Journal of Universal Computer Science, vol. 21, no. 13 (2015), 1746-1766
submitted: 4/5/15, accepted: 6/7/15, appeared: 28/12/15 © J.UCS

An Interactive Design Pattern Selection Method

Nadia Bouassida
(Mir@cl Laboratory,
Institut Supérieur d'Informatique et de Multimédia, Sfax University, Tunisia
nadbouassida@gmail.com)

Salma Jamoussi
(Mir@cl Laboratory,
Institut Supérieur d'Informatique et de Multimédia, Sfax University, Tunisia
salma.jamoussi@isimsf.rnu.tn)

Ahmed Msaed
(Institut Supérieur d'Informatique et de Multimédia, Sfax University, Tunisia
ahmed.msaed@hotmail.com)

Hanêne Ben-Abdallah
(King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia
HBenAbdallah@kau.edu.sa)

Abstract: Any inexperienced designer may not take advantage of design patterns due to their high level of abstraction, on the one hand, and their overwhelming number, on the other hand. In this paper, we propose a new approach that first retrieves and recommends a design pattern that is adequate to a designer's modeling context, it then helps them in its instantiation. Our approach learns past pattern reuse cases and it interacts with the designer through a questionnaire to ensure that the retrieved pattern corresponds to their needs and intentions. It uses the text mining technique Principal Component Analysis on past experiences of design pattern reuses; the choice of this technique was based on an experimental evaluation we conducted to determine the most adequate text representation and mining technique for our problem. In a final assistance step, after retrieving the most appropriate design pattern, our approach transforms the design situation at hand into the pattern constituting the solution.

Keywords: Design pattern reuse, recommendation, instantiation process, text representation
Categories: D.2.13, D.2.2, H.3.3

1 Introduction

Design patterns [Gamma, 95] provide for the development of higher quality software and a reduced development cost. To be able to reuse design patterns and to attain their intended benefits, designers are expected to have a good understanding and experience with them in order to be able to select the right pattern for their context; such high expertise is unfortunately not evident to acquire. In fact, a designer must overcome the difficulties relative to the understanding and then to the selection and application of the appropriate design patterns. Furthermore, face to an increasing number of proposed design patterns in various application domains, even for an experienced designer, the selection and instantiation/reuse of the design pattern

pertinent to their application is not a trivial task. The problem is further accentuated for an inexperienced designer who would need assistance both with the selection and instantiation steps.

To offer such assistance, several approaches for design pattern recommendation were proposed. Existing approaches can be classified into two categories depending on their inputs: either a UML diagram—often the class diagram, or a textual query. Among the UML diagram-based approaches, some works adopt Case Based Reasoning (CBR) to learn past design experiences (e.g., [Gomes, 02], [Muangon, 13]), and others define a set rules (e.g., [Kim, 07], [Hsueh, 07], [Kim, 08], [Bouassida, 12]). The approaches that start from a textual query adopt one of three techniques: ontologies ([Blomqvist, 08], [Pavlic, 09]), expert systems (e.g., [Moyihan, 06], [Kung, 03], [Palma, 12], [Pavlic, 14]), or text classification techniques (e.g., [Hashminejad, 12]). Starting the recommendation from a UML diagram is very convenient for designers. However, because multiple design choices/uncertainties exist during its elaboration, an UML diagram may not encode all semantic information that the designer has in mind; hence, relying solely on a UML diagram for the retrieval of the appropriate design pattern often produces imprecise recommendations. On the other hand, in the currently available text-based recommendation systems, designers enter their queries by choosing from a fixed list of keywords; these latter are extracted from the design pattern documentation. Being abstract, such keywords can be hard to map to the problem description of the application under development, which again limits the efficiency of the recommendation. We believe that a hybrid recommendation approach is more beneficial: the UML diagram is explored along with an interactive way of soliciting the designer's intention through a text-based questionnaire.

In this paper, we propose to combine CBR and questionnaires to determine the pattern that appropriately matches both the structural information encoded in the UML diagram and the semantic intention of the designer. We use CBR to retrieve a list of candidate patterns based on the UML diagram similarity to learned reuse cases. In addition, to filter the list of candidates and validate the choice of a design pattern, a questionnaire is formulated with the application's terms in order to identify the design pattern that best matches the designer intention.

To search for the adequate patterns in the past experiences repository, we opted for information retrieval techniques. To identify the most efficient information retrieval method, we experimented with three widely used methods: TF-IDF [Salton, 65], Latent Semantic Indexing (LSI) [Binkley, 11], and Principal Component Analysis (PCA) [Hotelling, 33]. Our experiment showed that PCA is the best choice for design pattern representation and search. Hence, PCA is implemented in our interactive design pattern assistance tool, named PRT (Pattern Recommender Tool), which allows the designer to draw a design fragment that illustrates the problem. PRT retrieves the list of candidate design patterns based on the cases already learned, it then interacts with the designer through a questionnaire to narrow down the list of candidates to the one appropriate to the context at hand. PRT uses linguistic resources to consider the case where the designer uses a different language such as French or different vocabularies from those of the design pattern description and past experiences. Finally, PRT instantiates the selected design pattern by transforming the

design problem into the pattern solution. This latter step is among the essential advantages of using the UML diagram-based cases.

The remainder of this paper is organized as follows: Section 2 overviews current works for design pattern recommendation, then it presents the basic principles of the three evaluated text mining techniques: PCA, LSI and TF-IDF. Section 3 presents our approach for pattern recommendation. Section 4 discusses the experimental results through which we decided to use PCA as the appropriate information retrieval technique for pattern recommendation based on past experiences. Section 5 describes the tool PRT. Section 6 summarizes the paper and outlines our future work.

2 Related work

2.1 Existing pattern recommendation approaches

Several works examined pattern recommendation and assistance with pattern selection (e.g., [Guéhéneuc, 07], [Gomes, 02], [Suresh, 11], [Diaz, 11], [Palma, 12], [Hashminejad, 12]). In their recommendation approach, Guéhéneuec et al. [Guéhéneuc, 07] use an analysis of descriptive texts of design patterns in order to extract the most important key words. These latter are used for the retrieval of a pattern by comparing them against the set of words selected by the designer. One disadvantage of this approach is that the designer ends up with multiple solutions to their problem. This is partially due to the fact that the approach does not allow designers to enter their own query words (they just select from the keywords); in addition, because the keywords originate from the abstract documentation of the design patterns, they are too abstract for the designer to choose a restricted set of keywords so as to target one design pattern. Furthermore, this abstract keyword-based approach cannot manage complex design problems. Finally, as presented, this approach does not offer any feedback to the designer, for instance to guide them in a pattern instantiation.

Gomes et al. [Gomes, 02] recommend patterns through a CBR technique that relies on past design experiences designed with UML diagrams. Cases are stored in a library and indexed using WordNet [Miller, 95]. The main drawback of this approach is that cases are described with class diagrams which may not be available for some patterns; e.g., security patterns have just textual descriptions. Moreover, as argued in the introduction, the class diagram may not encode all semantic information needed to apply a particular pattern like the pattern context and intention.

Also adopting a CBR approach, Muangon et al. [Muangon, 13] propose to adapt design patterns recommendation systems based on Formal Concept Analysis. FCA [Ganter, 96] is a mathematical approach that uses a data analysis method based on a concept lattice. The aim of FCA is to organize indexes in order to retrieve more appropriate design patterns. By using only the class diagrams, this approach has the same drawbacks as the approach of [Gomes, 02].

Hashminejad et al. [Hashminejad, 12] propose to recommend design patterns using an automatic two phase approach based on a text classification technique. The first phase learns one classifier for each design pattern class. The second phase uses a text classification method to retrieve a design pattern class similar to the textual query describing the design problem. This approach relies only on a textual description, and

does not allow the designer to propose a model of the problem; consequently, the approach cannot transform the pattern problem in a model of a pattern solution. In addition, the quality of the description of the design problem has a decisive impact on the overall quality of the proposed method.

Palma et al. [Palma, 12] present an expert system named DPR (Design Pattern Recommender) that suggests patterns based on a simple Goal-Question metric (GQM). GQM is a ranking based selection approach. DPR operates in four steps: first it performs knowledge acquisition through the identification of circumstances in which patterns can be applied (pattern intent), second it refines the circumstances with sub-conditions (pattern applicability); third it formulates questions to the designers; finally, it formulates GQM model with the defined questions. This system can be beneficial for patterns that are described textually like security patterns. This approach is very interesting since it relies on questions, however it does not learn from past experiences. In addition, it does not transform the solution by adapting the pattern problem to the application problem.

All of the aforementioned approaches present different views for pattern recommendation; some of them focus only on learning cases, others on keywords retrieval. Based on our analysis of the results of these approaches, we noticed that it would be interesting to have an approach combining efficiently the Case Based Reasoning technique with a text mining technique and that filters the results thanks to a questionnaire about the design intent and context.

2.2 Basic principles of TF-IDF

TF-IDF (term frequency-inverse document frequency) [Salton, 1965] combines two criteria: TF and IDF. TF is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document. IDF: Inverse Document Frequency gives a more representative weight to each term according to its relative rarity in the set of documents. Thus, it weights down the terms appearing in all documents while scaling up the rare ones. TF-IDF assigns a weight to a term *j* in a document *i* as follows:

$$w_{ij} = tf_{ij} * idf_{ij} = tf * \log(m/(D(j)))$$

where:

- w_{ij} is the weight of the word *j* regarding the document *i*
- $tf_{i,j}$ is the frequency of the word *j* in the document *i*
- *m* is the total number of documents in the collection; and
- *D(j)* is the number of documents where the word *j* occurs.

Many variants of the TF-IDF weighting exist in the literature. The most known case is the normalized TF which is used to account for documents of different lengths. In our case, we do not consider the pattern length (in terms of words) because of the relatively reduced number of terms in each design pattern. For this reason, we do not consider this issue in our work and we use the basic TF-IDF form aforementioned.

When a query is entered, its numerical representation is created as a document using the TF-IDF technique. This representation is then compared with the cosine-distance to the other documents. The calculus of this similarity distance is as follows:

$$(\widehat{\vec{dl}, \vec{q}}) \cos(d_i, q) = \frac{\sum_{tj \in T} w_{ij}\, w_{qj}}{\sqrt{\sum_{tj \in T} w_{qj}^2 \sum_{tj \in T} w_{ij}^2}} \in (0,1)$$

where:

- d_i is the document i
- q is the query (corresponding to the pattern class names candidates);
- $cos(d_i,q)$ is the angle between the vectors d_i and q;
- w_{ij} is the weight of the term t_j in d_i;
- w_{qj} is the weight of the term t_j in q; and
- T is the set of terms contained in the documents.

2.3 Basic principles of LSI

LSI (Latent Semantic Indexing) [Dumais, 93] assumes that words that always appear together are related and that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts. The context of a word is defined as the set of words that appear together with it.

After the construction of the occurrence matrix (the term-document matrix), LSI uses singular value decomposition (SVD) and finds a low-rank approximation to the initial matrix. In that case, the initial matrix A is decomposed as the product of three other matrices as: $A=TSD^T$. The matrices obtained during the singular value decomposition are then reduced to a given number k of dimensions to result in the truncated matrices T_k, D_k and S_k of the latent semantic space (see figure1).

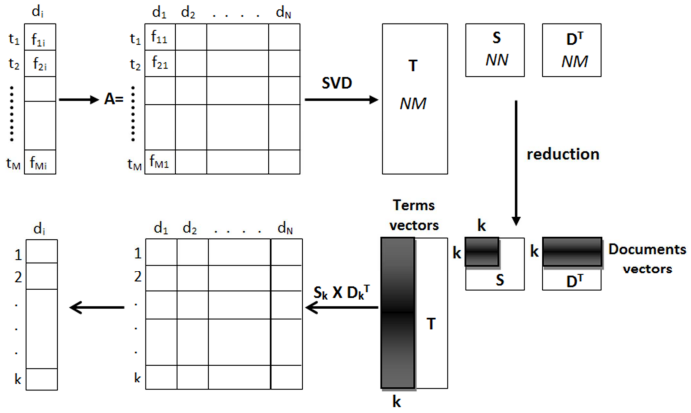

Figure 1: The process for calculating LSI

Figure 1 illustrates the process for calculating the LSI. When the matrices T_k,S_k and D_k are multiplied, they generate a new matrix $A_k = T_K S_K D_K{}^T$ which is considered as the approximation of A with a reduced number of dimensions. A_k represents then the information in this new subspace of K dimensions.

2.4 Basic principles of PCA

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. It reduces the number of variables to significant factors. These factors define a new subspace where similarity between individuals is better exposed.

The new factors are the eigenvectors, ordered by eigenvalues in descending order, of the data covariance matrix. In other words, they are the main dispersion axis of the data, in decreasing order of importance; the corresponding eigenvalues indicate the share of variance expressed by each axis. Principal components are linear combinations of the original variables expressed as follow:

$$PC(i)=w_{1i}\,.\,x_1 + w_{2i}\,.\,x_2 + w_{3i}\,.\,x_3 + \ldots. + w_{qi}\,.\,x_q$$

where w_{1i} is the weight of the first variable in the ith principal component and q is the number of the original variables.

The principal axes thus calculated allow, in the same time, a reduction of data and an easier interpretation in the treated domain. Indeed, as the new dimensions are often very significant, only k (k<q) dimensions will be retained to better describe the data.

Mathematically, PCA seeks the axes (directions) of the data points where the variance is maximal, i.e. calculating the square matrix of covariance of the input data and its values and eigenvectors. To do this, we must first calculate the average $\bar{p}$ of our data points with the following function:

$$\bar{p} = \frac{1}{n}\sum_{k=1}^{n} p_k \qquad (1)$$

where n is the number of data points. The covariance matrix C of our input data is:

$$C = \frac{1}{n-1}\sum_{k=1}^{n}(p_k - \bar{p})(p_k - \bar{p})^T \qquad (2)$$

$$= \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1q}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \cdots & \sigma_{2q}^2 \\ \vdots & & & \\ \sigma_{q1}^2 & \sigma_{q2}^2 & \cdots & \sigma_{qq}^2 \end{pmatrix} \qquad (3)$$

where σ_{ij}^2 represents the covariance between variables i and j in our input data.

Finding the principal components of our data implies determining the eigenvalues and eigenvectors of the matrix C. The eigenvectors of C define in R^n the orientations of the principal components of our input data when the origin of the vector space is moved to $\bar{p}$. The eigenvalues represent the significance of each of these components. They correspond to the variances of the data when projected on each of these directions. Let Z be the matrix whose columns are the q eigenvectors of C (z_1, z_2, ..., z_q,). Subsequently, the principal components analysis consists of choosing the first k eigenvectors associated with the biggest eigenvalues, i.e., those that maximize the variance. To reduce the size of the space of representation of our data, it suffices to construct the matrix W as follows:

$$W = \left[z_1, z_2, \ldots z_k,\right];\ k < q \qquad (4)$$

3 Our design pattern selection approach

Our design pattern selection approach (illustrated in Figure 2) operates according to the following steps:

- Pattern cases repository construction: this step starts from a set of past experiences containing design pattern instances and constructs a repository of pattern cases that will be used in the recommendation steps.
- Design query formulation: the user inputs a design fragment in the form of a UML diagram. Then, the terms in the diagram are extracted automatically and used as the query in the following step.
- Pattern retrieval based on learning: to handle this task, first the cases (pattern participant's names, their roles and the corresponding pattern name) are entered in the database and a model to describe each pattern class is learnt. Henceforth, pattern retrieval consists in searching and retrieving cases from the case library. This is done namely by means of an information retrieval method such as TF-IDF, LSI and PCA. Cases relevant to the target design are identified by applying a similarity metric that ranks the learnt cases.
- Questionnaire: asks the designer many questions to validate one of the retrieved patterns as appropriate to the design problem.
- Design pattern instantiation: transforms the design to include the validated, recommended pattern.
- Case retaining: learns the validated, recommended pattern. This step enriches the pattern cases base with new experiences managed through our approach.

3.1 Pattern cases repository construction

The pattern cases database was composed of different design patterns extracted from well proven case studies: JHotDraw [Gamma, 07], JUnit [JUnit, 07], Jrefactory [JRefactory, 07], Lexi, etc. Other design patterns were also extracted from proven cases in the literature [Gamma, 95]. We next present the pattern cases repository construction steps and the resulting learning pattern base.

3.1.1 Past pattern experiences pre-processing

The pre-processing step starts from past design experiences where various design patterns were instantiated and it is applied to all patterns in the case base. It uses classical Natural Language Processing operations to "clean" the names used in the past design experiences so as to facilitate the creation of the learned pattern cases repository; this latter is used in the pattern retrieval step.

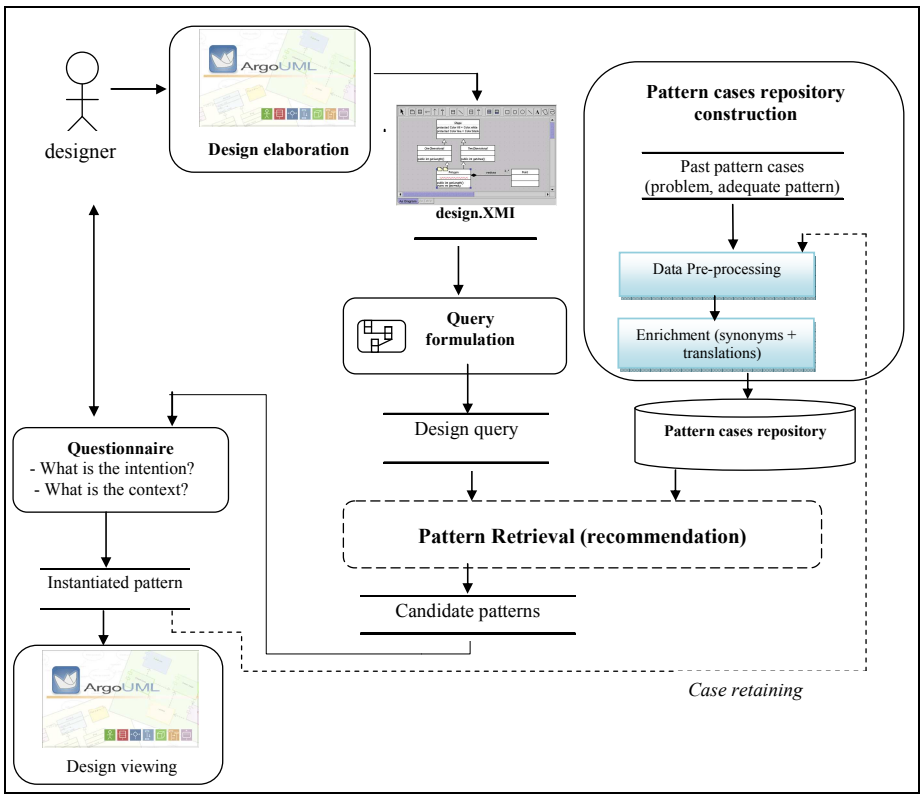

Figure 2: Steps of our recommendation approach as implemented in PRT

More specifically, the pre-processing step extracts the words used to name the elements of the past design experiences (i.e., the names of patterns, classes, roles, methods …), and it applies on them the following three operations to extract the linguistic units:

- Stop words elimination: For each element name, its composing words are extracted based on spatial markers like spaces, capital letters and special characters. Afterwards, stop words like “to”, “of”, “the”, etc. are removed since they do not bring any additional signification. The retained words are basic and carry meaningful information.
- Lemmatization: The objective of this morphological operation is to group together the different inflected forms of a "basic" word so they can be analyzed as a single word. Lemmatization, which is a morphological process, transforms each word to its basic form also called lemma. Consequently, different forms of a basic word that may have similar meaning are grouped together and handled as one word (the lemma). For example, in English, the verb 'to walk' may appear as 'walk', 'walked', 'walks', 'walking'; the form 'walk' is the lemma of all these words.

- Filtering to retain only the most common terms: We eliminated those terms with a frequency below a fixed threshold; in our case the threshold was set at 5. The main hypothesis is that infrequent terms are less likely to appear in the test set also because of their random selection.

Note that the literature of Natural Language Processing offers several tools for handling the above morphological operations. In our work, we have chosen to use the "TreeTagger" tool [Schmidt, 15]. Thanks to this pre-processing phase, our learning lexicon is reduced from 698 to 665 words for the same number of patterns, and then it is further reduced to 551 words after eliminating stop words and filtering.

3.1.2 Enrichment step

The enrichment step aims to construct a description of the learned design patterns that offers a large vocabulary in more than one language. In this step, we try to describe a design pattern case by a maximum amount of information and features by:

- expanding the list of terms used to name the components of design patterns in past experiences by adding their synonyms; and
- providing for different design languages such us French.

The main idea is to use linguistic resources to find synonyms and appropriate translations. In our case, we make use of the WordNet ontology to retrieve synonyms of words occurring in our patterns [Miller, 95]. For the French translations we use a bilingual dictionary. Actually, there are plenty of free bilingual dictionaries available on the net and that we can use in such case. In our work, we chose the French-English Collins Dictionary.

Overall, the enrichment step has remarkably changed our vocabulary, which passed from a size S=551 to S=2125. This means that the vocabulary was expanded for almost four times. Each word was replaced by average of 3.85 words, including synonyms and translations. For each word we have a mean of 2.6 synonyms and 1.25 translations with a standard deviation of 1.5 in the first case and of 0.7 in the second one.

3.1.3 The learning pattern base

In our current pattern cases repository, we actually used only the class names (pattern participants), roles and pattern names. In future works we will include also attributes and methods names. Note that, we did not make use of the structure of the class diagram (relations between classes: aggregation, inheritance …) because we suppose that the user is inexperienced with patterns and consequently the relations drawn are probably incomplete/uncertain and they will be changed after the recommendation in the adaptation step. However, the sequence diagrams that represent the interactions could in the future be considered to compute similarities between the design drawn by the user and the sequence diagrams corresponding to the patterns to recommend.

Among the 23 design patterns of the Gang of Four (GoF) [Gamma, 95], our pattern cases repository contains the following 18 patterns:

- Creation patterns: Abstract factory, Builder, Factory and Singleton.
- Structural patterns: Adapter, Bridge, Composite, Decorator and Proxy.

- Behavioral patterns: Command, Mediator, Observer, State, Strategy, Template, Visitor, Iterator and Chain of Responsibility.

The remaining patterns will be addressed in future work since we estimate that the different categories are sufficiently represented and we believe that even if we add the remaining 5 patterns, the trend of the results would not change. An extract of the data used for learning is presented in Figure 3.

Our learning pattern base is initially composed of 161 cases of design patterns. We *randomly* divided our pattern base into 70% for the learning phase (111 pattern cases) and 30% for the test (50 pattern cases). The set of terms is built by selecting all the words that appear in the pattern cases.

As mentioned previously the enrichment step produced 2125 terms. However, to avoid working with large matrices, we retained in the learning base the original 551 terms produced after the pre-processing step. The remaining terms which came from the synonyms and translations were used in the frequency count: for each term, its frequency is incremented at each appearance of one of its synonyms or translations. For instance, the frequency of the word "women" will be incremented when we find "women", "lady", "miss", "madame", etc.

Note that the number of terms depends on the number of classes participating in each pattern. For example, the pattern *Singleton* contains just one class whereas the pattern *Factory* contains four participating classes. Consequently, the number of terms extracted varies from one pattern to another. Indeed, *Singleton* participates only with 8 terms whereas *Factory* participates with 68 terms. It is also worth mentioning that there are a lot of common terms among the different patterns. This further decreased the number of retained terms. In fact, when omitting overlapping terms, each pattern participates with a mean of 30.6 terms (as opposed to the original a mean of 33.1 terms).

Adapter2	Client	Target	Adapter	Adaptee
	CafeSetup	ActionListener	ReloadActionAdapter	MultipleDirClassDiagramReloader

AbstractFactory 2	Clients	AbstractProduct	Produc:		
	Clients	StampingEquipment	RightDoor	LeftDoor	Hood

Observer2	Subject	ConcreteSubject	Observers		ConcreteObserver
	Editor	HTMLEditor	MouseListener	KeyListener	HTMLEditor

Composite 9	Leafs		Composite	Component
	Button	Menu	CompositeOperand	Component

Figure 3: Extract of the pattern cases repository used for learning

3.2 Design pattern retrieval based on learning past experiences

As mentioned in the introduction, our pattern selection/recommendation uses an information retrieval technique whose core tasks are learning, classification and similarity measure. Several techniques for learning and classification exist in the literature, but it is unclear which one is definitively better than another in our context. This performance information shortage prompted us to experiment with different

learning methods to find the most suitable one for the recommendation task. In our experiments, we considered three IR methods which are: TF-IDF, LSI and PCA. We applied the same learning phase for the three IR methods. We developed these learning techniques using the R software [R, 05], a language and environment for statistical computing and data analysis.

Note that one advantage of our approach is that in the learning some terms are up weighted to reflect the relative importance of their corresponding participants in the pattern. This weighing decision stems from the fact that, in a design pattern, the various pattern participants play different roles with different levels of importance. For example, in the Observer pattern, the roles played by the classes are Observer, Subject, ConcreteSubject and ConcreteObserver; the Observer and Subject roles are the most important since they encode the essence of this pattern. Thus, we consider that the terms representing the Observer and subject are more significant. Consequently, in the learning step, these terms are up weighted to reflect their importance. In fact, we have multiplied the frequency of these terms in the learning corpus (term-document matrix) by two.

3.2.1 Learning with TF-IDF

TF-IDF uses a matrix containing the weights of terms appearing in the documents, the lines correspond to the patterns (n) and the columns correspond to the terms (p). The TF-IDF weights calculation need two matrices: the first one is the matrix of occurrences (n*p), it presents local weights determining the importance of a term in a pattern and represented by its frequency (tf). The other matrix represents the global weights that determine the distribution of each term in the database of patterns. Once these matrices are calculated, we obtain the values of TF-IDF of terms (t) for a particular pattern (d) in a set of patterns D.

The same TF-IDF computation is done for the query. The cosine similarity between any query and the TF-IDF matrix of the database of patterns varies between 0 and 1. The pattern of the database having a cosine value close to 1 is considered the most pertinent.

3.2.2 Learning with LSI

LSI identifies the pattern cases stored in the repository that are textually similar to the current case. For this purpose, a similarity matrix is calculated. First, LSI creates a term-document matrix which describes the occurrences of terms in documents; its rows correspond to terms and its columns correspond to documents. The element (*i*,*j*) of the matrix corresponds to the number of occurrences of the term *i* in the pattern case *j*.

On the other hand, the query contains terms extracted from the design fragment drawn by the designer. LSI uses each term belonging to the design fragment as a query to retrieve all terms similar to it in the stored cases, according to a cosine similarity. It is worth mentioning that LSI generates a new matrix A_k which represents the patterns in the dimension *K*. It is calculated as $A_k = T_K S_K D_K^{T}$. Therefore, when calculating the cosine similarities between the query and learning cases, we have to consider the same K dimensions obtained by the LSI method. Then for a query q a new query q_K must be calculated as $q_K = q^T T_K S_K^{-1}$. The cosine measure will then be

calculated between the new query vector q_K and the reduced document vector depicted from the D_K matrix. K represents the reduced number of dimensions. In order to determine the best value of K, we proceeded to many experiments where we try different values for this parameter.

3.2.3 Learning with PCA

Similar to the above methods, PCA uses the term-document matrix which describes the occurrences of terms in documents where columns correspond to terms extracted from the learning patterns and the rows correspond to the different patterns.

The PCA technique starts by calculating the covariance matrix between all terms where each term is described by a vector containing its occurrences in the n patterns. Then, the eigenvectors and the eigenvalues of this matrix are determined. The obtained eigenvectors define the orientations of the principal components of our input data. These components are linear combinations of the p=551 considered terms of our learning dataset. Thereby, each design pattern will have new coordinates in the new resulting space. As the new components describe better the original patterns, only a reduced dimension of this new space would be sufficient to find similarities between patterns and between a query and the set of learning design patterns.
In this case similarity between a query q and a pattern d is given by the Euclidian distance between these two elements. After determining the representative vectors of these two elements in the new PCA sub-space we proceed to the distance calculus then we choose the pattern having the minimal distance with the given query q. This distance is given by:

$$D(d,q) = \sqrt{\sum_{i=0}^{K} (q_i - d_i)^2}$$

where $q=(q_1, q_2, \ldots, q_K)$ represents the query vector in the new subspace, $d=(d_1, d_2, \ldots, d_K)$ is the pattern vector in the same subspace and K represents the reduced dimensions retained from the new PCA space.

As in the LSI case, the K value must be comprised between 2 and the number of pattern cases; in order to determine its best value, we have to proceed to many experiments and to depict the K value giving the best recommendation results. It is worth mentioning that the first dimensions in the PCA space are the most important since they account the most variation in the patterns set. Thus, neglecting the last dimensions will not affect results and will not cause any information lost.

3.3 Questionnaire

Our approach has the advantage of validating the UML diagram-based recommendations with an interactive set of questions about the design intention and context. Unlike existing text-based approaches which use abstract keywords, our approach expresses the questions with concepts taken from the application design. The questions are reformulation of the intent and context descriptions documenting the design patterns. The asked questions depend on the retrieval results already done in the preceding step. Thanks to the questionnaire, more recommendation accuracy can be achieved. Note that, unlike existing approaches that propose multiple questions to the designer (e.g., [Kung, 03], [Palma, 12], [Pavlic, 14]), ours reduces the number

of questions and it reformulates the questions while using the names of the elements that were identified as pattern participants in the pattern retrieval step. The reformulation facilitates the comprehension of the questions to the designer.

Making sure not to burden the user with many questions, our approach limits the questions to only the three patterns with the highest similarity values. For each of the three top-ranked patterns, three characterizing questions are presented to the user; if one of the questions is answered negatively, we move to recommend the following pattern and we present its own three characterizing questions, and so on so forth. Hence, at most the user will have to answer nine questions. Evidently, the question formulation is very important and we have prepared the characterizing questions of the 18 patterns included in our pattern repository with care, relying on our expertise with the patterns.

To elaborate the characterizing questions, we extracted for each pattern its intent, its applicability and its context. We extracted this information from the book of Gamma [Gamma, 95] which proposes a commonly used documentation format for patterns containing: the pattern name and classification, the intent (i.e., a description of the goal behind the pattern and the reason for using it), the motivation (i.e., a scenario describing the problem), the applicability (i.e., situations in which the pattern is usable), the structure (i.e., class diagrams and interaction diagrams representing the pattern), the participants (i.e., a listing of the classes and objects used in the pattern and their roles in the design), the collaboration and the consequences (i.e., a description of the results).

For an example of questionnaires, suppose that the Strategy pattern was retrieved as a candidate in a design where the *NetPayable* class was identified as playing the role of *Strategy* and the *Invoice* class was identified as playing the role of a *Context;* then the questionnaire step will formulate the following three characterizing questions:

Strategy questionnaire	***Yes***	***No***
Do you want to define a family of NetPayable?	■	□
Do you want to encapsulate each NetPayable, and make them interchangeable?	■	□
Do you want to vary the behavior of the Netpayable independently from the class Invoice that use it?	■	□

If any of the answers to the questions about the intention are negative, we move to questions about the intentions of the next retrieved patterns, according to similarity ranking; this alternative analysis is especially needed when the similarity values are very close between the first pattern and those following it.

In order to evaluate the questionnaires formulated about the 18 design patterns, we used our tool PRT (described in Section 5) to test the approach with ten graduate students taking a Master's software engineering course. We proposed three case studies to each student who used PRT to find recommended patterns and then to answer their questionnaire; that is, a total of 30 experimentations were conducted with PRT. Afterwards, we asked the students about the clarity and usefulness of the questions: Only one out of the ten students said that the questions were not adapted to the case studies. Moreover, in the 30 cases, only 4 cases needed to go to a second round of the questionnaire based on the user responses. This shows the efficiency of

both the retrieval step and the questionnaire formulation which assisted the user to decide with no hesitation.

3.4 Pattern instantiation

The final step in our approach is instantiation which adapts the recommended pattern to the domain by modifying, if needed, certain design properties. Pattern instantiation essentially changes the design structure in order to conform it with the structure of the recommended pattern.

To illustrate the instantiation step, let us consider the design fragment drawn in Figure 4 which was recommended to match the *Composite* pattern. In the recommendation step, it was determined that the classes Circle, Triangle Shape, Square must play the role of a Leaf thus there must be an inheritance relation between the class Form that plays the role of Composite and these classes. Moreover, the class Group plays the role of Component consequently the relationship that must exist between Group and Form is an aggregation. Thus, instantiation must remove the old relationships and replace them with these new relations (inheritance and aggregation).

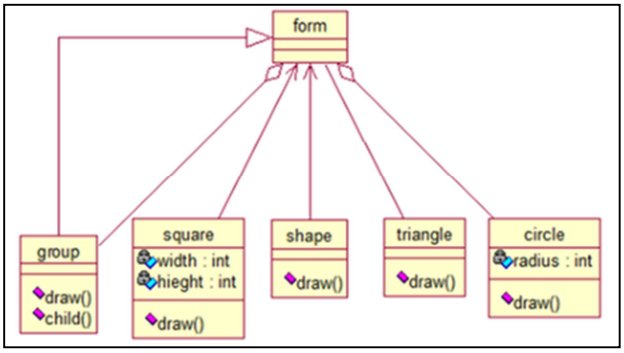

Figure 4: An example of designer input

The participants in the recommended pattern are: Form (Composite), Group (Component), Square (Leaf), Shape (Leaf), Triangle (Leaf), and Circle (Leaf). It has been found similar to the learned pattern case corresponding to Composite and where GroupFigure plays the role of a component, Figure plays the role of a composite and the leafs are (TextFigure, PolygonFigure, RoundRectangleFigure).

4 Pattern retrieval methods evaluation

In order to choose the best information retrieval technique (among TF-IDF, LSI, PCA) for our pattern selection system, we evaluated the results of the three experimentally. For evaluation purposes, we used the measures of precision and recall introduced in the domain of information retrieval.

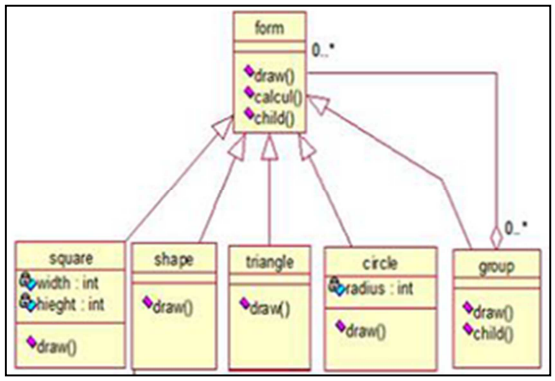

Figure 5: The recommended Composite pattern instantiated in Figure 4

In our experiment, precision and recall are computed separately for each design pattern type. In this case, precision assesses the number of design patterns correctly recommended by our tool for a given type among all the recommended patterns for that type. In the same way, recall assesses the number of correct design patterns recommended by our tool for a given pattern type among all the existing patterns in this type. We also used the F-measure, which takes into account the two evaluations (recall and precision). In fact, it is a measure of overall efficiency which is given by:

$$\mathbf{F\text{-}measure} = \frac{2 \times (\mathbf{Precision} \times \mathbf{Recall})}{(\mathbf{Precision} + \mathbf{Recall})}$$

Seeking a detailed and complete evaluation of the three proposed methods, we performed two series of experiments. The first one is dedicated to a detailed evaluation according to the above explained steps. In that, we split our database into two parts. One part containing 111 design pattern cases and it is devoted to the learning step and the second part contains 50 cases and it is used for the test step. Thus, the same cases are used to evaluate the three methods.

In the second series of experiments, and for more accurate results, we consider a 5-fold cross-validation where our dataset is divided in 5 folds. A fold contains 32 design patterns selected randomly. Each time, one of the folds is used for testing and the rest for training. Therefore, the evaluation step is repeated 5 times with different learning and test sets. The final result is calculated as average of the 5 obtained F-measures. This way, the obtained F-measure values are more statistically significant.

Table 1 presents the evaluation results for the different patterns when using TF-IDF in the first series of experiments. After calculating the obtained F-measure value for each class, the overall value of efficiency is 68.4%. This efficiency is obtained as an average of all F-measure values and it is considered as a relatively low value. Note that, the zeroes in Table 1 are explained by the fact that the data distribution for these patterns is unbalanced and inequitable, which makes TF-IDF non efficient in these cases. More specifically, for these two patterns, TF-IDF produced bad results since there is no *direct* similarity with existing cases. In fact, despite the extension of the

vocabulary of both patterns, their low number of occurrences did not help to improve the results of TF-IDF. In contrast, the results of these patterns with LSI and PCA are slightly better since these techniques exploit the context-based similarity.

	R	***P***
Abstract Factory	0	0
Adapter	1	0.625
Bridge	1	0.666
Builder	1	1
Chain of responsibility	1	1
Command	0.666	1
Composite	0.5	0.285
Decorator	0.666	1
Factory	0.666	0.666
Iterator	1	1
Mediator	1	1
Observer	1	0.6
Proxy	1	1
Singleton	1	0.333
State	0.666	1
Strategy	0.333	1
Template	0	0
Visitor	0.5	1
Average (R/P)	0.657	0.660
Average efficiency	0.684	

Table 1: Evaluation results with the TF-IDF technique

For the LSI technique, we tried several values of K to adjust this parameter to its best value. Table 2 presents the evaluation results for the different design pattern classes when using LSI. The best value of the overall efficiency is 93.2% obtained with K=30.

Table 3 presents the evaluation results for the different design pattern classes when using PCA. In the same manner, we vary the value of K to depict the best results. In this case, K=28 gives the best overall efficiency which is calculated as the average F-measure value and it is equal to 95.6%.

In the evaluation (illustrated in Table 3), the value of precision is 97.2 % which is explained by the fact that we found some false positives (i.e., incorrectly recommended patterns). The average Recall value is 95.8 % indicates that we have some false negatives (i.e., true patterns not recommended). However, the obtained F-measure value is very encouraging and it shows the capacity of the PCA method to retrieve the hidden relations between terms and to fulfil the semantic requirements needed when searching for similar pattern cases.

When performing the second series of experiments with 5-fold cross validation, the performances of the three methods were confirmed. Indeed, the final average F-measure value obtained with the TF-IDF method is 68.1% which is very close to the value of 68.4% obtained in the first series of experiments. For LSI, the best results were given when K is equal to 30 with an average F-measure value of 93.8%. Finally,

the PCA method achieved the best performances with an average F-measure value of 95.1% with a reduced number of dimensions (average value of K=29).

	K=10		K=20		K=30		K=40		K=25		K=35	
	R	P	R	P	R	P	R	P	R	R	R	P
Abstract Factory	0.666	0.5	1	1	1	1	0.333	1	0.666	0.666	0.666	1
Adapter	1	0.714	1	0.714	1	1	1	0.833	1	1	1	0.714
Bridge	1	1	1	1	1	1	1	0.666	1	1	1	0.4
Builder	1	1	1	1	1	1	1	1	1	1	1	1
Chain of responsibility	1	1	1	1	1	1	1	1	1	1	1	0.75
Command	1	1	1	1	1	1	1	1	1	1	1	1
Composite	0.75	0.75	0.75	1	0.75	1	0.75	1	0.75	0.75	0.5	0.666
Decorator	0.666	1	1	0.75	0.666	1	0.666	1	0.666	0.666	0.333	1
Factory	0	0	0.666	0.5	1	0.75	0.666	0.666	0.666	0.666	0.333	0.5
Iterator	1	1	1	1	1	0.666	1	1	1	1	1	1
Mediator	1	1	1	1	1	1	1	0.666	1	1	0.5	1
Observer	0.666	1	1	1	1	1	0.666	1	1	1	1	0.75
Proxy	1	0.6	1	0.75	1	1	1	1	1	1	1	1
Singleton	1	1	1	1	1	0.5	1	0.333	1	1	0	0
State	1	1	1	1	1	1	0.666	1	1	1	1	0.75
Strategy	0.666	1	0.666	1	0.666	1	0.666	1	0.666	0.666	0.666	1
Template	0	0	0	0	1	1	1	1	1	1	1	1
Visitor	1	1	0.5	1	1	1	1	1	0.5	0.5	0.5	1
Average(R/P)	0.800	0.809	0.865	0.873	0.949	0.939	0.856	0.898	0.884	0.884	0.749	0.807
Average efficiency	***0.794***		***0.857***		***0.932***		***0.845***		***0.866***		***0.87***	

Table 2: Evaluation results with the LSI technique

Through the different evaluations, we noticed that the results of TF-IDF are far from those of PCA and LSI, this is explained by the fact that TF-IDF suffers from the sparse representation problem. In fact, as query contains only few words, its corresponding TF-IDF vector will be sparse. It is the same for patterns vectors. Therefore the cosine similarity will only rely on scarce common words and similarity values will be distorted. In contrast, LSI and PCA do not rely on words but rather on concepts, in that, words having same contexts can be revealed similar. It is this propriety that makes all the difference between the two kinds of techniques.

Henceforth, the similarity measure can be properly calculated between queries and patterns even they do not share enough words. The obtained results show that it is possible to define a kind of "concepts" as design patterns. In fact, combination of words with corresponding weights can define latent composition of a design pattern. This justifies the good results especially with the PCA and the LSI methods. In contrast, TF-IDF based only on existing words in the design pattern achieves lower performances. The most likely explanation is that terms with a low frequency will not appear in the test cases and cannot participate in the design pattern detection step. Note that the concepts constitute good solutions in this case because they are not limited to only one term. Moreover, we notice that for example the template pattern gives bad results with LSI and PCA when using low values for K, this is explained by the fact that when reducing k, the coverage of the lexicon becomes insufficient since it is very extended and since the frequency is low. Finally, the fact that PCA gave the best results is explained by the efficiency of the pre-processing step initiated by this

method. Indeed, while the singular value decomposition: SVD is applied on the term-document matrix with the LSI method, it is applied on the covariance matrix with the PCA method and this makes difference. Thus, the obtained PCA space reflects the variance between data points and allows to better describing similar and dissimilar patterns and queries.

	K=2		K=10		K=20		K=28		K=40	
	R	P	R	P	R	P	R	P	R	P
Abstract Factory	0.333	0.5	1	1	1	1	1	1	1	1
Adapter	0.4	0.666	**1**	**1**	1	0.833	1	0.833	1	0.833
Bridge	0.5	0.5	0.5	0.5	1	1	1	1	1	1
Builder	0.333	1	1	1	1	1	1	1	1	1
Chain of responsibility	0.666	0 .666	1	1	1	1	1	1	1	1
Command	0	0	1	1	1	1	**1**	**1**	**1**	**1**
Composite	0.5	1	0.75	1	0.75	0.75	0.75	1	0.75	1
Decorator	0.333	0.5	1	1	1	1	1	1	0.666	1
Factory	0.333	0.25	0.666	1	1	1	1	1	1	1
Iterator	0	0	1	0.5	1	1	1	1	1	0.666
Mediator	0.5	0.142	1	1	1	1	1	0.666	1	0.666
Observer	0.333	0.25	1	1	1	1	1	1	1	1
Proxy	0.666	0.5	1	1	1	1	1	1	1	1
Singleton	0	0	1	1	1	0.5	1	1	1	1
State	0	0	1	1	1	1	1	1	1	1
Strategy	0	0	1	0.75	1	1	1	1	1	1
Template	0	0	0	0	0	0	1	1	1	1
Visitor	1	0.666	1	1	1	1	0.5	1	0.5	1
Average (R/P)	***0.327***	***0.351***	***0.884***	***0.875***	***0.93***	***0.893***	***0.958***	***0.972***	***0.939***	***0.953***
Average Efficiency	***0.32***		***0.87***		***0.906***		***0.956***		***0.934***	

Table 3: Evaluation results with the PCA technique.

In order to compare our work against state of the art, we consider the work of Hasheminejad et al. [Hashminejad, 12] which is based on textual queries and which has the best results in design patterns selection. Its evaluation results for learning GoF patterns using 19 real design problems when using the learning technique Support Vector Machine (SVM) produced a precision of 89%, a Recall of 84% and an efficiency of 86%. Our technique has a precision of 97.2% and our evaluation shows that our design pattern recommendation approach has an average improvement of 8.2% in terms of precision over the best known approach. In order to have a fair comparison with other existing approaches; one must evaluate all the approaches on the same corpus. We believe that it is necessary to have a benchmark for design pattern recommendations and that this would be an interesting research axis.

5 Tooling

The functional architecture of our Pattern Recommendation system Toolset (PRT) contains five modules: Design query formulation, Patterns retrieval with PCA, Questionnaire, Pattern instantiation, and Design viewing. The designer first draws with the ArgoUML editor the application design. PRT imports class diagrams drawn

according to the UML language and saved in XMI files. From the design saved as XMI file, PRT extracts the names of the classes and formulates with them a query that is passed to the retrieval module. This latter applies the PCA technique and recommends a list of appropriate patterns. Each recommended pattern in this list is described in terms of: the pattern name, and a set of pairs of pattern element-corresponding design element. The questionnaire module takes this list of recommended patterns and starts reformulating their characterizing questions in terms of the design elements. In the example of Figure 4, three cases were identified as having the highest textual similarity; this is done using the learnt models. The first pattern to recommend is Composite with a similarity value that equals 0.92. To validate this recommendation, PRT asks the designer the questions characterizing this pattern; Figure 6 illustrates the question relative the pattern intention. If any of the questions is answered negatively, PRT re-conducts the questionnaire phase with the pattern ranked next on the list of recommended patterns.

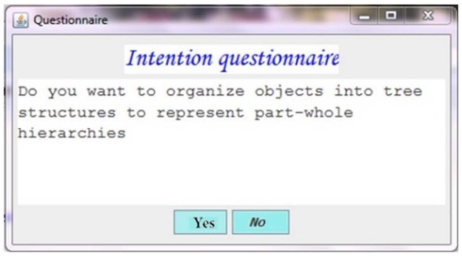

Figure 6: PRT Questionnaire for Figure 4: The intention part

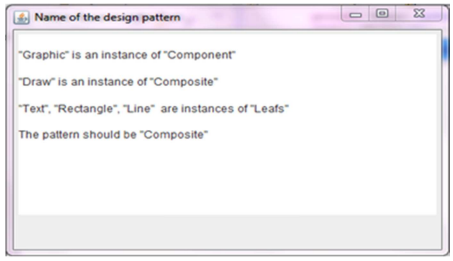

Figure 7: PRT report for the recommended pattern of Figure 4

Once a recommended pattern is validated by the designer, PRT produces a recommendation report; Figure 7 shows the recommendation report for the validated Composite pattern for the example of Figure 4. Finally, the instantiation module transforms the design into the pattern solution by modifying the XMI file thanks to the API JDOM and to the tag DOM Builder and the design viewing module displays the structure of the recommended pattern with ArgoUML.

6 Conclusion

In this paper, we presented a pattern recommendation approach and its interactive toolset. The approach assists the designer in choosing the appropriate design pattern and instantiating it. In the presented work, the recommendation of a design pattern is guided by the use of the PCA technique (using the UML diagram) and a questionnaire (expressing in a textual form the design intent). Compared to TF-IDF and LSI, PCA was experimentally shown to be the best retrieval technique to identify a list of candidate patterns from past pattern experiences. Compared to existing questionnaire-based approaches, our approach has the merit of validating the PCA-

based suggestions by formulating questions about the recommended pattern context and intention in terms of the design terminology.

As future works, we plan to include more case studies to cover a richer terminology containing methods and attributes names. In addition, we plan to apply the approach on all GoF design patterns and other domain specific design patterns (e.g. Service Oriented Architecture (SOA) patterns) in order to judge its effectiveness. Furthermore, we plan to conduct a study on other learning algorithms such as the K-nearest neighbors.

References

[Binkley, 11] Binkley, D., Lawrie, D.: Information retrieval applications in software maintenance and evolution. In: Encyclopedia of Software Engineering, 454–463, 2011.

[Blomqvist, 08] Blomqvist, E.: Pattern ranking for semi-automatic ontology construction. In: Proceedings of the ACM Symposium on Applied Computing, 2248–2255, 2008.

[Bouassida, 12] Bouassida, N., Koas, A., Ben-Abdallah, H.: A design pattern recommendation approach. In: 2nd IEEE International Conference on Software Engineering and Service Sciences, 15-17 July, Beijing China, 2012.

[Díaz, 11] Díaz, P., Aedo, I., Navarro, I.: Using recommendation to help novices to reuse design knowledge. In: International Symposium for End User Development, IS-EUD, LNCS 6654, 331-336, 2011.

[Deerwester, 90] Deerwester S.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391-407, 1990.

[Dumais, 93] Dumais, S. T.: LSI meets TREC: A status report. In: D. Harman (Ed.), The First Text REtrieval Conference (TREC1), National Institute of Standards and Technology, Special Publication, 137-152, 1993.

[Gamma, 95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Pattern - Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.

[Gamma, 07] Gamma, E., Eggenschwiler, T.: http://www.jhotdraw.org (2007), Accessed February 2015.

[Ganter, 96] Ganter B., Wille R.: Formal concept analysis: Mathematical foundations. Springer-Verlag, 1996.

[Gomes, 02] Gomes, P., Pereira, F.C., Paiva, P., Seco, N., P. Carreiro, Ferreira, J., Bento, C.: Using CBR for automation of software design patterns. In: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, 534- 548, 2002.

[Guéhéneuc, 07] Guéhéneuc, Y., Mustapha, R.: A simple recommender system for design patterns. In: Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories, 2007.

[Hasheminejad, 12] Hasheminejad, S. M. H., Jalili, S.: Design patterns selection: An automatic two-phase method. The Journal of Systems and Software 85, 408– 424, 2012.

[Hotelling, 33] Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24, 417-441, 1933.

[JRefactory, 07] JRefactory, htstp://jrefactory.sourceforge.net/(2007). Accessed February 2015

[JUnit, 07] JUnit, http://www.junit.org, (2007). Accessed 26 February 2015

[Kim, 07] Kim, D.K., Khawand, C.E.: An approach to precisely specifying the problem domain of design patterns. Journal of Visual Languages and Computing, 18, Elsevier, 560–591, 2007.

[Kim, 08] Kim, D.K., Shen, W.: Evaluating pattern conformance of UML models: a divide- and-conquer approach and case studies. Software Quality Journal, Springer, 16 (3), 329–359, 2008.

[Kung, 03] Kung, D. C., Bhambhani, H., Shah, R., Pancholi G.: An Expert system for suggesting design patterns: A methodology and a prototype. The Springer International Series in Engineering and Computer Science (731), 287-318, 2003.

[Hsueh, 07] Hsueh, N.-L., Kuo, J.-Y., Lin, C.: Object-oriented design: a goal-driven and pattern-based approach. Software and Systems Modeling (springer), 8(1), 1–18, 2007.

[Moynihan, 06] Moynihan, G. P., Suki, A., Fonseca, D. J.: An expert system for the selection of software design patterns. Expert Systems 23(1), 39-52, 2006.

[Maher, 95] Maher, M.L., Balachandran, M., Zhang, D.: Case-Based reasoning in design. Lawrence Erlbaum Associates, 1995.

[Miller, 95] Miller, G. A., WordNet: a lexical database for English, Communication of the ACM 38 (11), 39–41, 1995.

[Muangon, 13] Muangon, W., Intakosum, S.: Case-Based reasoning for design patterns searching system. International Journal of Computer Applications 70 (26), 2013.

[Muangon, 09] Muangon, W., Intakosum, S.: Adaptation of Design pattern retrieval using CBR and FCA. In: Fourth International Conference on computer sciences and convergence Information technology, 2009.

[Palma, 12] Palma F., Farzin, H., Guéhéneuc, Y.G., Moha N.: Recommendation system for design patterns in software development: a DPR overview. In: Third International Workshop on Recommendation Systems for Software Engineering (RSSE), Zurich, 2012.

[Pavlič, 09] Pavlič, L., Heričko, M., Podgorelec V., Rozman, I.: Improving Design Pattern Adoption with an Ontology-Based Repository, Informatica 33, 189–197, 2009.

[Pavlič, 14] Pavlic L, Podgorelec V., Hericko M.: A question-based design pattern advisement approach. Comput. Sci. Inf. Syst. 11(2), 645-664, 2014.

[Robertson, 04] Robertson, S.: Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation 60 (5), 503–520, 2004.

[R, 05] R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Austria. ISBN 3-900051-07-0, 2005.

[Salton, 65] Salton, G., Buckley, C.: Term-weighing approach in automatic text retrieval. Information Processing & Management 24(5), 513-523, 1965.

[Suresh, 11] Suresh, S., Naidu, M., Asha Kiran, S.: Design Pattern Recommendation System Methodology (Data Model and Algorithms). In: International Conference On Computational Techniques and Artificial Intelligence, 2011.

[Schmidt, 15] Schmidt, H.: Treetagger, www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (2015). Accessed 26 February 2015