposed approach is based on extending the unigram features of the traditional vector space model [15], [16] with topics generated by Latent Dirichlet Allocation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(LDA) topic model [17], [18]. The recommended pattern will be the one most similar to the target problem. Similarity is measured using a modified version of the Cosine similarity called Improved Sqrt-Cosine similarity measure (ISCS). The motivation for this approach could be summarized as follows:

1. The proposed approach 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.
3. The ability of topic models to analyze a collection of documents to find out the patterns of word-use (topics) and how to attach documents that share similar topics.
4. Topic model which were used to solve other similar problems in software engineering like mining bug report repositories for the purpose of automating bug triage [19]. Bug reports are short documents written in natural language same as design problem scenarios. Mining bug repositories is similar to mining a repository of design pattern descriptions.

The structure of the paper is as follows: Section 2 introduces Gang of four design patterns. Section 3 explains the proposed approach while section 4 discusses the experiments and the results. Section 5 discusses the previous work in the field of design pattern selection. Finally, section 6 concludes the paper and recommends ideas for extending this work.

2. DESIGN PATTERNS

The concept of design pattern was initiated in software development in 1994 when four software engineers 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. Gamma et al. [5] proposed a two dimensional matrix categorization to the patterns based on two criterion which are purpose and scope. They classified the patterns based on purpose into three categories which are creational, behavioral and structural, while they classified the patterns based on scope into Class inheritance and Object composition patterns. Research studies were conducted to classify the design patterns in general for example Zimmer [20] organized GOF patterns using three types or relationships which are "uses", "is similar to" and "can be combined with" for example: Composite pattern can be combined with Iterator pattern, Visitor pattern is similar to Iterator pattern. Gamma et al. [1] defined a template to describe the patterns which 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 includes mainly the class diagram that describes the static structure of the pattern, description of the consequences of applying the pattern and the anti-patterns. Table 1 shows the description of the Class Adapter design pattern.

3. METHODOLOGY

Our proposed approach is based on analyzing the corpus of pattern descriptions to extract the topics, then transferring each design pattern description and each problem scenario into a vector of features. These features are unigrams and topics. The selected pattern is the one whose vector is the nearest to the problem vector. Figure 1 illustrates the steps of the proposed approach which starts with the textual preprocessing to the corpus of pattern descriptions followed by two parallel processes which are: 1) Topics extraction through training an LDA model, then the generation of a vector space model of topics (LDA VSM) for the patterns. 2) Building a vector space model of unigrams (Unigram VSM). Both of the unigram and the LDA vector space models are concatenated such that each pattern will be represented by a vector of features which includes unigrams and topics. The built LDA model and the unigram VSM are used in generating a vector of features for each target design problem scenario after the scenario is preprocessed. Finally, the similarity between the feature vector of the problem scenario and each design pattern feature vector is computed and the selected pattern is the one closest to the problem scenario. The following subsections discuss these procedures in more details.

3.1. Preprocessing

Each design pattern description and each design problem scenario is pre-processed through three activities which are: Tokenization then Noise Removal then Stemming [15] , [16], as depicted by figure 2. Tokenization process splits each document at the delimiters into unigrams (tokens). Afterwards, all the tokens are transferred into the lower case such that words like "Object", "object" and "OBJECT" are treated the same. Noise removal stage disregards the non-descriptive words like linking verbs and pronouns. These non-descriptive words are considered noise as they increase the size

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of the Vector Space Model and do not contribute to the retrieval process itself. Finally, the words are normalized to their root forms through Stemming. For example a stemmer can reduce each of the words "creating" and "created" to the word "create". Porter stemming algorithm [11] was adopted in our work.

Table1. Problem and solution domains of Class Adapter design pattern

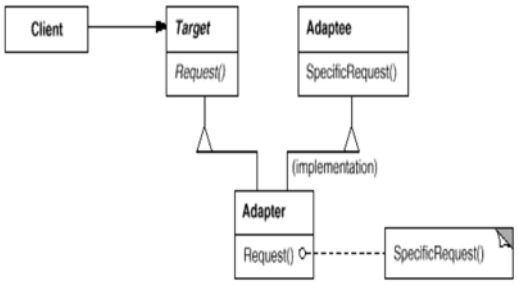
Intent: Change the interface of a class into another interface. It let the classes work together without modifying their source code.

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.

3.2. Unigram VSM

In this stage each pattern (DP) is represented as a vector of unigrams. All the vectors have the same size which is equal to the number of unique words in the corpus of pattern descriptions. To build the Unigram VSM [16], all the unique words in the corpus of patterns are collected and each word is given an index. The pattern vector will have zeros in the cells that correspond to the words that do not exist in the pattern description and ones in the cells that correspond to the existing words. Equation 1 shows the unigram VSM of N design patterns in a space of M unique words.

$$\text{Unigram VSM} = \begin{matrix} \\ DP_1 \\ \cdot \\ \cdot \\ DP_N \end{matrix} \begin{matrix} W_1 & \ldots\ldots & W_M \\ \end{matrix} = \begin{bmatrix} P_{11} & \cdots & P_{1M} \\ \vdots & \cdots & \vdots \\ \cdot & \cdots & \cdot \\ P_{N1} & \cdots & P_{NM} \end{bmatrix} \quad (1)$$

Where, P_{KL}= {0, 1}, $\{W_1, \ldots\ldots W_M\}$ are the unique key words extracted from the pattern descriptions.
And DP_K= $\{P_{k1}, P_{k2} \ldots\ldots P_{kM}\}$

In order to enhance the performance of retrieving the correct pattern and clean the noises of the corpus; a feature weighting scheme was adopted, where the weight of each unigram in the Unigram VSM reflects the relative importance of this word for a specific design pattern description within the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

corpus. We adopted TF*IDF weighting scheme [7] as it is one of the widely used schemes in the field of information retrieval. 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 corpus (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 (2) as follows:

$$\mathrm{TF} * \mathrm{IDF}(\mathrm{t}, \mathrm{d}, \mathrm{D}) = \mathrm{TF}(\mathrm{t}, \mathrm{d}) \times \mathrm{IDF}(\mathrm{t}, \mathrm{D}) \quad (2)$$

Where, $IDF(t,D) = 1/DF(t,D)$

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.

However, TF*IDF computed using equation (2) does not take the document length into consideration. Also, TF value indicates that if a term occurs five times in a document, it is five times more 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 (3) is used to compute TF*IDF.

$$TF * IDF(t,d,D) = Sqrt\big(TF(t,d)\big) * IDF(t,D) * 1/Sqrt(length)) \quad (3)$$

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

The document length could be disregarded in equation 3, as each of the design pattern descriptions and the problem scenarios are short documents.

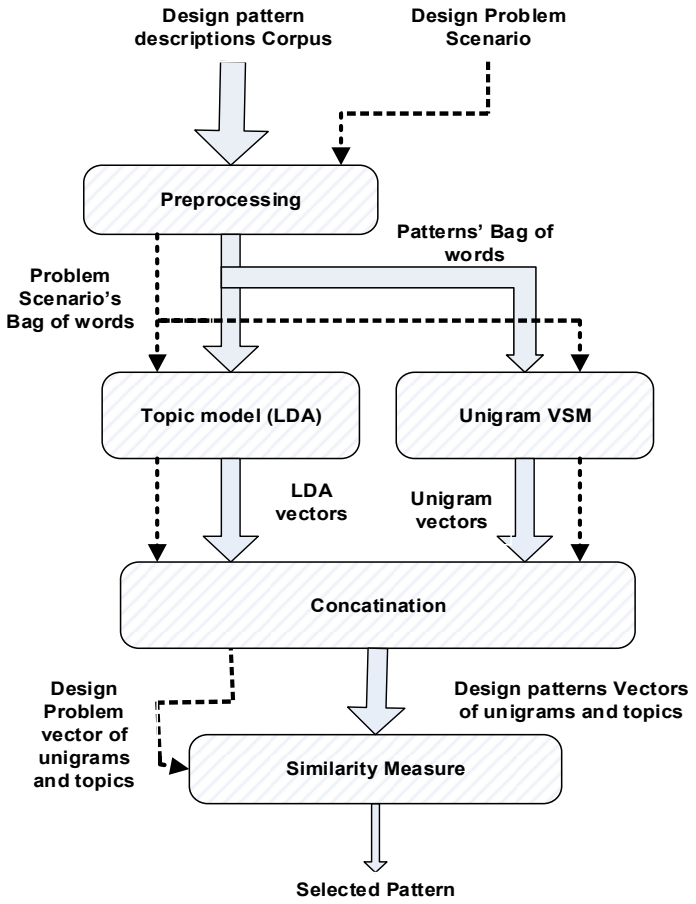


Figure 1. Proposed framework for automatic selection of design pattern

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 2. Preprocessing stages of each design pattern description and problem scenario

3.3. Topic Model

Topic model Latent Dirichlet Allocation (LDA) is a probabilistic generative model that allows the discovering of topics in a collection of documents [17], [18]. Each topic includes a group of terms (N-grams) from the corpus that may occur together. The terms belong to the topic with different probabilities. In this paper we consider only 5 terms (per topic) with highest probabilities.

The distribution of each pattern description and problem scenario in the produced topics could also be obtained. This distribution is based on the appearance of the topic terms in the document of pattern or problem and is calculated by summing all terms' probabilities in each topic. Therefore, each document could be represented by a vector which includes the distribution of the document over the topic, which we called LDA vector. LDA vectors hold beneficial features to the semantic similarity between the documents. The number of topics and number of terms per topic are parameters which should be selected during the experiments.

In this work, LDA model is trained and topics are extracted from the corpus of design patterns' descriptions. Then, LDA VSM is created where each pattern is represented by a vector which includes the distribution of the topics over the description of this pattern. The size of the vectors is equal to the number of topics.

The topic distributions over the target design problem is generated (design problem LDA vector) using the previously trained LDA model. Furthermore, a vector of unigrams for the target problem is generated based on the previously built Unigram VSM. Finally, LDA vectors are concatenated with the Unigrams vectors for each design pattern and each design problem.

3.4. Similarity measure

The selected pattern for a given design problem is the pattern whose vector of unigrams and topics is the closest to the problem's vector. Cosine similarity (CS) is one of the popular measures in text mining as it measures the angle between the vectors. However, Cosine similarity is derived from Euclidian distance which is not effective in text mining applications (examine equations 4 and 5).

$$\mathrm{CS}(\mathrm{V},\mathrm{W}) = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{v_i w_i}}{\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{v_i^2}}\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w_i^2}}} \quad (4)$$

Where, V is the feature vector of one of one of the patterns, W is the feature vector of the given design problem and n is the size of the vectors.

$$d_{Euclid} = \sqrt{2 - 2\sum_{i=1}^{n} v_i w_i} \quad (5)$$

Zhu et al. [21] proposed another similarity measure for information retrieval, called Sqrt-Cosine similarity (SCS). SCS is derived from Hellinger distance which is meant to measure the distance between two probabilities. They conducted text clustering experiments and proved that using Sqrt-Cosine similarity resulted in better performance than using Cosine similarity. Hellinger distance and SCS are given by equations 6 and 7. Sohangir and Wang [22] proposed a modified version of SCS and called Improved Sqrt-Cosine similarity (ISCS) as given by equation 8. They conducted experiments to compare the impact of using CS, SCS, and ISCS on the performance of each of the text classification and text clustering techniques. It was found that ISCS is the superior similarity measure, than CS then SCS. In this work we used both of Hellinger distance and ISCS.

$$SCS(v,w) = \frac{\sum_{i=1}^{n} \sqrt{v_i w_i}}{(\sum_{i=1}^{n} v_i)(\sum_{i=1}^{n} w_i)} \quad (6)$$

$$d_{Hellinger} = \sqrt{2 - 2\sum_{i=1}^{n} \sqrt{v_i w_i}} \quad (7)$$

$$\mathrm{ISCS}(\mathrm{v},\mathrm{w}) = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} \sqrt{\mathrm{v_i w_i}}}{\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{v_i}}\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w_i}}} \quad (8)$$

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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Case#1: Select the K[th] pattern V[k] with the highest value of ISCS (V[k] ,W).

Case#2: Select the K[th] pattern V[k] which satisfy equation 9.

$$\text{ISCS}(V^k, W) > \theta \quad (9)$$

Where, θ is a threshold value for the similarity

Case#3: Select the K[th] pattern V[k] which satisfy equation 10.

$$|ISCS_{max} - \text{ISCS}(Vk, W)| \leq \theta \quad (10)$$

Where, $ISCS_{max}$ is the maximum value of similarity between any design pattern and the given problem scenario, and θ is a threshold.

We adopted Case #1 in this work.

3.5. Evaluation metric

Pattern recommendation problem can be treated as a classification problem. Where the number of classes is the number of design patterns included in the corpus of design pattern descriptions (23 patterns in this paper). So we used micro-average Precision metric to assess the proposed methodology. It is evaluated by summing the individual true positive rates TP_i and individual false positive rates FP_i of all the design patterns (classes). It is given by equation (11) as follows:

$$Micro - Average\ Precision = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FP_i} \quad (11)$$

Where, m is number of design patterns, $\sum_{i=1}^{m} TP_i$ is the number of correctly recommended design patterns. While, $\sum_{i=1}^{m} FP_i$ is the number of incorrectly recommended patterns. Macro-average precision could also be used but we selected micro-average level due to the imbalance in the testing data used in the evaluation (number of problem queries are not equal across the design patterns)

4. EXPERIMENTS AND RESULTS

To evaluate the effectiveness of our approach, two corpus were created. One of which includes the textual descriptions of the GOF design patterns. Each pattern document includes the intent and applicability, in addition to part of the solution domain (participants, and collaborators). GOF book in addition to Wikipedia.com were used to prepare a rich description document to each pattern that includes the pattern distinctive words. The other corpus includes 29 real design problem scenarios collected from various sources including various design patterns books and Wikipedia.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. Seven samples of these design problems were defined as follows:

Problem #1: The Company class is the main class that encapsulates several important features related to the system as a whole. We need to ensure that only one instance of this class is present.

Problem #2: The system should have only one printer spooler although it can identify many printers.

Problem#3: The system has an interface named "Media Player". This interface is implemented by a concrete class Audio Player. Audio Player 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.

Problem#4: It is required to use an existing user interface toolkit in developing software applications that work on different platforms. Hence, it is important to include a portable window abstraction in the toolkit such that the user can create a window without being committed to a certain implementation as the window implementation is related to the application platform.

Problem#5: The system approves purchasing requests. There are four approval authorities and the selection of the approval authority depends on the purchase amount. If the amount of the purchase is higher than one million dollar, the owner is the one who approves. However, if it ranges from 500k to less than one million, the CEO is the one who approves and if it ranges from 25k to less than 500k, the head of the department is the one who approves. Finally, if the purchase is less than 25k, the vice is the one who approves. The system needs to be flexible such that the approval authority for each amount of money can change at run time.

Problem #6: 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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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. Another menu style prints a letter in front of each string (e.g., A, B, and so on) and lets the user enter a letter to make a choice. Still another menu style prints each string out, and lets the user type in the first few characters of the string to make that 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. It should be extremely easy for us to modify the program so that it uses a different menu style whenever needed.

Problem # 7: The developer of a game desires the player to be able to pick up and drop off a variety of elements which exist in the environment of the game. Two types of these elements are bags and boxes, each of which may contain individual elements as well as other bags and boxes.

Pre-processing was performed using the natural language toolkit NLTK [23]. While, Genism [24] was used for training the LDA topic model. Experiments were tried with the number of topics set equal to 5, 8, 10 and 20. Number of terms per topic was set equal to the number of topics. The best precision obtained during experiments is about 72 % which occurred when setting the number of topics equal to 8 or 10. Table 2 depicts the sample of the topics generated. It shows the most important five terms in each topic and their probabilities. Table 3 depicts samples of the results. It shows for each problem scenario listed above, both the correct and the first three recommended design patterns using the proposed approach. It should be noted that although problem #2 is written briefly, the proposed approach was able to select the right pattern. According to table 3, a wrong pattern (Adapter) was selected for problem #6. But experts agreed that Adapter pattern could fit this problem scenario. All design problem scenarios of failed cases were reviewed and it was noted that either these cases do not include descriptive words of the pattern, or are not well written such that the recommended pattern could fit same as the case of problem #6.

For the purpose of assessing our proposed approach, two extra experiments were conducted. In the first experiment, both of the design patterns and the problem scenarios are represented using vectors of unigrams only. While in the second experiment, patterns and problems are represented using vectors of topics only. ISCS similarity measure was used in the first experiment while Hellinger distance was used in the second experiment. It was found that our proposed approach is superior to both of the VSM of unigrams approach and topics only approach in terms of micro-average precision as illustrated by table 6. Tables 4 and 5 list the first three selected patterns in case of using topics only and in case of using unigrams only. As shown by table 4, topic only based approach failed to recommend patterns for some problems and recommended wrong patterns for others. This is because each of the pattern descriptions and problem scenarios are short documents and topics do not hold enough information to differentiate between the patterns. As depicted by table 5, unigrams only approach failed to recommend the fit patterns for some problems like problem#2 which our proposed approach succeeded to recommend the correct pattern for. We found out that integrating topics and unigrams in representing the patterns and the problems improved the precision of the recommendation process and does not have a great influence on the size of the traditional unigram VSM.

5. LITERATURE REVIEW

The current research for automatic recommendation of the fit design pattern can be categorized into four approaches which are: UML based, question-answer model based, case based reasoning, anti-patterns based and finally text based approach. The following subsections review the previous research in each of these approaches.

5.1. UML based approach

Kim and Khwand [3], Kim and Shen [4] use class and collaboration diagrams of the analysis phase to select patterns for the design phase. To achieve this purpose they generate a meta-model for each design pattern from its UML diagrams.
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 versus Strategy patterns and Façade versus Adapter patterns. 2) This approach is not scalable due to the overhead resulting from generating the meta-models. In addition, the increase in the number of patterns causes an increase in the similarity between the meta-models).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table2. Sample of the topics generated

Topic ID	Terms per Topic and their probabilities
1	Object (0.058), operate (0.023), component (0.016), interface (0.015), responsible (0.015)
2	Subsystem (0.047), request (0.037), interface (0.029), object (0.028), class (0.027)
3	Object (0.023), strategy (0.022), algorithm (0.020), state (0.018) , client (0.017)
4	Request (0.034), operation (0.028), command (0.027) , state (0.019), receive (0.016)

Table 3. First three selected patterns and similarity values for each of the 7 design problems listed using the proposed approach (topics and unigrams)

ID	Correct Pattern	1st Selected Pattern	2nd Selected Pattern	3rd Selected Pattern
1	Singleton	Singleton (0.258)	Prototype (0.184)	Adapter (0.162)
2	Singleton	Singleton (0.186)	Visitor (0.067)	Strategy (0.062)
3	Adapter	Adapter (0.407)	Bridge (0.307)	Composite (0.216)
4	Bridge	Bridge (0.422)	Facade (0.230)	Template (0.166)
5	Chain of Responsibility	Chain O.R.(0.380)	Observer (0.191)	State (0.185)
6	Strategy	Adapter (0.15)	Visitor (0.139)	State (0.137)
7	Composite	Composite (0.011)	Bridge (0.007)	Visitor (0.005)

Table 4. First three selected patterns and similarity values for each of the 7 design problems listed using topics only approach.

Problem ID	Correct Pattern	1st Selected Pattern	2nd Selected Pattern	3rd Selected Pattern
1	Singleton	Singleton (0.277)	Decorator (0.245)	Visitor (0.234)
2	Singleton	Strategy (0.31)	Visitor (0.16)	Observer (0.008)
3	Adapter	Adapter (0.33)	Bridge (0.175)	Composite (0.124)
4	Bridge	Decorator (0.19)	Composite (0.177)	Adapter (0.131)
5	Chain of Responsibility	Failed	Failed	Failed
6	Strategy	Visitor& Strategy (0.015)	Singleton& State (0.011)	Command& Template (0.008)
7	Composite	Failed	Failed	Failed

Table 5. First three selected patterns and similarity values for each of the 7 design problems listed using unigrams only approach.

Problem ID	Correct Pattern	1st Selected Pattern	2nd Selected Pattern	3rd Selected Pattern
1	Singleton	Singleton (0.244)	Adapter (0.178)	Prototype (0.162)
2	Singleton	Strategy (0.37)	Visitor (0.015)	Prototype (0.014)
3	Adapter	Bridge (0.241)	Visitor (0.124)	Composite (0.122)
4	Bridge	Bridge (0.272)	Façade (0.084)	Template (0.076)
5	Chain of Responsibility	Chain (0.057)	Observer (0.03)	Command (0.025)
6	Strategy	Strategy (0.028)	Façade (0.025)	Observer (0.022)
7	Composite	Composite (0.204)	Decorator (0.184)	Visitor (0.128)

TABLE 6. Comparison between the precision of our proposed approach, Unigrams only approach and Topics only approach

Approach	Precision
Proposed (Topics and Unigrams)	72%
Topics only	36%
Unigrams only	60%

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.2. Question-Answer Model based Approach

This approach is a semi-automatic approach based on providing the user with a set of "yes" or "no" questions which the user answers with "yes", "no" or "don't know". Based on these answers, patterns are ranked and the pattern with the highest rank is the selected one. 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 conditions, where the second layer has sub-conditions; both of the conditions and sub-conditions are extracted from the pattern definitions. The model was evaluated by a total of six graduate students and two information technology professionals. The success ratio of the system reached 50%. Pavlie et al. [6] used the question-answers to build an ontology-based model for design patterns recommendations. AlSheikSalem and Qattous [25] investigated ten patterns of GOF catalogue and generated the questions that describe these patterns. They proposed to build an expert system based on these questions. They developed a prototype expert system for three patterns only and had four undergraduate students evaluate the system. The students mentioned that some questions were not easy to understand and the prototype could not provide any recommendations in many cases.

The main challenge that faces this approach is constructing the questions especially with the large number of patterns. Furthermore, the set of questions is usually biased towards the specifics of the design patterns themselves rather than the software design problem.

5.3. 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 of the problem and the solution to it (fit design pattern). Gomes et al. [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 based on Case Based Reasoning (CBR) and Formal Concept Analysis (FCA). FCA was used for forming an index lattice for the design pattern case base.

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.

5.4. Anti-patterns based approach

This approach is based on identifying anti-patterns at the design level [11] or at the code level [12] then recommending the suitable pattern. Smith and Plante [12] recommend patterns at the code-level, where patterns are recommended dynamically during the implementation phase. They identify anti-patterns using structural and behavioral matching in the code, and then the fit design patterns are recommended to overcome the identified anti-patterns. The motivation for this approach is that the pattern catalogues include information about the anti-patterns, in addition to the ambiguity exist with the textual problem scenarios.

However, design pattern recommendation during implementation phase is too late as the software has already been designed and should be changed.

5.5. Text based approach

This approach is based on matching the design problem textual description against design pattern textual descriptions [7], [8], [9], [10]. Hasheminejad and Jalili [9] proposed a two phase system. During the first phase, the category of the patterns the design problem belongs to is determined. Then, the recommended pattern is retrieved based on the Cosine similarity between the textual description of the problem and each of the patterns that belong to the apriori determined category. Therefore, they trained a set of classifiers to recognize the category. For example they trained three classifiers for GOF catalogue to differentiate between creational, behavioral and structural patterns. Sanyawong et al. [7] grouped GOF patterns based on their usage into five categories and trained five classifiers to differentiate between these categories. They used popular classification techniques: Naive Bayes, J48 and K-nearest neighbor (k-NN). The performances of Naïve Bayes based classifiers were the lowest except with one category. However, the core challenges of these two research studies were the need to train a large number of classifiers (one classifier for each category) and the need for an adequate dataset to achieve an acceptable classification accuracy. To overcome these challenges, Hussain et al. [10] proposed to use unsupervised learning technique (clustering using Fuzzy C-means) instead of classification (supervised

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

learning technique). Suresh [8] proposed a framework for design pattern recommendation that depends on two approaches which are: text retrieval and question-answer. In this framework, the problem is represented as a collection of unigrams and matched against the intent of each pattern. Then the intents of the top candidate design patterns are displayed for the designer to select the most suitable pattern. Nevertheless, they have partially implemented and tested their framework. In addition to measuring the similarity between the problem scenario and the pattern intent only will not result in a high recommendation accuracy.

However the work proposed in [7], [8], and [9] and [10] suffer from one main shortcoming which is representing each pattern and each problem as a vector of unigrams only and did not consider the semantic similarity between the problem scenario and the patterns. So the performances of their models will be very sensitive to the words used in describing the design problems. This is the reason we proposed to use topic model in this paper to enhance the features included in the vector space model to reduce the sensitivity of the system to the quality of the problem scenario description. Moreover, we used the Improved Sqrt-cosine similarity (ISCS) measure instead of the Cosine similarity. Using ISCS in addition to enhancing the feature vectors with the topics improved the accuracy of recommendation as illustrated in the experiments section.

6. CONCLUSION AND FUTURE WORK

This paper proposed a novel methodology that automatically recommends a design pattern to solve a given design problem scenario. For this purpose an LDA model was trained and topics were extracted from the textual description of the design patterns. LDA model is able to discover the hidden semantic in a text through relating words with similar meaning and differentiating between uses of words with multiple meanings. Then, A vector space model was constructed to represent each pattern and the target problem scenario by a vector of unigrams and topics. Improved Sqrt-cosine similarity measure was used to measure the similarity between the target problem and each pattern. The fit pattern is the closest one to the problem scenario. To test our approach we had to build two repositories, one for the design pattern descriptions and the other for the design problems as there is no benchmark dataset, researchers can use. The experimental results illustrated that our proposed approach is promising and outperforms the approaches based on unigrams features only or topics features only in terms of precision. However, the precision of the proposed approach is still influenced by two factors which are: Firstly, the existence of an efficient dataset to the design patterns descriptions. This dataset should include as much information as possible about the pattern. 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 recommending the right design pattern. Using a lexical database like WorldNet [26] can alleviate the influence of the second factor on the results.

Currently, we work on extending our datasets to include more catalogues of patterns which are patterns of concurrency and security. In addition to, integrating case based reasoning approach with our proposed approach to enhance the precision.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Pattern: Elements of Reusable Object-Oriented Software." Addison-Wesley, 1995.

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

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

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

[5] F. Palma, H. Farzin, Y.-G. Gu´eh ,eneuc, and N. Moha, "Recommendation System for Design Patterns in Software Development: An DPR Overview," 3rd International Workshop on Recommendation Systems for Software Engineering, IEEE, 2012, pp. 1–5.

[6] L. Pavlic, V. Podgorelec, and M. Hericko, "A Question-based Design Pattern Advisement Approach,"Computer Science and Information Systems , vol. 11, no. 2, 2014, pp. 645–664.

[7] N. Sanyawong, E. Nantajeewarawat, "Design Pattern Recommendation: A Text Classification Approach". 6th International Conference on Information and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Communication Technology for Embedded Systems, IC-ICTES, 2015.

[8] S. Suresh, M. Naidu, S. A. Kiran, and P. Tathawade, "Design Pattern Recommendation System: a Methodology, Data Model and Algorithms," in Proceedings of the International Conference on Computational Techniques and Artificial Intelligence (ICCTAI) , 2011.

[9] S. M. H. Hasheminejad, S. Jalili, "Design patterns selection: An automatic two-phase method", Journal of systems and software, elsevier, 2012, pp. 408-424.

[10] S. Hussain, , J. Keung, A. A. Khan, "Software design patterns classification and selection using text categorization approach" Applied soft computing, 2017, pp. 225-244.

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

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

[13] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. L. Ferreira, and C. Bento, "Using CBR for Automation of Software Design Patterns," Advances in Case-Based Reasoning, Springer Berlin Heidelberg , vol.2416, 2002, pp. 534–548.

[14] W. Muangon and S. Intakosum, "Case-based Reasoning for Design Patterns Searching System," International Journal of Computer Applications , vol. 70, no. 26, 2013, pp. 16–24.

[15] K. S. Jones and P. Willet, "Readings in Information Retrieval," San Francisco: Morgan Kaufmann, 1997.

[16] M. Melucci,"Vector-space model", Encyclopedia on Database Systems,2009, pp. 3259-3263.

[17] C. Chemudugunta, P.S.M. Steyvers, "Modeling general and specific aspects of documents with a probabilistic topic model", Proceedings of the 20th Annual Conference on Neural Information Processing Systems, 2007 .

[18] D.M. Blei, J.D. Lafferty, A correlated topic model of science. Ann. Appl. Stat.,2007, pp.17–35.

[19] T. Zhang, J. Chen, G. Yang, B. Lee, X. Luo , "Towards more accurate severity prediction and fixer recommendation of software bugs", Journal of systems and software, elsevier,2016.

[20] W. Zimmer, "Relationships Between Design Patterns." In J. O. Coplien and D. C. Schmidt (eds.), Pattern Languages of Program Design. Reading, MA: Addison-Wesley, 1995, pp. 345-364.

[21] S. Zhu , Lizhao Liu, Yan Wang, Information Retrieval using Hellinger Distance and Sqrt-cos Similarity, The 7th International Conference on Computer Science & Education (ICCSE 2012), Melbourne, Australia, July 14-17, 2012.

[22] S. Sohangir and D. Wang, "Improved sqrt-cosine similarity measurement", Journal of Big Data , Springer, 2017.

[23] NTLK http://www.nltk.org

[24] Gensim: https://radimrehurek.com/gensim/

[25] O. AlSheikSalem and H. Qattous, "An Expert System for Design Patterns Recognition", IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.1, January 2017.

[26] Wordnet: http://www.nltk.org/howto/wordnet.html