A New Approach for Interactive Design Pattern Recommendation

Imène Issaoui, Nadia Bouassida, and Hanène Ben-Abdallah

Abstract—Design patterns offer design and expertise reuse which promotes fast development and high quality products. However, to acquire the benefits of design patterns, a designer must have a high expertise to determine those design patterns suitable for a given design context and situation.

In this paper, we present an approach for interactive design pattern recommendation that suggests patterns to help novice designers during the design process. Our approach uses the semantic information encoded both in the design patterns and modeled in the design. It uses a similarity matrix to compare semantically the words extracted from the design (class names, method names, etc.) drawn by the designer with the roles played by patterns’ participants. For a better contextual assistance, it interacts with the designer to get textual description of the context of his/her design fragment and formulates questions about the pattern intentions using the designer’s description.

Index Terms—Interactive recommendation, software reuse, design pattern, semantic similarity.

I. INTRODUCTION

Design patterns offer proven solutions to recurring design problems. However, the large amount of knowledge associated with each design pattern requires high expertise to identify the suitable pattern for a particular design situation/application. In fact, determining the applicability of a design pattern to a given problem relies heavily on the experience of the software developers [1]. Moreover, it is extremely difficult for novice developers, unfamiliar with design patterns, to find the pattern that fits a given situation.

Indeed, as recognized by Gamma et al. [2], “with more than 20 design patterns in the catalog to choose from, it might be hard to find the one that addresses a particular design problem, especially if the catalog is new and unfamiliar to you”. That is, choosing the right design pattern is recognized to be a subtle task since the number of proposed design patterns is continuously increasing.

Recognizing the need for assistance in pattern selection, several approaches propose to determine the potential similarities between the design and existing patterns ([3]-[6]). These approaches differ mainly in the techniques they consider: Guéhéneuc et al. [4] use a simple textual analysis of design patterns descriptions, Gomes et al. [3] use Case-Based Reasoning, Palma et al. [5] use Goal-Question-Metric, Hasheminejad et al. [6] use a text classification approach. Most of these approaches do not combine the structural aspect of the design provided by the designer and the semantic aspect expressed by his/her intention. Relying solely either on the structural aspect or the semantic aspect to recommend a design pattern, in general, produces imprecise results.

To overcome these difficulties, we propose in this paper an interactive, semi-automatic approach for the suggestion of design patterns that better fit the designer’s design fragment (structure) and intention (semantics). The proposed method adopts a two-phase approach. In the first phase, it computes a semantic similarity matrix that assesses the resemblance between the drawn design and the patterns. The matrix lines are keywords characterizing the pattern and its columns are the design terms.

In the second phase which is interactive, our approach asks some questions to the designer. These questions use the design elements to reformulate the intentions of the design patterns identified in the first phase. Depending on the designer’s answers, the recommendation system refines the list of design patterns that are susceptible to be recommended. This second phase can be repeated until the appropriate design pattern is identified for the designer.

The rest of this paper is organized as follows: in Section II, we present an overview of design patterns recommendation approaches. In Section III, we describe our recommendation approach which relies on the semantic similarity matrix and on questions about the design context and intentions. In Section IV, we illustrate our method through a case study illustrating the approach. Section V presents the Pattern Recommender toolset.

II. WORKS ON DESIGN PATTERNS RECOMMENDATION

Several researchers proposed recommendation systems for design patterns. These works differ in the pattern/design's aspect they rely on to propose candidate patterns: some rely on the pattern documentation (representing the intention of the design pattern), others analyze the pattern/design structure, yet others examine the semantics represented through the names of the pattern/design elements. Consequently, each of the existing works on patterns recommendation adopts a different technique and operates with a different degree of interaction with the designer.

Guéhéneuc et al. [4] rely on the textual documentation associated with design patterns. They propose a simple recommendation system that helps users to choose among the 23 design patterns of the GoF [2]. The proposed recommendation system uses a set of keywords extracted...
through an analysis of the descriptive texts associated with the patterns. This set of keywords is presented to the designer who, depending on his/her choices, will receive recommendations. One inconvenience of this approach is that the user may not find an exact solution to his/her problem, because the system is limited to the 23 GoF design patterns. In addition, this approach does not allow designers to enter their own query, instead they are limited to selecting words from those proposed by the system. Because the proposed list stems from keywords of the design pattern descriptions, it is generic (i.e., independent of any particular application domain). This genericity of the key words makes it difficult for the designer to map them to his/her application domain. This disadvantage may make the recommendation of very little use in case of complex design problems.

Also relying on the pattern textual documentation, Suresh et al. [7] propose a design pattern recommendation system that tries to offer the designers a high degree of freedom in expressing their queries. In this system, the designer query is a collection of words describing a problem scenario. The system parses and analyzes the query words to find patterns whose intents matches the query words. The candidate pattern intents are then displayed, and the designer has to choose the ones that better fit his own intent. To further narrow down the candidate patterns, the system interacts with the designer through a set of questions. Depending on the designer's answers which the system scores, the system finally offers comments to the designer guiding him/her in identifying the most appropriate design pattern: strongly recommended, or satisfactorily recommended or not recommended. If no candidate pattern is found, the system looks for a similar query from the query history database; if a similar query is found, then suggests the corresponding design pattern. The main drawback of this approach is that the system supports the recommendation of GoF patterns only. To extend it to other categories of patterns, one needs a method to formulate the appropriate questions; such a method is not presented in this work.

Palma et al. [5] propose a Design Pattern Recommender (DPR) that suggests design patterns using a simple Goal-Question-Metric (GQM) approach. DPR is based on an expert system and operates in four steps:

1) The system first identifies the circumstances in which patterns can be applied in order to construct the knowledge base. Knowledge is extracted thanks to a study of the individual patterns, (properties, applicability, intents, examples, etc.). As a result, various trees are obtained.

2) It then refines the circumstances of each pattern application with sub-conditions. Consequently, various trees are obtained.

3) In the third step, the system formulates questions to ask the designers tree representation of knowledge about obtained patterns. Based on the answers to these questions, the patterns receive weights.

4) Finally, the system formulates a Goal-Question-Metric (GQM) model with the defined questions. In the formulated GQM model, the pattern names are the goals. In the question levels, 2-layered questions are proposed, the first layer represents conditions, followed by the sub-conditions in the second layer.

DPR was evaluated by six graduate students and two IT professionals. The reported evaluation results show a success certainty ratio of 50%.

Also with the objective of helping novice designers, Diaz et al. [8] present a recommendation system integrated in a visual environment for pattern-based design, named VEISIG. The recommendation module is based on collaborative filtering. The knowledge base is made up of the pattern language and the solutions formulated by expert users. The pattern language items and structure gather explicit expert design knowledge, while expert solutions gather implicit expert design knowledge. VEISIG provides users with a visual representation of the pattern language using design goals. Patterns are tied to the goals they satisfy. Goals are organized in six design views to facilitate their exploration by the users: structure, navigation, presentation, personalization and security. Once the user selects the goals, he/she can ask for recommendations. The system then starts by rating all the patterns that are not in the initial user selection. The rating of a pattern is obtained using many factors, e.g. the number of occurrences where the pattern appears in combination with any of the patterns in the initial set. If a pattern is categorized in the same design view as patterns in the initial set, its rating is increased. Once all the ratings are obtained, the recommendation algorithm chooses the pattern with the highest rating. One drawback of this approach is that the recommendation process is nontransparent to the users.

Gomes et al. [3] present an approach that automates the design pattern selection and application using Case-Based Reasoning and WordNet [9]. The proposed approach stores situations (cases) where the design patterns were applied. These usage cases can be reused in similar situations to guide the design pattern selection and application. This approach was implemented and tested in a CASE tool named REBUILDER. As argued by the authors, the main shortcoming of this approach is that the system performance depends on the quality and diversity of the case library.

Hasheminejad et al. [6] propose a two-phase method to select the appropriate design pattern. The proposed method is based on a text classification approach that takes as an input design patterns. Each one is annotated with a specific label called Design Pattern Class that was manually pre-determined by experts; these labelled design patterns represent the problem definition. In other words, the input of the proposed method is the problem definitions of some design patterns and their labels. The proposed method proceeds in two steps. The first step consists in learning patterns and the second step consists in retrieving the adequate pattern to a certain situation described textually. In the first step, a pre-processing of the pattern problem definitions is performed, then, for each design patterns cluster, one classifier is learned. In the second step, a candidate design pattern cluster similar to a given design problem is determined based on the text classification approach. Finally, the most similar design pattern(s) from the candidate design pattern cluster is suggested to the developer.

In summary, independently of the technique adapted, one common source of performance limitation of the above reviewed propositions is due to the fact that they examine one
aspect of the pattern and/or design. Furthermore, we believe that it is important to offer a high degree of freedom to the designers in entering their queries to the recommendation system. It is equally important to interact with them using their particular application elements. These are the main motivations behind our method which we present in the following section.

III. DESIGN PATTERN RECOMMENDATION APPROACH

We propose a two-phase approach for an interactive design pattern recommendation. Our method uses the semantic information encoded both in the design patterns and modeled in the design.

In the first phase, our method computes a semantic similarity matrix, $GSSMatrix$, that approximates the resemblance between the design drawn by the user and any set of design patterns. The lines of $GSSMatrix$ are the keywords of the pattern and the columns are the terms of the design. We obtain different resemblance matrices and we choose those matrices with maximum scores of $GSSMatrix$. For example, given two patterns $P_1$ and $P_2$, to decide upon which pattern better corresponds to a design $D$, we compute the sum of the $GSSMatrix$ scores for all the matched pattern classes in $P_1$ and $P_2$; the pattern with the maximum sum is the one that better instantiates the pattern.

In the second phase, our method interacts with the designer through a set of questions. To facilitate their comprehension and thereafter increase the success of the recommendation process, these questions use design elements to reformulate the intentions of the design patterns identified in the first phase. More specifically, the classes that have been identified as participating in a recommended pattern are used to formulate the questions. For example, the designer can be asked: "Do you want to compose the figure into tree structures to represent part-whole hierarchies?" Depending on the designer answers, the recommendation system refines the list of design patterns that are susceptible to be recommended. This second phase can be repeated until the appropriate design pattern is identified for the designer.

A. Semantic Similarity Matrix

The (global) semantic similarity matrix, $GSSMatrix$, estimates the resemblance between the design drawn by the user and each pattern the system has in its knowledge base. It should be noted that the calculation of $GSSMatrix$ is performed only on the design fragment entered by the user in order to manage the exponential temporal complexity of the pattern recommendation.

The calculation of the semantic similarity matrix is composed of two parts: class names resemblance followed by method name resemblance. The first step relies entirely on class name comparison, whereas the second step relies on method name comparison.

1) ClassNotFoundException: $CSSMatrix$

The class names resemblance between a pattern and a design starts by calculating the resemblance between each class of the pattern to all the classes of the chosen design fragment. This resemblance is measured through the class name semantic similarity matrix $CSSMatrix$ where the lines are the patterns classes’ names and the columns are the classes’ names of the design.

For this semantic comparison, we use a set of criteria which were inspired from [10] and which express the linguistic relationships between a design class and a pattern class.

- **Synonym $(E, K)$**: implies that the name $E$ is a synonym of the pattern keyword $K$. (e.g Synonym (Complex, Composite)).
- **InterDef $(E, K)$**: implies that the definitions of $E$ and $K$ given by WordNet [9] dictionary have common words. The common words list is obtained after eliminating the stop words such as ‘a’, ‘and’, ‘but’, ‘how’, ‘or’, and ‘what’.
- **DefContain $(E, K)$**: implies that the definition of the element $E$ contains a certain keyword $K$. (e.g. DefContain (Paper, Observation)).
- **NameInclude $(E, K)$**: implies that the name $E$ includes the name $K$. $E$ is a string extension of the keyword $K$. (e.g. NameInclude (CompositeFigure, Composite))

We collect in CSSMatrix the resemblance scores between the classes’ names of the design and the classes’ names of the pattern using ClassNameSim defined as follows:

$$
\text{ClassNameSim}(\text{NameDesignClass}, \text{NamePatternClass}) = \begin{cases} 
1 & \text{if Synonym(NameDesignClass, NamePatternClass)} \\
0.9 & \text{if NameInclude(NameDesignClass, NamePatternClass)} \\
0.8 & \text{if IsAKindOf(NameDesignClass, NamePatternClass)} \\
0.7 & \text{if DefContain(NameDesignClass, NamePatternClass)} \\
0.6 & \text{if InterDef(NameDesignClass, NamePatternClass)} \\
0 & \text{else}
\end{cases}
$$

Given a design class $A$ and a pattern class $B$, to calculate the similarity between $A$ and $B$, we use the WordNet dictionary to extract the lists of synonyms and the definitions of both classes. If we find several semantic relationships between $A$ and $B$, we use the relationship that has the highest coefficient.

Let us illustrate the calculation of the class name semantic similarity matrix of the design shown in Fig. 1 with the Decorator pattern. Using the WordNet dictionary we found two relations: Synonym(Component, Constituant) and Synonym(Decorator, InteriorDesigner). This gave us the
summing up the and method names similarities. In our running example, after Semantic Similarity Matrix) by adding the scores of the class relationships: A method combine both types of information in pattern. To determine the overall semantic resemblance, we information about the class and method names of the design classes’ names has been calculated, the pattern recommendation continues with the calculation of methods’ similarity matrix of the design shown in Fig. 1 with the class (column).

Let us illustrate the calculation of method semantic similarity matrix in the design shown in Fig. 1 with the Decorator pattern. Using WordNet [9] dictionary, we found the following three semantic relationships: InterDef (Draw, Operation), NameInclude (AddCurtain,Add), and NameInclude (AddColors,Add). Thus, we obtain the following MSSMatrix:

<table>
<thead>
<tr>
<th>Room</th>
<th>Interior</th>
<th>Simple</th>
<th>Curtain</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSSMatrix(D, Decor)</td>
<td>Component</td>
<td>Decorator</td>
<td>1 0 1 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Cr Comp</td>
<td>Cr Decorator A</td>
<td>0 1 0 1</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Cr Decorator B</td>
<td>0 1 0 1</td>
<td>1 1</td>
<td></td>
</tr>
</tbody>
</table>

2) MethodSemanticSimilarityMatrix: MSSMatrix

Once the similarity between pattern classes’ names and design classes’ names has been calculated, the pattern recommendation continues with the calculation of methods’ names similarity. Again, the resemblance is based on semantic similarity determined through semantic relationships: A method m is said to resemble another method m’, if the name of m is related to the name of m’ through a semantic relationships.

To determine the correspondences among the design and pattern methods, we use MSSMatrix (Method Semantic Similarity Matrix) which gives, for each design class (row), the percentage of resembling methods it has with each pattern class (column).

Let us illustrate the calculation of method semantic similarity matrix of the design shown in Fig. 1 with the Decorator pattern. Using WordNet [9] dictionary, we found the following three semantic relationships: InterDef (Draw, Operation), NameInclude (AddCurtain,Add), and NameInclude (AddColors,Add). Thus, we obtain the following MSSMatrix:

<table>
<thead>
<tr>
<th>Room</th>
<th>Interior</th>
<th>Simple</th>
<th>Curtain</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSSMatrix(D, Decor)</td>
<td>Component</td>
<td>Decorator</td>
<td>0.6 0.6 0.6 0.6 0.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cr Comp</td>
<td>Cr Decorator A</td>
<td>0.6 0.6 0.6 0.9 0.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cr Decorator B</td>
<td>0.6 0.6 0.6 0.9 0.75</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3) Global pattern resemblance

So far, we managed to derive semantic resemblance information about the class and method names of the design pattern. To determine the overall semantic resemblance, we combine both types of information in GSSMatrix (Global Semantic Similarity Matrix) by adding the scores of the class and method names similarities. In our running example, after summing up the CSSM(D,Decorator) and MSSMatrix(D,Decorator), we obtain the following matrix:

<table>
<thead>
<tr>
<th>Room</th>
<th>Interior</th>
<th>Simple</th>
<th>Curtain</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSSMatrix(D, Decor)</td>
<td>Component</td>
<td>Decorator</td>
<td>1.6 0.6 1.6 0.6 0.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cr Comp</td>
<td>Cr Decorator A</td>
<td>1.6 0.6 1.6 0.6 0.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cr Decorator B</td>
<td>1.6 0.6 0.6 1.9 1.75</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now, given this matrix, we can decide upon which correspondence better represents the pattern instantiation. For each pattern class, its corresponding design class is the one with the maximum resemblance score in GSSMatrix. Also, given two patterns P1 and P2, to decide upon which pattern P1 is better instantiated in a design, we compute the sum of the resemblance scores for all the matched designs in P1 and P2; the pattern with the maximum sum is the one that is better instantiated in the design.

B. Questions about the Intention

The intention cannot be directly interpreted; therefore, a preliminary step that consists in a pre-processing is performed. During the pre-processing phase, usually two actions are performed on the intention:

1) Filtering stop words (e.g., articles, conjunctions, prepositions, etc.) and stemming words. Stemming is used to reduce the number of words in the intention and try to build basic forms of words, e.g. strip the plural’s from nouns, the ‘ing’ from verbs, etc.

2) Re-expressing the intentions of the patterns list, provided by the first step, in terms of a questionnaire form. In other words, reformulate the questions about patterns intentions in terms of concepts taken from the design.

To illustrate this step, let us consider an intention expressed by a designer and try to compare it with the Strategy pattern’s intention and the State pattern’s intention.

<table>
<thead>
<tr>
<th>Strategy questionnaire</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you want to define a family of robots?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do you want to encapsulate each robot, and make them interchangeable?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do you want to vary the behavior independently from robots that use it?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State questionnaire</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you want to allow a robot to alter its behavior when its internal state changes?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IV. Example

To illustrate the two phases of our pattern recommendation approach, let us consider the design fragment D illustrated in
Fig. 2 and try to recommend the suitable pattern. First, the designer draws the design fragment D shown in Fig. 2. Second, the designer expresses the problem that has to be resolved by the pattern, which corresponds to the intention of the pattern.

In the first phase, we propose to compute the semantic similarity matrix $SSMatrix$ of the design D drawn by the user with the Composite design pattern. Once the class name similarity of the Composite pattern is calculated within the design D, the pattern recommendation continues with the calculation of method name similarity. For the design D and the Composite pattern, we get the following matrix:

$$CSSMatrix(D, \text{Composite}) = \begin{bmatrix}
0.9 & 0 & 0 & 0 \\
0 & 0.9 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}$$

After summing up the $CSSM(D,\text{Decorator})$ and $MSSMatrix(D,\text{Decorator})$, we obtain the following $GSSMatrix(D,\text{Decorator})$:

$$GSSMatrix(D,\text{Decorator}) = \begin{bmatrix}
1.9 & 0 & 0 & 0 \\
1 & 1.9 & 0 & 0 \\
1 & 0 & 0.6 & 0
\end{bmatrix}$$

Now given two patterns: composite and decorator, to decide upon which pattern better corresponds to a design D, we compute the sum of the $GSSMatrix$ scores for all the matched pattern classes in composite and pattern. Furthermore, the sum of the $GSSMatrix(D,\text{Composite})$ (6.4)
is greater than the sum of the $\text{GSSMatrix}(D,\text{Decorator})$ (3.8); thus this composite pattern is the most suitable.

V. TOOL SUPPORT

<table>
<thead>
<tr>
<th>Questionnaire</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you want to attach additional responsibilities to a shape dynamically?</td>
<td>□</td>
<td>■</td>
</tr>
<tr>
<td>Do you want to provide a flexible alternative to shape for extending functionality?</td>
<td>□</td>
<td>■</td>
</tr>
</tbody>
</table>

We developed a CASE support for our method, baptised PatternRecommender, which is integrated in the ArgoUML editor [11]. The main functionalities of PatternRecommender are essentially composed of three parts: 1) extraction of the design tree, 2) calculating the GSSMatrix, and 3) generation of a questionnaire and interpretation of the designer intention.

The design tree extraction consists in parsing the XMI file representing the UML design and finding the pertinent information (class name, operation name). This extraction is done thanks to an XSLT processor.

Once the XML tree is generated, the recommendation process begins. The design recommendation starts by calculating the GSSMatrix.

Fig. 3 shows a screen shot illustrating the CSSMatrix, MSSMatrix and GSSMatrix comparing a design fragment and the Adapter pattern.

After calculating the GSSMatrix, PatternRecommender formulates and asks the designers some questions to better understand his/her intention.

VI. CONCLUSIONS

In this paper, we presented a pattern recommendation approach. The approach assists the designer in choosing the appropriate design pattern. The recommendation of a particular design pattern is guided by a set of semantic criteria and by the retrieval of the pattern intention that better fits the situation. A particular emphasis is given to the semantic comparison criteria.

We are currently examining how to offer also assistance for patterns composition and recommend the composition of a selected pattern with others.

REFERENCES


I. Issaoui is preparing a doctorate degree in computer science at the Faculté des Sciences économiques et de Gestion de Sfax, Tunisia. She is a teaching assistant at the Institut préparatoire aux études d’ingénieurs of the University of Monastir, Tunisia.

N. Bouassida received a Ph.D in computer and information science from the University of Science of Tunis, Tunisia. She is currently an assistant professor at the Department of Computer Science of the Institut Supérieur d’Informatique et du Multimédia at the University of Sfax, Tunisia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory (Mir@el), University of Sfax. Her research interests include reuse techniques, such as design patterns, frameworks and software product lines.

H. Ben-Abdallah received a BS in computer science and BS in mathematics from the University of Minnesota, MPLS, MN, a MSE and Ph.D in computer and information science from the University of Pennsylvania, Philadelphia, PA. She worked at University of Sfax, Tunisia from 1997 until 2013. She is now a full professor at the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia. She is a member of the Multimedia, Information Systems and Advanced Computing Laboratory (Mir@el), University of Sfax. Her research interests include software design quality, reuse techniques in software and business process modelling.