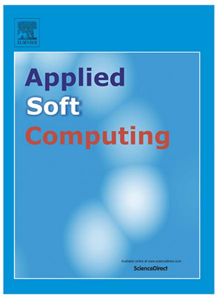
Automated framework for classification and selection of software design patterns

Shahid Hussain, Jacky Keung, Muhammad Khalid Sohail, Arif Ali Khan, Manzoor Ilahi



PII: S1568-4946(18)30617-3
DOI: https://doi.org/10.1016/j.asoc.2018.10.049
Reference: ASOC 5171

To appear in: *Applied Soft Computing Journal*

Received date : 15 June 2018
Revised date : 3 October 2018
Accepted date : 25 October 2018

Highlights

1. Propose a framework to overcome the limitation of existing automation techniques
2. Unsupervised learning techniques are used to exploit the proposed framework.
3. The aim of proposed framework is the classification and selection of the software design pattern(s).
4. Propose an evaluation model to assess the effectiveness of the proposed framework.
5. Present three case studies in different domains such as object-orinted development, real time and security based application.

Automated Framework for Classification and Selection of Software Design Patterns

[1]Shahid Hussain, [2]Jacky Keung,[3]Muhammad Khalid Sohail, [4]Arif Ali Khan, [5]Manzoor Ilahi,

[1,3,4,5]Department of Computer Science, COMSATS Institute of Information Technology, Islamabad

[2]Department of Computer Science, City University of Hong Kong

[1]shussain@comsats.edu.pk, [2]Jacky.Keung2-c@cityu.edu.hk, [3]khaled_sohail@comsats.edu.pk, [4]arifkhan@comsats.edu.pk, [5]tammimy@comsats.edu.pk,

ABSTRACT

Though, Unified Modeling Language (UML), Ontology, and Text categorization approaches have been used to automate the classification and selection of design pattern(s). However, there are certain issues such as time and effort for formal specification of new patterns, system context-awareness, and lack of knowledge which needs to be addressed. We propose a framework (i.e. Three-phase method) to discuss these issues, which can aid novice developers to organize and select the correct design pattern(s) for a given design problem in a systematic way. Subsequently, we propose an evaluation model to gauge the efficacy of the proposed framework via certain unsupervised learning techniques. We performed three case studies to describe the working procedure of the proposed framework in the context of three widely used design pattern catalogs and 103 design problems. We find the significant results of Fuzzy c-means and Partition Around Medoids (PAM) as compared to other unsupervised learning techniques. The promising results encourage the applicability of the proposed framework in terms of design patterns organization and selection with respect to a given design problem.

Keywords

Design Patterns, Design Problems, Unsupervised Learning, Text categorization, Feature Selection, Supervised Learning

1. INTRODUCTION

In SDLC (Software Development Life Cycle), software designing phase is a stimulating and important task. Like other design methods, architecture based method uses many architectural styles. Each design method is used as a coordination tool to describe the design components and the existence of their relationship. Moreover, each method is considered as a bridge between critical requirements and implementation [1] of a system.

Though, several architectural based design methods have been introduced. However, Attribute Driven Design (ADD) is widely used to apply certain patterns and strategies in order to achieve the quality attributes. Developers can understand the design complexity of a selected design pattern or its component via the description of design requirements [2].

The developers solve a recurring design problem via their skills and encapsulate the solution in canonical form. For example, the canonical solution of Bridge design pattern is encapsulated to decouple the abstraction from their implementation and help in independent working [3]. Subsequently, the canonical solution of Fat Menus pattern [4] in terms of website designing can be used to organize numerous pages into different categories and aid people to expose most of the pages. The declined quality of software systems can trigger the use of the refactoring and employment of design patterns [3, 4, 5, 6]. Though, in the recent studies [7, 8, 9], the authors have reported the interest of developers and users to employ the design patterns and their implications with respect to certain aspects. However, searching and selection of design patterns are the two main issues which can be addressed before the employment of right pattern [10, 11]. The first issue that is searching for correlated patterns which depend on the classification schemes for pattern organizations. Such as, in the field of object-oriented design, Gamma et al. [3] (Focused on the instantiation, composition, and communication of class objects), Coad et al. (Stereotypical responsibilities and scenario interactions aspect) [12] and Pree et al. [13] (Recursive Structures and Abstract Coupling between objects) presents their own classification scheme to group the design pattern with respect to certain aspects. Moreover, there are certain online pattern libraries such as L. Rising - Patterns Almanac 2000[1] and euroPLoP[2], which are used to catalog the patterns without using well-defined classification, which might cause of inconsistency and overlapping issues in the pattern catalog.

The second issue is the selection of right patterns in the existence of spoiled patterns [14], challengeable design patterns [15], closeness between catalogs on online repositories, the number of existing and emerging pattern, heterogeneity in the pattern description [11], and interdisciplinary design patterns [16]. A novice designer is required knowledge to understand the closeness between patterns [10, 13] and to select the appropriate design pattern(s) for given design problem at hand. The promising results of automated systems for the identification of right design patterns depict their applicability during the designing phase. In this regard, existing efforts are grouped into three categories, namely UML-Based [17, 18, 19, 20], Ontology-Based [21, 22, 23] and Text categorization based approach [24, 25, 26, 63]. Formal specification of new patterns in existing catalog, patterns organization with respect to expert opinions, and large sample size for classifier's training are the key issues reported in terms of complex and interdisciplinary design pattern collections [17-26].

The applicability of Text categorization approach in numerous domains such as web page classification [28], author identification [30], spam e-mail filtering [27], sentiment analysis [29], and the design pattern selection [24, 25, 26] encourage us for its use to discuss these issues.

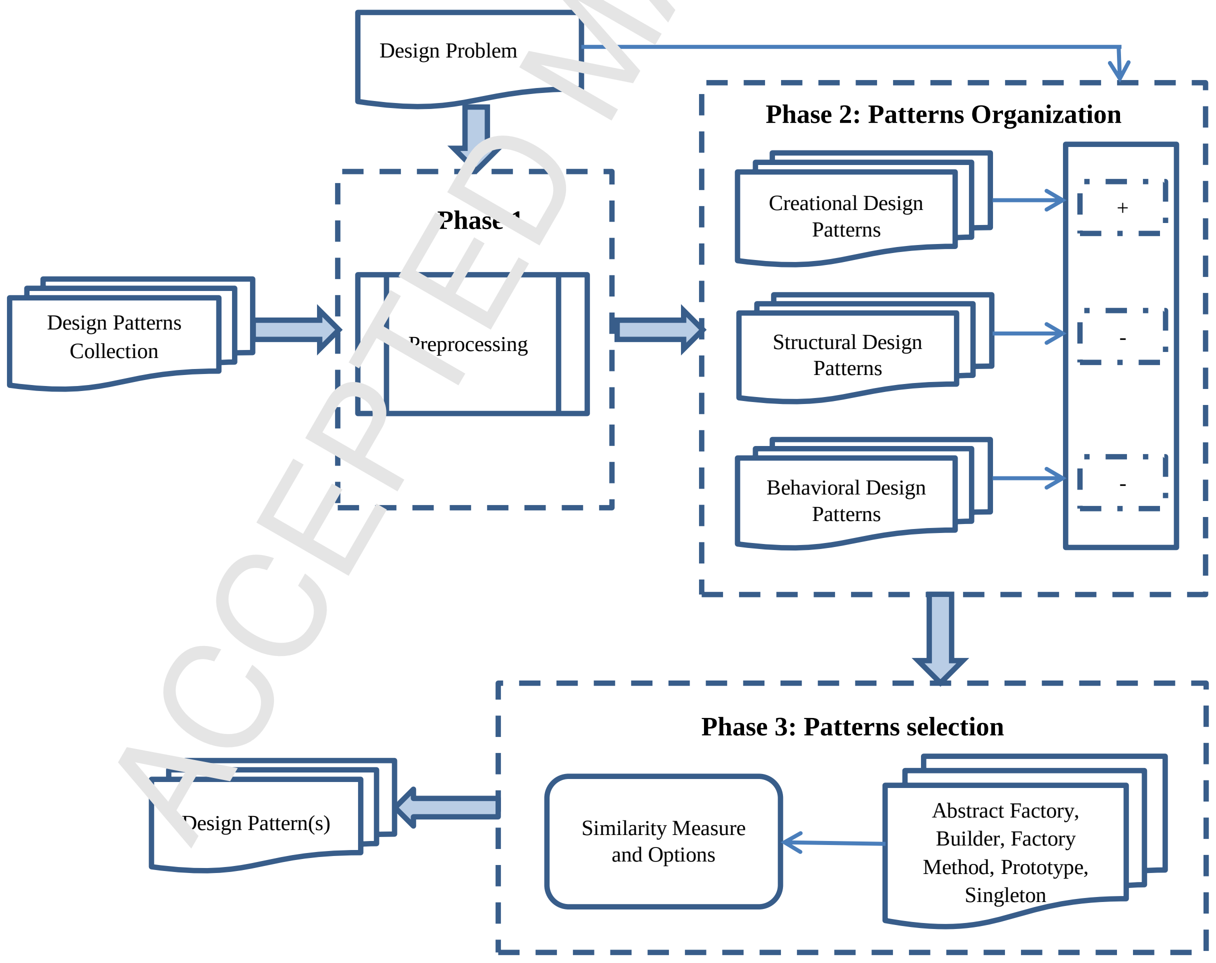

Design Problem
Phase 2: Patterns Organization
Creational Design Patterns
+
Phase 1
Preprocessing
Design Patterns Collection
Structural Design Patterns
-
Behavioral Design Patterns
-
Phase 3: Patterns selection
Design Pattern(s)
Similarity Measure and Options
Abstract Factory, Builder, Factory Method, Prototype, Singleton

Figure 1. Overview of the proposed framework

[1]http://www.smallmemory.com/almanac/
[2]http://europlop.net/content/repositories

Consequently, we propose a framework using Text categorization approach [24, 25, 26] and unsupervised learners to achieve three objectives 1) to organize the patterns into apposite numeral of pattern classes (i.e. Categories), 2) to select the appropriate design pattern class for a design problem, and 3) to suggest and select the appropriate design pattern(s) for a given design problem. The layout of the proposed framework is shown in Figure 1. The inputs of the proposed framework are the descriptions of design patterns of a target pattern catalog and/or a design problem. The proposed framework is functional in three phases. In Preprocessing phase, we recommend a set of activities such as the Removal of stopwords, words stemming, creation of Vector Space Model (VSM), weighting methods, and feature selection. Moreover, in Patterns organization phase, the unsupervised Learning is performed via certain techniques such as Fuzzy c-means, K-means, Agglomerative, Partitioning Around Medoids (a.k.a. K-medoids with distance measures Euclidean and Manhattan). In this phase, we achieved two objectives. The first objective is to organize the pattern(s) and second objective to suggest a pattern class for a problem. Finally, in the design pattern selection phase, design pattern(s) are selected from the suggested pattern class.

The preliminary study of the proposed method (exploited with one unsupervised learner and three design problems in the context of GoF catalog) has been presented in an affiliated workshop of the 40th Annual Computer Software and Applications Conference (COMPSAC2016) [25]. Subsequently, in the subsequent study, the efficacy of the framework is evaluated with three pattern collections, the proposed evaluation model, and a set of 80 design problems. In these two subsequent studies [25, 26], the proposed method is employed through Fuzzy c-means. Since the selection of outperform classification and clustering algorithms depends on the problem's context [31]. Consequently, in this subsequent study, we consider four unsupervised learning techniques, namely Fuzzy c-means, k-means, Agglomerative, and Partitioning Around Medoids (a.k.a. K-medoids with distance measures Euclidean and Manhattan) [32, 33, 34] to assess the efficacy of the framework. Moreover, we also increase the sample set of design problems.

The rest of the paper is described in nine sections. In section 2, related work in terms of classification and selection of design patterns is presented. In section 3, software design patterns and text categorization approach is introduced. In section 4, we present our proposed framework. In section 5, we present the evaluation model to assess the effectiveness of the proposed framework. In section 6, we present pseudocodes to describe the working procedure of the proposed framework. In section 7, we discuss the results in the context of three case studies. In section 8, we summarize the results. In section 9, we discuss the limitations of the proposed framework. Finally, in section 10, we conclude the study.

2. RELATED WORK

We have summarized the existing efforts (in terms of automation) of software engineering research community in two aspects, 1) design patterns classification and 2) selection of correct design pattern(s).

2.1 Classification of Design Patterns

The interest and experience of developers is used to introduce the new classification schemes for the organization of design patterns of certain domains such as object-oriented development [3], real-time application [35] and so on [36, 37]. In this regard, research community has used the either problem's intent or relationship between the design pattern solutions to group them in different categories. The name, domain, and list of categories of problem-based categorization schemes are shown in Table 1. The summary of existing efforts describe the number of high-level categories which depends on the type and complexity of target problems. For example, Gamma et al [3] introduce three high-level categories namely Creational, Structural, and Behavioral in the context of object-oriented development to solve recurrent problems. In the context of real time applications [35], Douglass [35] introduce a catalog of 34 patterns, which are grouped in five categorizations on the bases their relevancy. Such as Fixed Size Buffer, Garbage Collection, Garbage Compactor, Pool Allocation, Smart

Pointer, and Static Allocation are grouped in a Memory category. Subsequently, Schumacher [37] classifies 49 security patterns in 8 groups on the bases of security issues. Such as, Order Locking, Priority Ceiling, Priority Inheritance, Simultaneous Locking, Critical Sections, Highest Locker are grouped in Resource category. In terms of the relationship between solutions, Zimmer [38] look at the all aspects related to link and integration of solutions. Similarly, Kim and Han [39] focus on the dependencies between design pattern groups and analyzed them through their solution structure.

Table 1. List of categories and domain for design patterns based problem-based categorization

Authors	Context	High-Level Groups
Gamma et al. [3]	Object-oriented Design Problems	Structural, Creational, and Behavioral
Pree [13]	Object-oriented Design Problems	Abstract Coupling and Recursive Structures based classification.
Coad et al. [12]	Object-oriented Design Problems	Transaction, Aggregate, Plan and Interaction
Tichy [40]	Software Design Problems	Convenience, Decoupling, State Handling, Variant Management, Control, Compound, Virtual Machines, Concurrency & Distribution
Rising [36]	Application Type	Air Defense, Accounting, System Modeling, Hypermedia, Database, Trading, Interactive, C++ Idioms, Persistence & Trading
Douglass [35]	Real Time	Memory, Safety and Reliability, Resources , Concurrency, and Distribution
Booch [15]	Architectural Design	45 categories
Castro et al. [41]	Software Development Process	Workflow, Participants, Input-Outputs and Knowledge
Russell et al. [42]	Workflow Management System	Data Interaction, Data Visibility, Data-based Routing, Data Transfer

2.2 Selection of Design Pattern

Literature depicts that three widely known approaches have been used by SE research community in the context of automation of the design pattern selection process namely, Ontology-based, UML-based, and text categorization approach.

Kim and Khawand [17] target the specification of problem domain for the suggestion of suitable design pattern(s). They aim at the relation of the known problem with the problem domain of a design pattern. In this regard, Kim and Khawand used an UML-based RBML (Role-Based Modeling Language), a pattern specification language to construct the meta-model. They used collaboration and class diagrams behavioral and structural features of software and identify a corresponding pattern. The main limitation in their work scalability and structural similarity. The former constraint is related to the number of variation of design patterns [19].

Like Kim and Khawand [17], Hsueh et al [18] also use the UML-based method and perform a case study with the inadequate numeral of design patterns. Hsueh et al present a systematic way for selection of suitable design pattern by considering four viewpoints, such as problem-view, activity-view, tradeoff-view, and requirement-views. The aim of activity-view perspective is to highlight the use of design pattern for a design activity. Similarly, the aim of problem-view perspective is to highlight the use of design pattern to prevent exceptions for a real design problem. Moreover, the aim of requirement-view perspective is to highlight the use of design pattern to enhance and accommodate the non-functional requirements. Finally, in order to overcome the design conflict, tradeoff-view is considered. The number of design patterns and their selection via framework are the main limitations [18]. In a recent study [20], Rouhi and Zamani addressed the formal basis and the interrelationship of patterns and present a new formalism way of patterns and Pattern Languages (PL).

In their study, Hasso and Carlson [21] accomplish the semantic analysis of sentences of problem domain of design patterns and perform pattern classification based on the words meaning. Authors used the linguistic theory for semantic analysis and perform pattern classification and selection via verbs meaning. Lack of evaluation is the main constraint. Khoury et al [23] used OWL (Ontology Web Language) to design an ontological interface. The aim of Khoury et al study was to discover the set of appropriate security patterns. The proposed an ontological interface aid to mapping the security requirements with threat model, security bugs, and security errors. Subsequently, Blomqvist [22] introduces an ontology-based approach named OntoCase to rank and suggest the suitable design pattern(s). The author used class matching, centrality, relation matching, semantic similarity, and density to describe the working criteria for OntoCase. Lack of evaluation and formal description of all software design domain are the main constraints. Recently, Velasco-Elizondo et al. [11] analyze the architectural pattern via knowledge representation and information extraction to ease the inexperience architects. However, the specification of complex patterns and their computation time are the main constraints. Though we can observe some difference between UML-based and ontology-based automated system/tool, however, the specification of design pattern is with respect to the specific catalog. For example, in the area of object-oriented, the Tsantalis [43] tool is used to identify the design pattern of GoF pattern collection rather than the [12, 13] patterns.

In the context of text categorization based approaches, Hasheminejad and Jalili [24] present a two-phase method for the selection of proper design pattern(s) via the similarity between the explanation of design pattern and design problems. The first phase focuses on the classifier's learning for each pattern category. The second phase is focused on the determination of a candidate pattern class for each given design problem. The use of limited number of feature selection techniques namely Document Frequency (DF), mapping between number of classifiers and pattern classes, and relation of classifier's performance with data sparsity and sample size are the main constraints [24]. In order to overcome these shortcomings, in our two consecutive studies [25, 26], we proposed a methodology for automation of the design pattern classification and selections. We employed the proposed method via an unsupervised learning technique namely Fuzzy c-means and make it functional by presenting three case studies. The difference of our proposed method from the existing efforts has been summarized in Table 2.

Table 2. The main attributes of related work in the context of patterns classification and selection (The symbol – in table means that the attribute is not reported in the corresponding study)

	Focus on Pattern Organization	Focus on Pattern Selection	Approach to employ the proposed method	Limited to the Context of a Catalog	Pattern Sample Size	Design Problems Sample Size	Problem or Solution Domain
Zimmer [38]	Yes	No	UML	Yes	20	-	Solution
Kim and Han [39]	Yes	No	UML	Yes	30	-	Solution
Kim and Khawand [17]	No	Yes	UML	Yes	4	4	Solution
Hsueh et al [18]	Yes	Yes	Ontology	Yes	23	23	Solution
Khoury et al [23]	No	Yes	Ontology	Yes	-	-	Solution
Blomqvist [22]	No	Yes	Ontology	Yes	-	-	Problem
Hasheminejad and Jalili [24]	No	Yes	Text Categorization	No	103	50	Problem
Hussain et al [26]	Yes	Yes	Text Categorization	No	80	80	Problem
Velasco-Elizondo [11]	No	Yes	Text Categorization	Yes	-	-	Problem

We conduct this study to evaluate the efficacy of proposed method [24, 25] in terms of increasing the number of unsupervised learners, sample size, and sparsity, and generalized the finding.

3. OVERVIEW OF DESIGN PATTERNS AND TEXT CATEGORIZATION

3.1. Software Design Patterns

The developer and researcher community of each domain usually targets the compiler and other developers of the same community when they describe and categorized the patterns with respect to different aspects. For example, in the field of Object-Oriented (OO) development, Gamma et al [3] describe 23 design patterns and categorized them into three categories such as structural, creational, and behavioral depends on the relationship between classes and objects. The description of the pattern structure depends on 1) pattern's domain, 2) developer's interest and knowledge, and 3) developers' awareness about the pattern catalog of the same domain. However, the pattern's structure of any domain can be looked in their roots and described in two broad sections called 1) Problem Definition Domain, and 2) Solution Domain. The Problem Definition Domain is similar to the context and problem description sections, while the solution Domain section is related solution section of Alexandar's given template [45]. The authors introduce new formal notations or barrow to describe the solution domain of catalog patterns are managed using author's own formal notations [12] or borrow techniques [3]. Such as, in case of GoF and Service Oriented Architecture (SOA)[3] pattern catalog, authors borrows the notations of UML and Data Flow diagrams (DFD). The illustration of Bridge (GoF patterns) is revealed in Table 3. The automated systems or tools which are implemented using the solution domain of patterns are specific to the context of catalog's patterns [17, 38, 39, 43], and the inclusion of new patterns in the existing catalog might be challenged in terms of time and efforts [11]. Recently, in an online survey entitled "Trends and Development of Object-Oriented Design Patterns"[4], 75% of respondents (i.e. Developers) reported the use of problem context when they look for a design pattern. Consequently, in our proposed framework, we target the problem domain rather than the solution domain of design patterns.

3.2. Text Categorization Fundamentals

Supervised and unsupervised learning methods are employed for text categorization in the context of text mining domain. In the case of supervised learning methods, text categorization is performed with respect to class labels assigned to documents. However, in the case of unsupervised learning methods, data, and dis(similarity) measures are used. Text preprocessing is considered as a prerequisite of text categorization [46]. The main steps are as follows.

3.2.1. Preprocessing

In the first step, the removal of stop words and word stemming activities are performed. The aims of both activities are 1) to remove those words which carry no information (though used many times, such as a preposition or an article), and 2) make groups with respect to their conceptual meanings. The Porter's algorithm (consist of a set of rules) [47] is a widely used stemming algorithm [25, 27], which aid to find the stems of English words in the target vocabulary.

3.2.2. Indexing

In the second step, VSM (Vector Space Model) is used as a widely used indexing technique which formulates the documents. Subsequently, the term-document, word-context, and pair-pattern are three main types of VSM, however, term-document is widely used [48]. In VSM, the documents are defined via a vector of words. Subsequently, a matrix D is referred to a word-by-document and used to describe a document collection. The entry of D described the occurrence of a word in the corresponding document.

$$D = W_{wd} \tag{1}$$

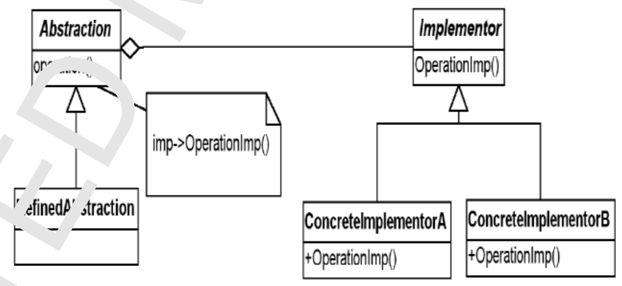
[3]http://soapatterns.org/

[4]http://www.my3q.com/research/cghmichelle/37053.phtml

Where W_{wd} is the weight of word w in document d. The weight W_{wd} of a word w can be determined in numerous ways. For example, in text categorization, the TF (Term Frequency), Binary, Entropy, TFC (Term Frequency Collection), LTC(Length Term Collection), and TFIDF (Term Frequency Inverse Document Frequency) are widely used [46]. The aim of these weighting methods is to reduced the noise from the documents collection and increase the performance of classification decisions.

Table 3. Template of GoF Bridge Design Pattern

Name of Design Pattern	**Composite**
<u>Problem Domain</u>	
Intent	"Decouple an abstraction from its implementation so that the two can vary independently and Publish interface in an inheritance hierarchy.
Motivation	"Decompose the component's interface and implementation into orthogonal class hierarchies. The interface class contains a pointer to the abstract implementation class. This pointer is initialized with an instance of a concrete implementation class, but all subsequent interaction from the interface class to the implementation class is limited to the abstraction maintained in the implementation base class. The client interacts with the interface class, and it in turn delegates all requests to the implementation class [3]."
Applicability	"The Bridge pattern decouples an abstraction from its implementation, so that the two can vary independently."
<u>Solution Domain</u>	

Structure	 Abstraction
	operation()
	Implementor
	OperationImp()
	imp->OperationImp()
	RefinedAbstraction
	ConcreteImplementorA
	+OperationImp()
	ConcreteImplementorB
	+OperationImp()

Participant Collaborations	Abstraction, Implementor, ConcreteImplementor
Consequences	The Bridge pattern • "decoupling the object's interface" • "improved extensibility (you can extend (i.e. subclass) the abstraction and implementation hierarchies independently)," • "hiding details from clients."
Implementation	• "Encapsulate the implementations in an abstract class." • "Contain a handle to it in the base class of the abstraction being implemented."
Related Patterns	• "The structure of State and Bridge are identical (except that Bridge admits hierarchies of envelope classes, whereas State allows only one)." • "State, Strategy, Bridge (and to some degree Adapter) have similar solution structures. They all share elements of the handle/body idiom."

3.2.3. Feature Selection

In the third step, numerous feature selection methods are used to reduce the number of features and construct a representative feature set, however, their discriminative power varies which might affect classification decisions [44]. Three feature selection approaches namely filter, wrapper, and embedded are used to construct a more illustrative feature set. In the context of text categorization, filter-based methods are widely used due to simplicity, efficiency, and low running costs, such as Mutual Information (MI), Document Frequency (DF), Chi-Square (CHI), Information Gain (IG), and Correlation Coefficient (CC) [46, 49].

4. OVERVIEW OF THE PROPOSED FRAMEWORK

An increase in the electronic documents is highly related with the fats development of software design patterns. Consequently, patterns organization is required to facilitate the software developers. This situation increases the importance for patterns organization with respect to text classification. Text categorization has been employed in certain domains such as spam email filtering [27], web page classification [28], sentiment analysis [29] and author identification [30]. In this regard, we consider the organization and selection of software design pattern problem as an IR (Information Retrieval) problem [50] and propose a framework. The aims of the proposed framework are 1) to use of unsupervised learner for patterns organization, 2) Selection of right pattern class for a real design problem via unsupervised learner, and 3) to recommend right pattern(s) for a given real design problem using different similarity options. Text classification steps are applied on the Problem domain part (Table 3) of design patterns and the design problem. The four main steps of our proposed framework are.

4.1 Preprocessing

Preprocessing is the first phase of the proposed framework and is applied to problem description of all design patterns and the design problem. Figure 2 depicts the set of preprocessing activities. The aim of preprocessing activities is to construct a more representative Vector Space Model (VSM). The first activity namely removal of stop words is performed to decrease the noise in textual data and reduce the feature space for the machine learning methods to produce the correct results. The term stop word is coined for those words in a natural language which are frequently used and are meaningless in joining of words in a sentence. For example, the pronouns, conjunctions, and prepositions in a sentence which carries no information and considered insignificant for certain natural language processing based applications such as information retrieval, text categorization, text summarization, and clustering. Word stemming activity is performed to remove the various suffixes (such as –ing, -ed, -able, -s, -er, -en, -est, –t, and so on) and find the root or stem of the words. Such as, the words “present”, “Presented”, “Presenting”, and “presentable” can be stemmed into a word “Present”. Like “Removal of stop words”, this activity is also used to reduce the numeral of words and reduce the feature and memory space. There are many text mining tools such as JPreText [51] and APIs[5] (Application Programming Interfaces) which can be used to perform these tasks. The indexing activity is performed to present a document as a vector of words (or a point in a space) and documents collection as a Vector Space Model (VSM). Subsequently, it is assumed that vectors (i.e. Points) will be semantically similar when they are close together otherwise they will be apart semantically. Feature vectors are constructed for description design problems and pattern’s problem domain.

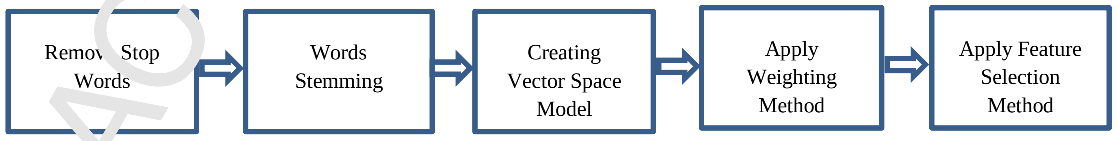


Figure 2. Steps of preprocessing phase

The aim of third activity that is indexing is to construct VSM(Vector Space Model) by including non-repeated terms of catalog documents, which are design patterns and problems in our case. The aim of last two activities

[5]http://www.opencalais.com/opencalais-demo/

(i.e. Weighting and feature selection methods) is analyze the improvement for the efficacy of proposed framework. The equation 2 is used to describe the structure of VSM (Vector Space Model) with N design patterns besides a design problem and M terms.

The weighting method activity is performed to find the weight ($W_{w_i d}$) of a word w_i in document d. The frequency of the word w_i in document d and collection c of N documents is referred as $t_{w_i d}$ and $tf_{w_i c}$. There are numerous ways to define the weight $W_{w_i d}$. For example, in text categorization, the TF (Term Frequency), Binary, Entropy, TFC (Term Frequency Collection), LTC (Length Term Collection) and TFIDF (Term Frequency Inverse Document Frequency) are widely used [46]. In text classification based applications, removal of stop words and word indexing aid to reduce the number of unimportant and meaningless words. The high dimensional feature space (computed in terms of $y = f(x)$ and need to mapped in n-dimensional x_n, $(n=1, \dots, N)$) is still a key issue for the performance of classifiers. The filters, wrappers, and embedded are three common approaches for feature selection [46, 49].

Create	Structure	Support	Client	Abstract	- - -	Class
1	1	0	1	0	- - -	1
1	0	1	0	0	- - -	0
0	1	0	1	0	- - -	1
-	-	-	-	-	- - -	-
0	1	1	0	1	- - -	1

Figure 3 Overview of the Vector Space Model (VSM)

4.2 Unsupervised learning for Design Patterns

The objective function of proposed framework is to classification of software design patterns with respect their appropriate labels and pattern selection without requiring enough knowledge. Since numerous unsupervised learning techniques have been employed. Consequently, deciding outperform unsupervised learner is still debatable [52]. The Keans and medoids (partitioning), Divisive and Agglomerative (hierarchical), Neural Network (model-based), Decision Tree, Fuzzy c-means (Soft Computing), DBSCAN, AUTOCLASS (Density based) and grid-based are widely knows learning techniques [46].
In order to fulfill our contribution, we assess several unsupervised learning techniques in the domain of our problem and recommend the best one which is more compatible with our problem domain. In this study, we consider five common unsupervised learning techniques have been reported in the text mining literature [46], which are Fuzzy c-means, K-means, Agglomerative, Partitioning Around Medoids (a.k.a. K-medoids with distance measures Euclidean and Manhattan) [32, 33, 34]. The detail description unsupervised learners which are employed in the context of the proposed framework is given in Appendix B. We used evaluation model describe in Section 5 to investigate the efficiency of learning techniques used in the proposed method.

$$VSM = \sum_{i=1}^{N} \sum_{j=1}^{M} w(T_{i,j}) \qquad (2)$$

4.3 Pattern Class Identification

The term design pattern class is coined for a group of design patterns which are suggested for a design problem in the proposed framework. Subsequently, the effectiveness of the proposed framework is assessed in terms of

unsupervised learner's decisions regarding the recommendation of a number of pattern classes. Such as, Figure 1 describe the patterns classes for the GoF design pattern collection.
Subsequently, each block inside the rectangle (labeled as a Phase 2) referred as a design pattern class, while + and – symbols depict the decision of the corresponding learning technique. The + symbol describes the closeness of a design problem with patterns (i.e. Members) of a candidate design pattern class (i.e. Cluster). For example, for Design Problem 1 (Section 5.4.1), structural pattern class is recommended as shown in Appendix A.

4.4 Suggestion and Selection of Design Patterns

The list of design patterns is suggested from a candidate design pattern class identified by a learning technique. Numerous similarity measures such as Extended Jaccard, Cosine, Dice Coefficient, and Pearson Correlation are used [53]. However, CS(Cosine Similarity) is the well-known measure which can assess the similarity between two vectors regardless of their documents length. The equation 3 and 4 are formulized to compute the correlation between documents namely design problems and suggested patterns.

$$CS_i = \sum_{j=1}^{N} w(Pattern_i, t_j) \times w(Problem, t_j) \tag{3}$$

$$W(d) = (w(d, t_1), w(d, t_2), \ldots, w(d, t_n)) \tag{4}$$

Where the terms $Pattern_i$ and Problem (equation 3) refered as ^{i}th pattern and problem respectively. The equation 4 is used to describe the weighted vector of ^{i}th pattern and design problem. Subsequently, equation 5-7 are computed for the suggestion of appropriate design pattern(s) to the developers just after finding the correlation via equation 3.

4.4.1 Right design pattern suggestion

In equation 5, arg max function is recommended and applied on CS values to suggest a right (K^{th}) design pattern for a developer.

$$K = \arg \max_i CS_i \tag{5}$$

4.4.2 Threshold Θ_1 based design pattern suggestion

In equation 6, a threshold Θ_1 is empirically recommended for the suggestion of zero, one or more than one pattern to a developer.

$$|CS_i| > \Theta_1 \tag{6}$$

4.4.3 Threshold Θ_2 based design pattern suggestion

In equation 7, a threshold Θ_2 is empirically recommended for the suggestion of design pattern(s) with their CS_i value more close to the best one with CS_{max} value to a developer.

$$|CS_{max} - CS_i| \le \Theta_2 \tag{7}$$

The Θ_1 and Θ_2 values can be obtained from the experiments performed to evaluate the efficacy of framework [24, 54]. For example, in our case, on sample of 9 design problems (Section 5.4), the range of Θ_1 and Θ_2 threshold is determined on the base of the highest value of RCD (Ratio of Corrected Design patterns). The detail description Θ_1 and Θ_2 is given in the pseudocode-3 of the Section 6 and Section 7.3.

5. EVALUATION MODEL FOR THE PROPOSED FRAMEWORK

We describe an evaluation model (for Figure 1) which comprise of three steps. In the first step, we recommend the measure to assess the efficacy of unsupervised learning techniques of the framework for identification of pattern class. In the second step, we recommend similarity measure and option for the suggestion of apposite design pattern(s). In the third step, we performed few case studies which consist of three pattern collections and 103 design problems.

5.1 Evaluation of Unsupervised Learning techniques

Like supervised learners evaluation, precision and recall measure can be used to assess the effectiveness of unsupervised learners. However, we need to assume that members (i.e. design patterns) of the same class are more relevant as compared to rest of members, and relevancy can be determined using similarity measures. Such as V-measure, MI (Mutual Information), and ARI (Adjusted Rand Index) measure needed ground reality information about members to assess the effectiveness of unsupervised learners [55]. Similarly, there are the certain measure which did not require ground reality truth regarding classes of members such as Silhouette Coefficient measure [56] to assess the unsupervised learner's performance. The unsupervised learner with the highest value of any one of these similarity matrices is considered as the best classifier. Though, we assess the unsupervised learning technique with certain weighting methods (Section 4.1). But, we compare the effectiveness of unsupervised learner with their best weighting method. Firstly, to determine the finest weighting method of each unsupervised learner (in each case study), we recommend a set of micro-averaging measures (equation 9-10), namely Precision(P), Recall(R), and F-measure [31, 49]. The aim of these measures is to figure out the best weighting method. For example, the finest weighting method for an unsupervised learner is decided on the bases of utmost F-measure value.

$$P = \frac{\sum_{c=1}^{N} TP_c}{\sum_{c=1}^{N} (TP_c + FP_c)}, \quad \text{Micro-Averaging} \tag{8}$$

$$R = \frac{\sum_{c=1}^{N} TP_c}{\sum_{c=1}^{N} (TP_c + FN_c)}, \quad \text{Micro-Averaging} \tag{9}$$

$$F = \frac{2 \times P \times R}{(P+R)} \tag{10}$$

Where the term N (equation 8 and 9) refered to the number of classes required to accommodate the design patterns. Such as, three classes (i.e. N=3) are required to accommodate the 23 GoF design patterns. Subsequently, the terms TP (True Positive), FP (False Positive), and FN(False Negative) describe the number of correctly identified, incorrectly identified, and missing design patterns for each class.
Secondly, to select the outperform unsupervised learner with their best weighting method, we recommend Adjusted Random Index (ARI) show in equations 11 and 12.

$$ARI = \frac{RandomIndex - E[RandomIndex]}{\max(RandomIndex) - E[RandomIndex]} \tag{11}$$

$$RandomIndex = \frac{P_a + P_b}{Cl_2^{n_{samples}}} \tag{12}$$

Where the terms CL and K referred to class labels based on ground truth and number of classes recommended by an unsupervised learner. Suppose P_1 and P_2 are two partitions consist of n_1 and n_2 elements, and n total elements. For example, the value of n is 23 for GoF design pattern collection. We also suppose that two disjunctive tables (N_1 and N_2) and a contingency table (T) with n_{ij} elements. We calculate the RI (Random Index) for each partition P which can characterized by $n \times n$ paired comparison and help to compute the pair agreements. The total pair of the agreement is computed in terms of same and different pairs in P_1 and P_2. While the total pair of discordance is computed in terms of same and different pairs either in P_1 or P_2. Finally, the terms P_a and P_b reffered to the number of pairs which are recognized in same and different set of CL and K respectively [60, 61].

Finally, to assess the efficacy of framework, we used certain feature selection methods such as Chi-Square, IG (Information Gain), DF(Document Frequency), and Correlation.

5.2 Evaluation process for Design Pattern Selection

Though we have recommend the similarity measure and options to assess the correlation between a design problem and suggested design patterns. However, in this section, we recommend RCD (<u>R</u>atio of <u>C</u>orrected <u>D</u>esign patterns) measure the select the best similarity option. The equation 13 is used to compute the ratio of numbers referred as right and total suggested design patterns.

$$RCD = \frac{Number\ of\ Right\ Suggested\ Design\ Pattern}{Total\ Suggested\ Design\ Patterns} \tag{13}$$

5.3 Design Pattern Collections

The domain of patterns, corresponding community interest, the number of pattern classes, and design patterns in each class are the core factors behind the rationale for the selection of design pattern collection. The description of the selected design pattern collection is as follows.

5.3.1 GoF Pattern Collection

In the context of Object-Oriented(OO) development, 23 GoF design pattern is published and categorized in three classes namely Structural, Creational, and Behavioral. After performing first three Preprocessing activities, 1465 words (non-repeated) are obtained. However, the number of terms is reduced by applying rest of preprocessing activities [3].

5.3.2 Douglass Pattern Collection

In the context of real time applications, 34 Douglass design pattern is published and categorized in five classes namely Memory, Resource, Safety and Reliability, Concurrency, and Distribution. After performing first three Preprocessing activities, 1271 words (non-repeated) are obtained. However, the number of terms is reduced by applying rest of preprocessing activities [35].

5.3.3 Security Pattern Collection

In the context of security applications, 46 Douglass design pattern is published and categorized in eight classes namely ACM (Access Control Model), SACA (System Access & Control Architecture), OSAC (Operating System Access Control), IA (Identification & Authentication), Accounting, ESRM (Enterprise Security & Risk Management), SIA (Security Internet Application), and FA (Firewall Architecture). After performing first three Preprocessing activities, 1230 words (non-repeated) are obtained. However, the number of terms is reduced by applying rest of preprocessing activities [37].

5.4 Real Design Problems

We asses the performance of the framework with 103 design problems which are retrieved from certain resources [8, 57, 58, 59]. The authenticity of the framework is evaluated with design problems which are mapped with respect expert's opinion shown in Table 4.

Table 4. Design problems resource information

Reference	**Authors**	**Participants**	**Pattern's Collection**
Bouhour et al. [14]	C. Bouhour, C. Percebois,and H. Leblance	126	GoF patterns
Silberschatz et al. [57]	A. Silberschatz, G. Gagne, and P.B. Galvin	3	Security Patterns
Tanenbaum et al. [58]	A.S Tanenbaum and H. Bos	2	Real Time Patterns
Shalloway et al. [59]	A. Shalloway and R. Trott	3	GoF patterns
Shvets [62]	A. Shvets	1	GoF patterns

In the following sub-section, we give the sample of 9 design problems. From each design pattern collection(Section 5.3), we collect three design problems.

5.4.1 GoF Patterns related problems

Design Problem 1: "Design a drawing editor. A design is composed of te graphics (lines, rectangles and roses), positioned at precise positions. Each graphic form must be modeled by a class that provides a method draw(): void. A rose is a complex graphic designed by a black-box class component. This component performs this drawing in memory, and provides access through a method getRose(): int that returns the address of the drawing. It is probable that the system evolves in order to draw circles."

Design Problem 2: "Design a DVD market place work. The DVD marketplace provides DVD to its clients with three categories: children, normal and new. A DVD is new during some weeks, and after change category. The DVD price depends on the category. It is probable that the system evolves in order to take into account the horror category."

Design Problem 3: "Many distinct and unrelated operations need to be performed on node objects in a heterogeneous aggregate structure. You want to avoid 'polluting' the node classes with these operations. And, you don't want to have to query the type of each node and cast the pointer to the appropriate type before performing the desired operation."

5.4.2 Douglass Pattern related problems

Design Problem 4: "One of the key problems with dynamic memory allocation is memory fragmentation. Memory fragmentation is the random intermixing of free and allocated memory in the heap. For memory fragmentation to occur, the two conditions must be met 1) the order of memory allocation is unrelated to the order in which it is released, and 2) Memory is allocated in various sizes from the heap".

Design Problem 5: "The problem of deadlock is such a serious one in highly reliable computing that many systems design in specific mechanisms to detect it or avoid it. As previously discussed, deadlock occurs when a task is waiting on a condition that can never, in principle, be satisfied. There are four conditions that must be true for deadlock to occur, and it is sufficient to deny the existence of any one of these."

Design Problem 6: "A distributed system using the mailboxes has two IPC primitives, send and receive . The latter primitive specifics a process to receive from and blocks if no message from that process is available, even though message may be waiting from other process. Is deadlock possible, if there are no shared resources, but process need to communicate frequently"

5.4.3 Security Pattern related problems

Design Problem 7: “We need to have a way to control access to resources, including information. The first step is to declare who is authorized to access resources in specific ways. Otherwise, any active entity (user, process) could access any resource and we could have confidentiality and integrity problems. How do we describe who is authorized to access specific resources in a system?”
Design Problem 8: “For convenient administration of authorization rights we need to have ways to factor out rights. Otherwise, the number of individual rights is just too large, and granting rights to individual users would require storing many authorization rules, and it would be hard for administrators to keep track of these rules. How do we assign rights based on the functions or tasks of people?”
Design Problem 9: “The ability to define an asset’s value is a key component of any risk assessment. Threats and vulnerabilities that target and expose an asset are only significant within the context of the asset’s value. Without this determination, an enterprise is unable to properly assess the risks posed to its assets. How can an enterprise determine the overall value of its assets?”

6. PSEUDOCODES FOR WORKING OF PROPOSED FRAMEWORK

6.1 Design Patterns Organization

Three pseudocode is given to describe the experimental procedures for achieving the target objectives. The detail of each pseudocode is given in [26]. The aim of the first pseudocode is to describe the procedure for design pattern organization. The aim of the second pseudocode is to describe the procedure how a design pattern class is suggested for a design problem. Finally, the aim of third pseudocode is to describe the procedure how right design pattern(s) are suggested.
The aim of pseudocode-1 is to describe in performing three activities. The first activity is related to the selection of finest weighting method for unsupervised learners given in Appendix B. First activity is related to the selected of an outperformed unsupervised learner. Finally, third activity is associated with the investigation of the effect of feature selection techniques on the efficacy of unsupervised learners.
The aim of pseudocode-2 is to describe the procedure that how design patterns could be organized with respect to a design problem and how the framework is effective in determination of right pattern classes.
The experimental results used to describe the usefulness of the framework to define the appropriate pattern class for the first sample of 9 design problems (out of 103) is shown in Appendix A. For example, the procedure of pseudocode-2 aid to determine the ‘Structure’ (a class of GoF pattern collection) pattern class for the DP-1(i.e. Design Problem 1) and relevant patterns (Adapter, Decorator, Bridge, Flyweight, Composite, Fascade, and Proxy) of the same class.

6.2 Design Pattern Selection

In section 6.1, we have described the process for the organization of design patterns with text relevancy to a given design problem and identify the pattern class. For example, in appendix A, the Structural (Table 5), Behavioral (Table 6), Behavioral (Table 7) are the pattern classes (i.e. Categories) for the DP1 (Design Problem 1), DP2 (Design Problem 2), and DP3 (Design Problem 3). The second objective of the proposed study is to recommend the right design pattern(s) from the pattern list of suggesting pattern class. For example, the Adapter, Composite, Bridge, Decorator, Flyweight, Fascade, and proxy patterns are included in the suggested pattern class (i.e. Structural) for Design Problem 1 (DP1). The selection of right design pattern(s) for a given design problem is described in the pseudocode-3. The pseudocode-3 describe the procedure for the selection of correct design pattern(s) (from the patterns of suggested pattern class through pseudocode-2) for a design problem. We present an example (Appendix C) to depict the implementation procedure for pseudocode which is designed to formulate the capacity of the proposed framework to achieve the target objectives.

7. DISCUSSION ON RESULTS

The proposed framework is assessed using measure recommended in the proposed evaluation model (Section 5).

We evaluate the efficacy of framework in three aspects; 1) the classification of design patterns (Section 4.2), 2) identification of appropriate design pattern class (Section 4.3), and 3) the selection of suitable design pattern(s) for a design problem (Section 4.4). Firstly, each unsupervised learner of the proposed framework is assessed with weighting methods in terms of F-measure and homogeneity and completeness of features is revealed explicitly. Secondly, each unsupervised learner with its finest weighting method is comparatively assessed in terms of ARI. An unsupervised learner with highest ARI value is considered as outperform learner.

7.1 Design Pattern Collection Evaluation

In order to evaluate the efficacy of the framework for the classification of design patterns with respect to expert opinion, we consider the well-known design pattern catalogs (Section 5.3) of different domains. Such as, in the case of Gang-of-Four (GoF) design patterns catalog, the efficacy of the proposed framework can be determined through the classification of 23 design patterns into the apposite number of pattern classes (i.e. Structural, Creational, and Behavioral). Since the proposed framework is exploited through the unsupervised learning techniques (Appendix B), so, the recommendation of outperforming unsupervised learner depends on the homogeneity and completeness of features. In order to evaluate the impact of homogeneity and completeness of features on the performance of unsupervised learners, we recommend two activities (i.e. Weighting and feature selection methods) in the preprocessing phase of the framework. The proposed evaluation model (Section 5.1) is applied to recommend the outperform unsupervised learner with best weighting (Binary, TFIDF, TFC, LTC, or Entropy) and feature selection (DF, IG, GI, GR, Correlation) method. The pseudocode-1 has been described in Section 6.1 to present the workflow of the proposed framework for the organization of patterns. The outcome of pseudocode-1 is to recommend the outperform unsupervised learner with finest weighting method and feature selection technique. The experimental results for the classification of GoF, Douglass, and Security design patterns (Section 5.3) are discussed as follows.

7.1.1 GoF Patterns Evaluation

The 23 GoF patterns are grouped into 3 classes. In terms of F-measure criterion, we find the finest weighting method for each unsupervised learner. In this regard, assessment results are shown in Figure 4-a.

The utmost F-measure value aid to recommend the finest weighting method for each corresponding unsupervised learner. For example, we find the utmost F-measure values (i.e. 0.73 and 0.78) for the TFIDF in case of Fuzzy c-means and PAM-Euclidean respectively, so, TFIDF is recommended as the finest weighting method for the Fuzzy c-means and PAM-Euclidean. The Binary weighting method is recommended for k-means and PAM-Manhattan due to highest the utmost F-measure values 0.70 and 0.65 respectively. Subsequently, the Entropy weighting method is recommended for the Agglomerative learner due to the utmost F-measure value that is 0.74.

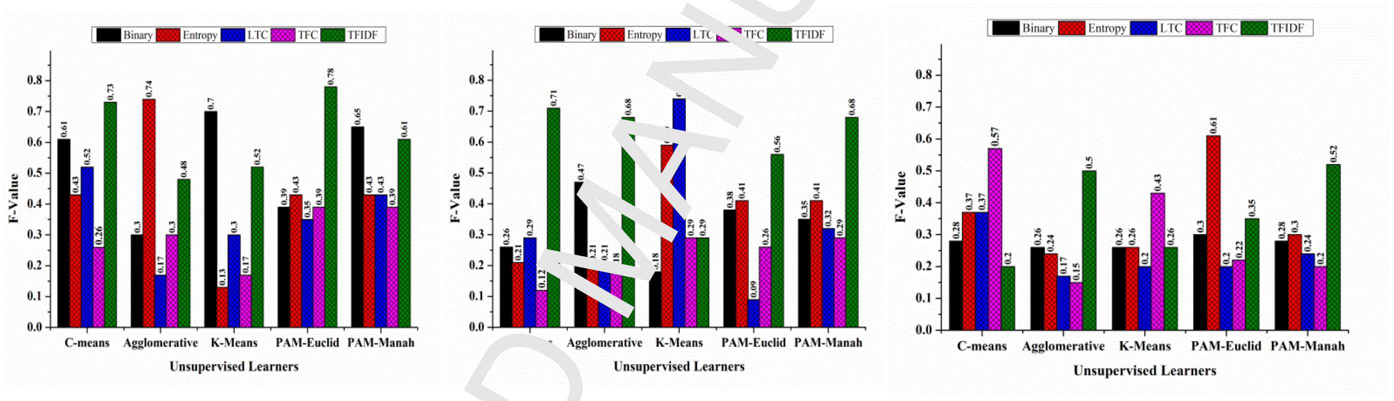
Subsequently, to select the outperform learner, all unsupervised learners with their finest weighting method (e.g. C-means with TFIDF) are comparatively investigated (in terms of ARI define in Section 5.1) by tuning with the different number of clusters (i.e. k). The experimental results are depicted in Figure 5-a. While Figure 5-a compares the evaluation results of five unsupervised learners (for classification of GoF patterns) with the certain number of clusters (i.e. Pattern class). However, the unsupervised learner (at k=3) with the highest ARI value is suggested as the best learning technique of the proposed framework in the context of GoF patterns. There are three main consequences which are retrieved from the experimental results.

- The performance of all unsupervised learners (in terms of ARI) remains better (with the minor difference) with the number of clusters (k=3) as compared to other numbers of clusters (k≠3), which show the applicability of the framework for the classification of the GoF design patterns.
- According to ARI criterion (in Figure 5-a), Fuzzy c-means obtain better results (i.e. ARI value is 0.92) than other unsupervised learning techniques.
- For the classification of the GoF patterns, the performance of outperforming unsupervised learner Fuzzy c-means (with ARI as 0.92) of the proposed framework is better than the outperform Vector Space Model (VSM) with Evaluation Metrics Fusion (EMF) as 0.73 [24]. Though there is a difference in the evaluating

criteria of both studies, however, the common focus of both studies is to perform the evaluation of proposed approaches on the base of ground truth.

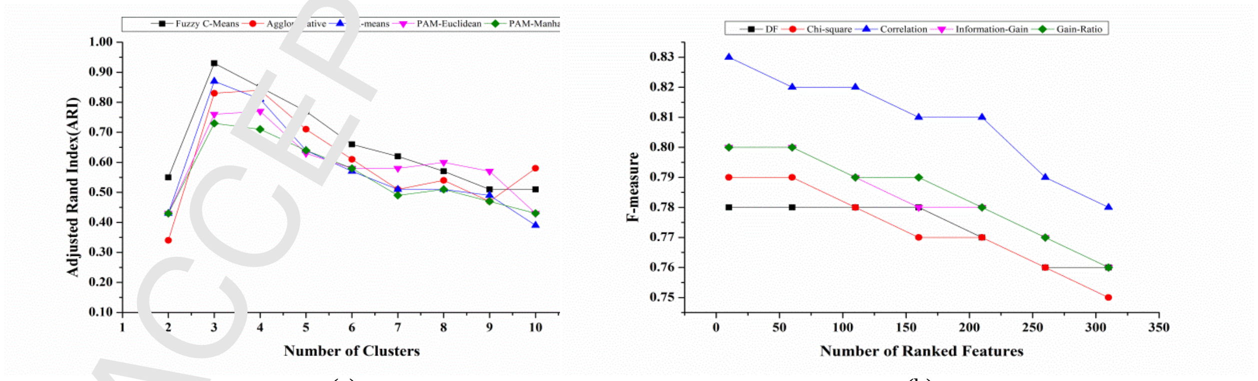
Moreover, we applied certain feature selection technique to construct a more illustrative feature set to investigate its effect on the effectiveness unsupervised learner in terms of ARI. Though, we have performed numerous experiments. However, we present the influence of feature selection techniques on the efficacy of outperformed learner namely Fuzzy c-measn with TFIDF(best weighting method) in Figure 5-b. The labels "Number of Rank Feature" is used "F-measure values" are used for x-axis and y-axis. The feature selection technique with utmost F-measure value(as compared to actual F-measure value 0.73) present its highest influence on the efficacy of framework and consider as best weighting method. There are two main consequences which are retrieved from the experimental results.

- In case of top-N (i.e. N=10) features which are ranked by each feature selection technique, we observe the average increase (i.e. 6.84% to 13.69% in terms of F-measure) in the performance of outperforming unsupervised learner namely Fuzzy c-means.
- We observed 13.69% increase in the efficacy of outperforming Fuzzy c-means with top-N (i.e N=10) ranked features via Correlation as the best method.



(a) (b) (c)

Figure 4. Best weighting methods for unsupervised learners
a) GoF patterns b) Douglass patterns c) Security patterns collection



(a) (b)

Figure 5. Evaluation results on GoF patterns
a) Unsupervised learners with their best weighting method b) Best unsupervised learner (Fuzzy c-means) with N rank features

7.1.2 Douglass Patterns Evaluation

The 34 Douglass patterns are grouped into 5 classes. In terms of F-measure criterion, we find the finest weighting method for each unsupervised learner. In this regard, assessment results are shown in Figure 4-b.

The utmost F-measure value aid to recommend the finest weighting method for each corresponding unsupervised learner. For example, we find the utmost F-measure values (i.e. 0.71, 0.68, 0.56,0.68) for the TFIDF in case of Fuzzy c-means, Agglomerative, PAM-Euclidean, and PAM-Manhattan respectively. Consequently, TFIDF is recommended as the finest weighting method for the Agglomerative, PAM-Euclidean, and PAM-Manhattan. Subsequently, the LTC weighting method is recommended for the k-means learner due to the utmost F-measure value that is 0.74.

Subsequently, to select the outperform learner, all unsupervised learners with their finest weighting method (e.g. K-means with LTC) are comparatively investigated (in terms of ARI define in Section 5.1) by tuning with the different number of clusters (i.e. k). The experimental results are depicted in Figure 6-a. While Figure 6-a compares the evaluation results of five unsupervised learners (for classification of Douglass patterns) with the certain number of clusters (i.e. Pattern class), however, the unsupervised learner (at k=5) with the highest ARI value is suggested as the best learning technique of the proposed framework in the context of Douglass patterns. There are two main consequences which are retrieved from the experimental results.

- The performance of all unsupervised learners (in terms of ARI) remains better (with the minor difference) with the number of clusters (k=5) as compared to other numbers of clusters (k≠5), which indicate the applicability of the framework for the organization of the Douglass design patterns.
- According to ARI criterion (in Figure 6-a), Fuzzy c-means and PAM-Euclidean (K-medoids) achieved better results (i.e. ARI value is 0.90 in rounded form) than other unsupervised learning techniques.
- For the classification of the Douglass design patterns, the performance of outperform unsupervised learner PAM-Euclidean (with ARI as 0.90) of the proposed framework is not better than the outperform Vector Space Model (VSM) and Naïve Bayes (NB) with Evaluation Metrics Fusion (EMF) as 1.0 [24], which is an ideal situation. Though there is a difference in the evaluating criteria of both studies, however, the common focus of both studies is to perform the evaluation of proposed approaches on the base of ground truth. We present the influence of feature selection techniques on the efficacy of outperformed learner namely PAM-Euclidean with TFIDF(best weighting method) in Figure 6-b. The labels “Number of Rank Feature” is used “F-measure values” are used for x-axis and y-axis respectively.

The feature selection technique with utmost F-measure value(as compared to actual F-measure value 0.56) present its highest influence on the effectiveness of the proposed framework and consider as best weighting method. There are two main consequences which are retrieved from the experimental results.

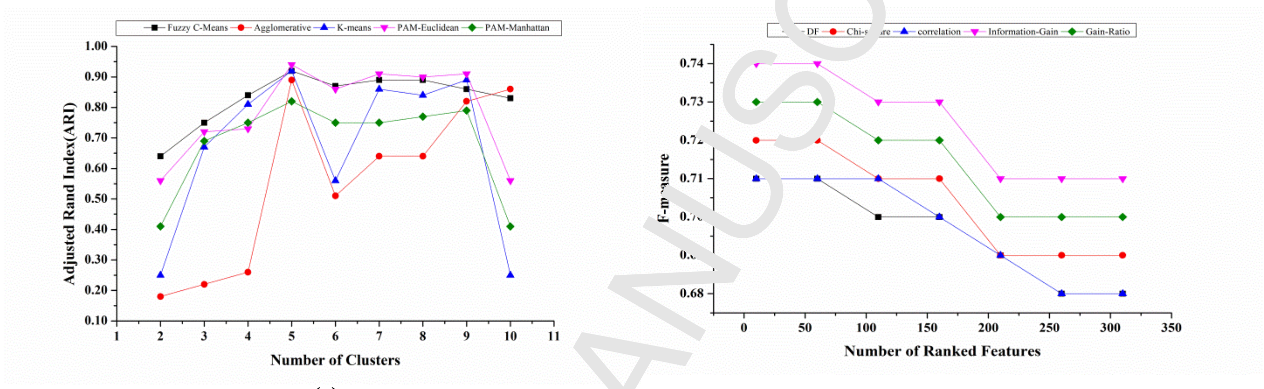
- In case of top-N (i.e. N=10) features which are ranked by each feature selection technique, we observe the average increase (i.e. 17.64% to 24.32% in terms of F-measure) in the effectiveness of outperforming unsupervised learner namely PAM-Euclidean.
- We observed 24.32% increase in the efficacy of outperforming PAM-Euclidean with top-N (i.e N=10) ranked features via Information Gain as the best method.

7.1.3 Security Patterns Evaluation

The 46 Douglass patterns are grouped into 8 classes. In terms of F-measure criterion, we find the finest weighting method for each unsupervised learner. In this regard, assessment results are shown in Figure 4-c.

The utmost F-measure value aid to recommend the finest weighting method for each corresponding unsupervised learner. For example, we find the utmost F-measure values (i.e. 0.50 and 0.53) for the TFIDF in case of Agglomerative and PAM-Manhattan respectively. Consequently, TFIDF is recommended as the finest weighting method for the Agglomerative and PAM-Manhattan. The TFC weighting method is recommended for Fuzzy c-means and k-means due to highest the utmost F-measure values 0.57 and 0.43 respectively. Subsequently, the Entropy weighting method is recommended for the PAM-Euclidean learner due to the utmost F-measure value that is 0.61.

Subsequently, in order to select the outperform learner, all unsupervised learners with their finest weighting method (e.g. PAM-Euclidean with Entropy) are comparatively investigated (in terms of ARI define in Section 5.1) by tuning with the different number of clusters (i.e. k). The experimental results are depicted in Figure 7-a. While Figure 7-a compares the evaluation results of five unsupervised learners (i.e. classification of Douglass patterns) with the certain number of clusters (i.e. Pattern class), however, the unsupervised learner (at k=8) with highest ARI value is suggested as the best learning technique of the proposed framework in the context of security patterns. There are two main consequences which are retrieved from the experimental results.



(a) (b)

Figure 6. Evaluation results on Douglass patterns
a) Unsupervised learners with their best weighting method b) Best unsupervised learner (PAM-Euclidean) with N rank features

- The performance of all unsupervised learners (in terms of ARI) remains better (with the minor difference) with the number of clusters (k=8) as compared to the other number of clusters (k≠8), which indicate the applicability of the framework for the organization of the Douglass design patterns.
- According to ARI criterion (in Figure 6-a), Fuzzy c-means achieved better results (i.e. ARI value is 0.60) than other unsupervised learning techniques.
- For the classification of the Security patterns, the performance of outperforming unsupervised learner Fuzzy c-means (with ARI as 0.90) of the proposed framework is better than the outperform Naïve Bayes (NB) with Evaluation Metrics Fusion (EMF) as 0.89 [24], with a minor difference. Though there is a difference in the evaluating criteria of both studies, however, the common focus of both studies is to perform the evaluation of proposed approaches on the base of ground truth.

We present the influence of feature selection techniques on the efficacy of outperformed learner namely Fuzzy c-means with TFC (best weighting method) in Figure 7-b. The labels "Number of Rank Feature" is used "F-measure values" are used for x-axis and y-axis respectively. The feature selection technique with utmost F-measure value(as compared to actual F-measure value 0.57) present its highest influence on the effectiveness of the proposed framework and consider as best weighting method. There are two main consequences which are retrieved from the experimental results.

- In case of top-N (i.e. N=10) features which are ranked by each feature selection technique, we observe the average increase (i.e. 5.26% to 10.52% in terms of F-measure) in the efficacy of outperforming unsupervised learner namely Fuzzy c-means.
- We observed 10.52% increase in the efficacy of outperforming PAM-Euclidean with top-N (i.e N=10) ranked features via Information Gain as the best method.

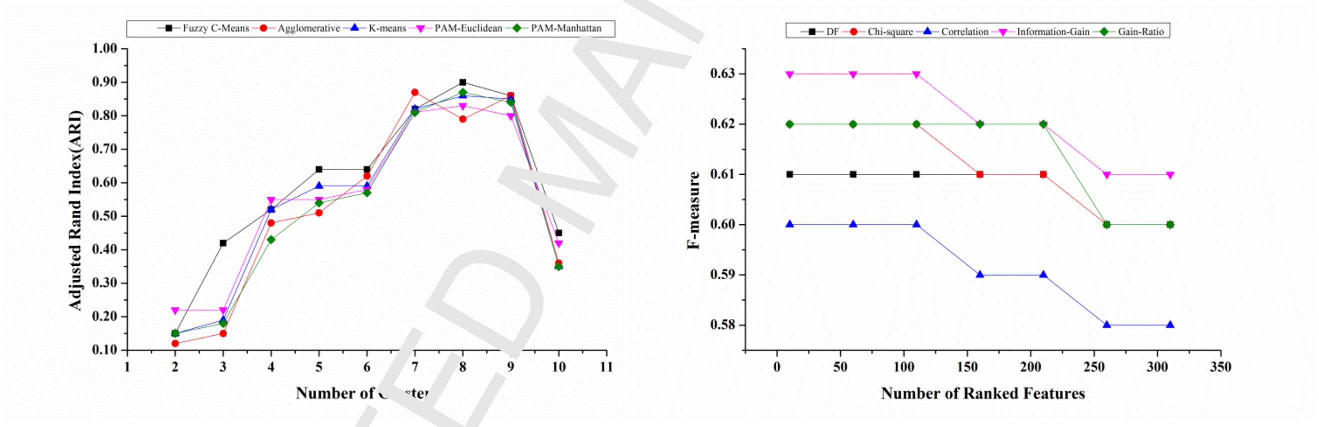
The information regarding outperformed unsupervised learner in terms of pattern's organization of each case study is shown in table 5.

Table 5. Case study wise summary of outperformed unsupervised learners

Design Pattern Collection	Unsupervised Learning Technique	Best Weighting Method	ARI value
GoF Patterns	Fuzzy c-means	TFIDF	0.92
Douglass Patterns	PAM-Euclidean	TFIDF	090
Security Patterns	Fuzzy c-means	TFC	0.90

7.2 Evaluation of Proposed framework for Design Problems

The efficacy of the proposed framework can be determined from the experimental results given in Section 7.1. Subsequently, to evaluate the efficacy of the framework to find the right design pattern class for the given sample of design problems, we follow the experimental procedure which is described through pseudocode-2. In this study, we discussed the experimental results with respect to three pattern collection and their corresponding samples of design problems.



(a) (b)

Figure 7. Evaluation results on security patterns
a) Unsupervised learners with their best weighting method b) Best unsupervised learner (Fuzzy c-means) with N rank features

7.2.1 Evaluation of GoF patterns related problems

We retrieved 31 real design problems from [14, 59, 62] (Section 5.4) resources in the context of GoF design pattern collection. The 3 out of 31 design problems are described in the section 5.4.1. We call pseudocode-2 with GoF design patterns (i.e. N=23) and set of 31 design problems. The value of k is set to 3 because GoF patterns have been classified into 3 categories according to expert opinion. Figure 8 depicts the comparative results of unsupervised learning techniques. The Fuzzy c-means and PAM-Euclidean with a minor difference attains better results than other unsupervised learning techniques (at k=3) in terms of ARI criterion. The ARI value 0.85 for the Fuzzy c-means learner (which is better than outperform VSM classifier with EWM as 0.80 with a sample of 19 design problems [24]) determined the capacity of the proposed framework to determine the apposite pattern class for 26 out of 31 design problems. Such as, in the case of design problems (section 5.4.1), the Fuzzy c-means and PAM-Euclidean determined the Structural, Behavioral and Behavioral design pattern class for the DP-1, DP-2 and DP-3. So, the results of the framework are with respect to expert opinion.

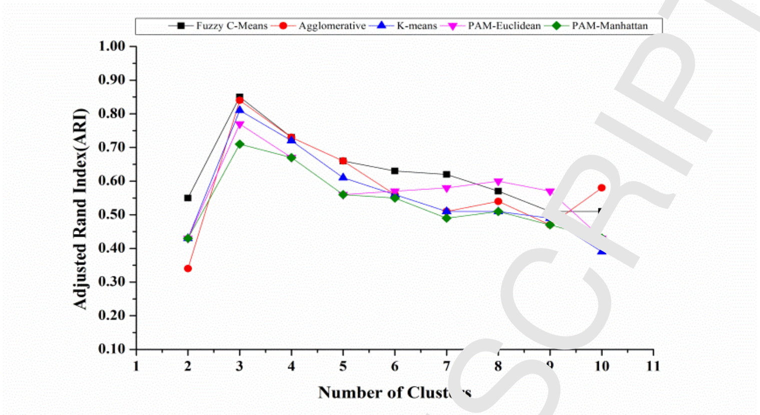


Figure 8. Evaluation results of unsupervised learners (with their best weighting methods) on GoF design problems and sample of 31 design problems.

7.2.2 Evaluation of Douglass patterns related problems

We retrieved 23 real design problems from [58, 59] (Section 5.4) resources in the context of Douglass design pattern collection. The 3 out of 23 design problems are described in the section 5.4.2. We call the pseudocode-2 with Douglass design patterns (i.e. N=34) and set of 23 design problems. The value of k is set to 5 because Douglass patterns have been classified into 5 categories according to expert opinion. Figure 9 depicts the comparative results of unsupervised learning techniques in the context to determine the right pattern classes for the given sample of Douglass related design problems. According to ARI criterion, the Fuzzy c-means and PAM-Euclidean with a minor difference attain better results than other unsupervised learning techniques (at k=5). The ARI value 0.87 for the PAM-Euclidean learner (which is better than outperform NB classifier with EWM as 0.71 with a sample of 19 design problems [24]) determined the capacity of the proposed framework to determine the appropriate design pattern class for 20 out of 23 design problems. Such as, in the case of design problems (section 5.4.2), the Fuzzy c-means and PAM-Euclidean determined the Memory, Resources and Distribution design pattern class for the DP-4, DP-5, and DP-6. So, the results of the framework are with respect to expert opinion.

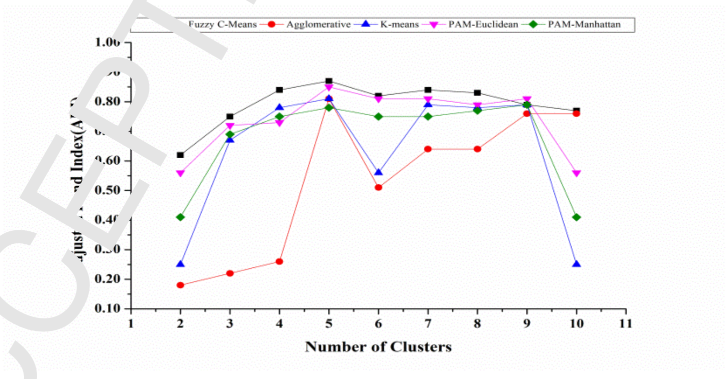


Figure 9. Evaluation results of unsupervised learners with their best weighting methods on Douglass design problems

7.2.3 Evaluation of Security patterns related problems

We retrieved 49 real design problems from [37, 58] (Section 5.4) resources in the context of security design pattern collection. The 3 out of 49 design problems are described in the section 5.4.3. We call the pseudocode-2 with Security design patterns (i.e. N=46) and set of 49 design problems. The value of k is set to 8 because Security patterns have been classified into 8 categories according to expert opinion. Figure 10 depicts the comparative results of unsupervised learning techniques in the context to determine the correct pattern classes for the given sample of related Security design problems. According to ARI criterion, the Fuzzy c-means and k-means with a minor difference attain better results than other unsupervised learning techniques (at k=8). The ARI value 0.80 for the PAM-Euclidean learner (better than outperform NB classifier with EWM as 0.64 with a sample of 19 design problems [24]) determined the capacity of the proposed framework to determine the appropriate design pattern class for 39 out of 49 design problems. Such as, in case of design problems (section 5.4.3), the Fuzzy c-means and k-means determine the Access Control Models(ACM), Accounting, and Enterprize Security and Risk Management (ESRM) design pattern class for the DP-7, DP-8, and DP-9. So, the results of the framework are with respect to expert opinion.

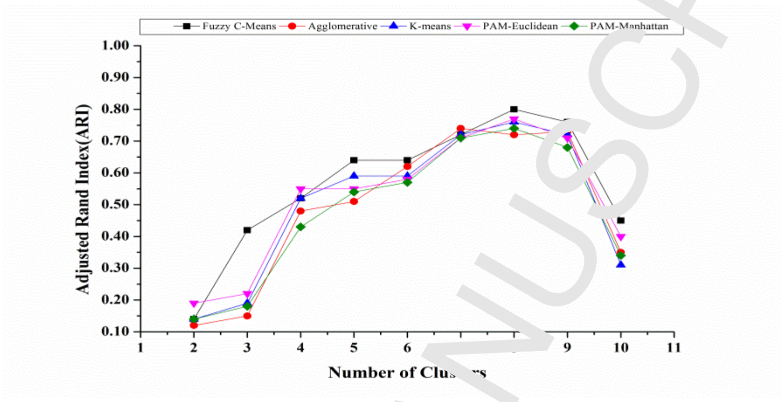


Figure 10. Evaluation results of unsupervised learners with their Best weighting methods on Security design problems

7.3 Design Pattern(s) Selection Evaluation

In the above sections 8.1 and 8.2, we have investigated the effectiveness of the proposed framework to classify/organize the patterns and to determine the right pattern class for the given design problems. In this section, we evaluate the efficacy of the framework in the perspective of pattern selection for the given design problems. Firstly, we used the evaluation criteria (Section 4.4) with the sample of 9 design problems described in Section 5.4, and secondly, we randomly select the sample of 30 design problems. It is assumed that the design problems of both samples have been categorized with the correct design pattern class. For example, in Appendix A, the correct pattern class for each design problem has been identified. We provide the analysis of 9 problems for pattern selection by applying the similarity options (Section 4.4). Moreover, we also randomly select a subset of 30 design problems and show the experimental work which aid to determine the best similarity options in terms of RCD (Equation 12). The pseudocode-3 has been described (Section 6.2) to present the scenario that how proposed framework can aid to recommend the design pattern(s) using different similarity options(Section 5.4). Firstly, the equation 2 is used to find the cosine similarities between each design problem and patterns of its recommended (through the proposed framework) class which is shown in appendix A (Tables 5-13).
The correct identification of right patterns indicates the effectiveness of the proposed framework in terms of highest RCD. Though, on the sample of 9 design problems (Section 5.4), similarity option (equation 4) with highest RCD value (i.e. Total right suggestions and total suggestion are 9) is recommended. However, we need to compute the equation 5 and 6, which can aid to find the ranges for the parameters Θ_1 and Θ_2 on the base of RCD values (equation 12) for each design problem. We analyzed each design problem and obtained the threshold for Θ_1 and Θ_2 on the base of RCD values. We obtained the threshold Θ_1 and Θ_2 range values using the RCD, and the description of corresponding analysis is shown in Table 15.

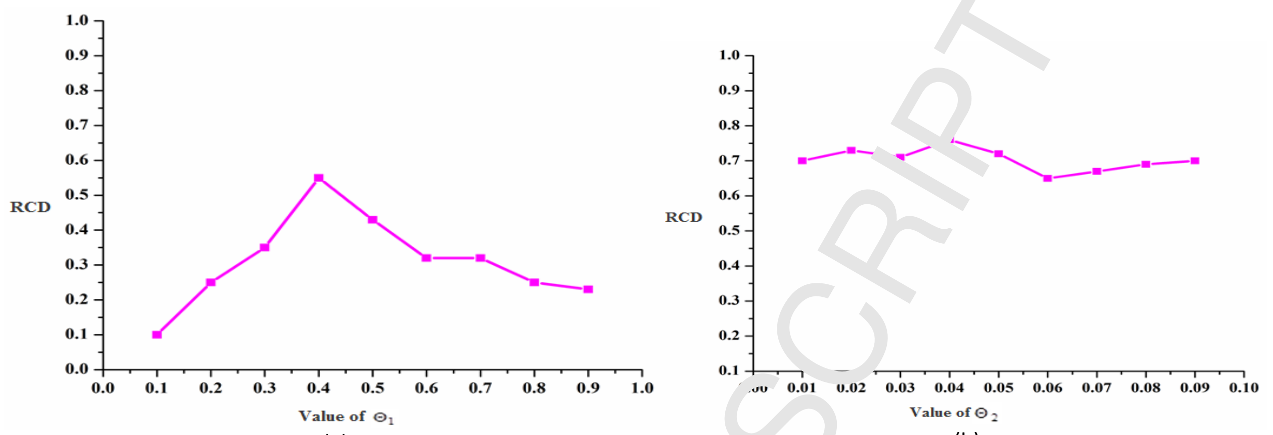
7.3.1 First Design Problems Sample (Total 9 Problems)

The pseudocode-3 is called for each design problem of the target sample and patterns of its identified class.mentioned in section 5.4. Selected patterns are highlighted using equation 4 and shown in Table 6-14.

Table 15. Suggested values of Θ_1 and Θ_2 parameters

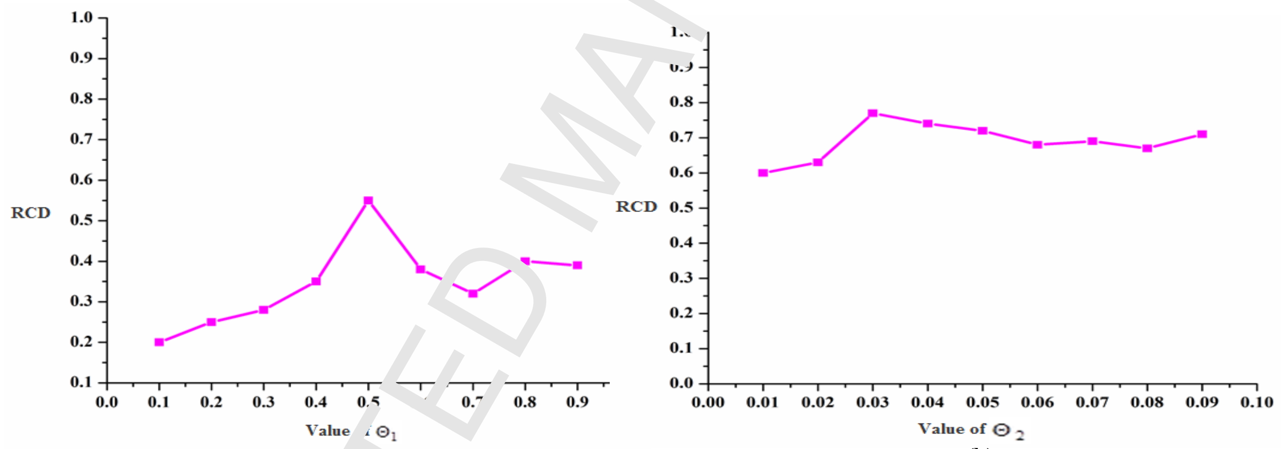
Design Problem	Θ_1	Θ_2	Analysis Description
Design Problem 1	0.36 to 0.51	0.0 to 0.16	The highest value of RCD can obtained when the values of Θ_1 (0.34, 0.51) and Θ_2 (0.0, 0.16) are in define range.
Design Problem 2	0.53 to 0.63	0.0 to 0.10	The highest value of RCD can obtained when the values of Θ_1 (0.53, 0.63) and Θ_2 (0.0, 0.10) are in define range.
Design Problem 3	0.56 to 0.71	0.0 to 0.16	The highest value of RCD can obtained when the values of Θ_1 (0.56, 0.71) and Θ_2 (0.0, 0.16) are in define range.
Design Problem 4	0.47 to 0.66	0.0 to 0.19	The highest value of RCD can obtained when the values of Θ_1 (0.47, 0.66) and Θ_2 (0.0, 0.19) are in define range.
Design Problem 5	0.42 to 0.66	0.0 to 0.24	The highest value of RCD can obtained when the values of Θ_1 (0.42, 0.66) and Θ_2 (0.0, 0.24) are in define range.
Design Problem 6	0.47 to 0.68	0.0 to 0.22	The highest value of RCD can obtained when the values of Θ_1 (0.47, 0.68) and Θ_2 (0.0, 0.22) are in define range.
Design Problem 7	0.45 to 0.60	0.0 to 0.15	The highest value of RCD can obtained when the values of Θ_1 (0.45, 0.60) and Θ_2 (0.0, 0.15) are in define range.
Design Problem 8	0.31 to 0.68	0.0 to 0.38	The highest value of RCD can obtained when the values of Θ_1 (0.31, 0.68) and Θ_2 (0.0, 0.38) are in define range.
Design Problem 9	0.41 to 0.64	0.0 to 0.24	The highest value of RCD can obtained when the values of Θ_1 (0.41, 0.64) and Θ_2 (0.0, 0.25) are in define range.

In order to evaluate the efficacy of the framework for design pattern selection, we also depict (Figure 11(a-b)) the relationship between the average highest RCD (for the sample of nine designated real design problems) and parameters Θ_1 and Θ_2 for the similarity options (section 4.4). The result of Figure 11-a indicates that the average highest value of RCD (0.75) is attained when Θ_1 is considered at 0.4. Subsequently, the result of Figure 11-b indicates that the average highest value of RCD (0.76) is attained when Θ_2 is considered at 0.04. Therefore, on the sample of 9 design problems, the values 0.4 and 0.04 can be recommended for Θ_1 (for the first similarity option) and Θ_2 (for the second similarity option). It is important to note that when the value of Θ_2 is considered 0, the first similarity option (equation 2) achieved the highest RCD value.



(a) (b)

Figure 11. First Sample of 9 design problems
a) Mean RCD for second similarity option (Equation 5) b) Mean RCD for third similarity option (Equation 6)



(a) (b)

Figure 12. First Sample of 30 design problems
a) Mean RCD for first similarity option (Equation 5) b) Mean RCD for second similarity option (Equation 6)

7.3.2 Second Design Problems Sample (Total 30 Problems)

Though in the section 7.3.1 we have evaluated the capacity of the proposed framework to select the correct design patterns for the first sample of nine design problems given in the section 5.4. However, in this section, we consider a sample of randomly selected 30 design problems from different resources [4, 24, 37, 38], and assess the efficacy of the framework. The pseudocode-3 is called for each design problem of target sample and the patterns of its corresponding class which is determined through the proposed framework. We depict (Figure 12(a-b)) the relationship between the average highest RCD (Second sample) and parameters Θ_1 and Θ_2 for the similarity options (section 4.4). The result of Figure 12-a indicates that the average highest value of RCD (0.55) is attained when Θ_1 is considered at 0.5. Subsequently, the result of Figure 12-b indicates that the average highest value of RCD (0.77) is attained when Θ_2 is considered at 0.03. Therefore,on the first sample of design problems, the values 0.5 and 0.03 can be recommended for Θ_1 (for the first similarity option) and Θ_2 (for the second similarity option). It is important to note that when the value of Θ_2 is considered 0, the first similarity option (equation 2) achieves the highest RCD value that is 0.82.

8. SUMMARY OF EVALUATION OF RESULTS

In this study, we have set certain objectives. The first objective is related to the classification of design patterns according to expert opinions. The second objective is related to selection of right design problems. Finally, the third object is related to the selection of appropriate design patterns. The proposed framework is functional in three phases, namely preprocessing, pattern's organization, and selection of design patterns. The design pattern catalog and related design problems are inputs of the proposed framework employed via unsupervised learning techniques. A set of performance measure is recommended in proposed evaluation model.

- We applied five unsupervised learning techniques in the proposed framework. In the evaluation results, we observe that the Fuzzy c-means and PAM-Euclidean achieve better results as compared to other techniques, to achieve the target objectives. However, Fuzzy c-means present better results in many cases.
- Since the proposed framework is employed through unsupervised learning techniques. For comparative performance evaluation, certain weighting methods are used. However, results indicate that we cannot recommend a single weighting method for all unsupervised learner. For example, TFIDF and TFC are recommended as finest weighting method for outperform unsupervised learner.
- Though, we pragmatic the highly influence of feature selection techniques on the effectiveness of unsupervised learners. However, we cannot suggest a single feature selection method. Such as, in case of outperform Fuzzy c-means learner, we observe its sensitivity more towards the Information Gain and Correlation.
- For each pattern collection and the sample of related design problems, we observed different values of Θ_1 and Θ_2 parameters. However, a developer can tune the values of these parameters by replicating the experiments.

9. LIMITATIONS OF THE STUDY

In our study, we have some limitations. The first limitation is related to external validity to generalize the results. We have reported the effectiveness of the proposed framework with five unsupervised learning techniques, three design patterns collection and 103 real design problems. The effectiveness of the proposed framework can be analyzed by considering other unsupervised learners, pattern collections and real design problem. Subsequently, the incorporation of the more unsupervised learner, design pattern collections, and real design problems can present the variation in the performance results of the proposed framework. Though, in the preprocessing phase of the proposed framework numerous weighting and feature selection methods are recommended to improve the performance of unsupervised learners. However, there are certain characteristics such as discriminative power of feature selection methods and sparsity in constructed VSM (Vector Space Model) for a design pattern collection may affect the results.

The second limitation is related to the internal validity of factors which could influence the results. We incorporate the unsupervised learners with default parameters. However, the calibration of parameters of unsupervised learners could improve the effectiveness of the proposed framework. For example, in case of Fuzzy c-means, m (i.e. Weighting component) and A (i.e. Weight Matrix) parameters are used to tune the objective function $J_m(X;U,V)=\sum_{t=1}^{c}\sum_{k=1}^{M}(U_{tk})^m\|x_k-v_t\|_A^2$. The value of m can be tuned to adjust the membership for the fuzziest state, however, $1.5 \le m \le 3.0$ is used to produce good results.

The third limitation is related to construction validity in terms of creation of Vector Space Model (VSM) for precise learning of unsupervised learners employed in the proposed approach. Though several experiments are performed to select the finest weighting and feature selection method, however, handling of multi-class problem can improve the effectiveness of the framework. In this work, selected feature techniques are biased towards their discriminative power. In order to improve the performance of unsupervised learners, a feature selection approach can be recommended to include the equal number of features with respect to each class into a constructed feature set.

10. CONCLUSION

The investigational results indicate that the proposed framework which is employed via text categorization approach and unsupervised learning techniques, give an encouraging base for the organization and selection of design pattern(s). In this context, researcher have made their efforts using Ontology, UML, and text categorization methodologies aims to decrease the computation time and preprocessing cost. However, formal specification, large sample size, and ground reality truth of class is required for precise learning are still issues with these automated system/tools. The aim of the proposed framework is to address these issues. We evaluate the proposed framework in the context of three design pattern groups and several real design problems.

The proposed framework is functional in three steps, Preprocessing, Organization of design pattern, and suggestion of a design pattern class via unsupervised learning techniques and Suggestion of Design Pattern(s) using different similarities options. The Fuzzy c-means, k-means, Agglomerative, Partition Around Medoids (PAM) with Euclidean and Manhattan distance (i.e. PAM-Euclidean and PAM-Manhattan) unsupervised learning techniques are used in the framework. An evaluation model is proposed to investigate the efficacy of the framework in the context of three well-known pattern catalogs and set of real design problem.

From experimental results, we concluded several important consequences. First, the finest weighting method for each unsupervised learner varies across the design pattern collections, which mean, the same weighting method cannot be recommended in the proposed method. For example, we observe the TFIDF, TFIDF, TFC as the finest weighting method for the outperform unsupervised learner Fuzzy c-means to organize the patterns of GoF, Douglass and Secuirty respectively. Consequently, we cannot recommend that TFIDF remains always better for fuzzy c-means in the case of GoF, Douglass and Security patterns organization. Second, in the case of determination of appropriate design pattern classes, fuzzy c-means and PAM-Euclidean achieves better results as compared to other learning techniques. Third, the efficacy of the framework is highly provoked by feature selection methods. Such as, there is an increase from 5.26% to 10.52% (in term of F-measure) in the efficiency of Fuzzy c-means when features are selected via Information Gain feature selection techniques. Finally, the threshold values of Θ_1 and Θ_2 parameters are determine through experiments.

In future work, we will study the semantics of two connecting words instead of a single word in the feature to increase the efficacy of the framework. Moreover, we will use the proposed framework with pattern groups published on online repositories.

REFERENCES

[1] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice. Addison-Wesley Professional. Third Edition, September 2012.

[2] W. G. Wood, A Practical Example of Applying Attribute-Driven Design (ADD), Version 2.0, Technical Report, Software Engineering Institute, 2007.

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

[4] J. Tidwell, Designing Interfaces:Pattern for effective interaction design, 2nd Edition, O'Reilly Media, pages. 578, 2010.

[5] M. O. Elish and M. A, Mohammad, Quantitative analysis of fault density in design patterns: An empirical study. Information and Software Technology, 66, p. 58-72, 2015.

[6] B. Walter and T. Alkhaeir, The relationship between design patterns and code smells: An exploratory study, Information and Software Technology, 74, p. 127-142, 2016.

[7] C. Zhang and D. Budgen, A survey of experienced user perceptions about software design patterns. Information and Software Technology, 55(5), p. 822-835, 2013.

[8] A. Torres, R. Galante, M. S, Pimentam and A. J. B. Martins, Twenty years of object-relational mapping: A survey on patterns, solutions, and their implications on application design, Information and Software Technology , 82, p. 1-18, 2017.

[9] A. Ampatzoglou, S. Charalampidou and I. Stamelos, Research state of the art on GoF design patterns: A mapping study, The Journal of Systems and Software, 86, p. 1945-1964, 2013.

[10] A. Birukou, A Survey of Existing Approaches for Pattern Search and Selection, Proceeding of PLoP, 2010.

[11] P. Velasco-Elizondo, R. Marín-Piña, S. Vazquez-Reyes, A. Mora-Soto and J. Mejia, Knowledge representation and information extraction for analyzing architectural patterns, Science of Computer Programming, 121, p. 176-189, 2016.

[12] P. Coad, D. North and M. Mayfield, Object Models: Strategies, Patterns, Applications, Yourdon Press, NJ, USA, 1995

[13] W. Pree, Design Patterns for Object-Oriented Software Development, ACM Press/Addison-Wesley, 1995

[14] C. Bouhours, H. Leblance and C. Percebois, Spoiled Patterns: How to Extend the GoF, Software Quality Journal, 23, p. 661-694, 2015.

[15] G. Booch, Handbook of Software Architecture, 2006. http://handbookofsoftwarearchitecture.com/

[16] H. Baraki et al., Interdisciplinary Design Patterns for Socially Aware Computing, Proceeding of the 37th International Conference on Software Engineering (ICSE). 2015.

[17] D. K. Kim and C.E. Khawand, An Approach to Precisely Specifying the Problem Domain of Design Patterns, Journal of Visual Languages and Computing, 18, p. 560 -591, 2007.

[18] N. L. Hsueh, J-Y. Kuo and C-C. Lin, Object-Oriented Design: A Goal-driven and Pattern-based approach. Journal for Software and Systems Modeling, 8 (1), p.1-18, 2007.

[19] D. K. Kim and W. Shen, Evaluating Pattern Conformance of UML Models: A divide and conquer approach and case studies. Software Quality Journal, 16 (3), p. 329 -359, 2008.

[20] A. Rouhi and B. Zamani, Towards a formal model of patterns and pattern languages, Information and Software Technology, 79, p. 1-16, 2016.

[21] S. Hasso and C. R. Carlson, A Theoretically-based Process for Organizing Design Patterns. Proceedings of 12th Pattern Language of Patterns, 2005.

[22] E.Blomqvist, Pattern Ranking for Semiautomatic Ontology Construction, Proceedings of SAC, 2008.

[23] P.E. Khoury,A. Mokhtari,E. Coquery and M.S. Hacid, An Ontological Interface for Software Developers to select Security Patterns. Proceedings of 19th International Conference on Database and Expert Systems Application, (DEXA'08), pp. 297-301, 2008.

[24] S. M. H Hasheminejad and S. Jalili, Design Patterns Selection: An Automatic two-phase Method, Journal of System and Software, 85, p. 408-424, 2012.

[25] S. Hussain, J. Keung and A.A. Khan, and K.E. Bennin, A Methodology to Automate the Selection of Design Patterns, Proceedings of International Conference on Computers, Software and Application, IEEE, 2016.

[26] S. Hussain, J. Keung and A. A. Khan, Software design patterns classification and selection using text categorization, Applied Soft Computing, 58, p. 225-248, 2017.

[27] I. Idris and A. Selamat, Improved Email Spam Detection Model with Negative Selection Algorithm and Particles Swarm Optimization, Applied Soft Computing, 22, p. 11-27, 2014.

[28] E. Saraç and S.A. Ozel, An Ant Colony Optimization based Feature Selection for Web Page Classification, p. 1-16, 2014.

[29] W. Medhat, A. Hassan and H. Korashy, Sentiment Analysis Algorithms and Applications: A Survey, Ain Shams Engineering Journal,5, p. 1093-1113, 2014.

[30] C. Zhang, X. Wu, Z. Niu and W. Ding, Authorship Identification from Unstructured Texts, Knowledge Based Systems, 66, p. 99-111, 2014.

[31] M. F. Delgado, E. Cernadas, S. Barro and D. Amorim, Do we need hundreds of Classifiers t solve real world classification problems?, Journal of Machine Learning Research, p. 3133-3181, 2014.

[32] A. Vattani, k-means Requires Exponentially Many Iterations Even in the Plane, Discrete & Computational Geometry, 45, p. 596-616, 2011.

[33] J. C. Bezdek, R. Ehrkich and W. Full, FCM: The Fuzzy c-Means Clustering algorithm, Journal of Computers and Geosciences vol. 10, No 2-3, p. 191-203, 1984.

[34] L. Kaufman and P. J. Rousseeuw, Clustering by means of Medoids, in Statistical Data Analysis Based on L_1-norm and Related Method, p. 405-416, 1987.

[35] B. P Douglass, Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison-Wesley/Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[36] L. Rising, The Pattern Almanac 2000. Addison-Wesley,2000.

[37] M. Schumacher, E. Fernandez, D. Hybertson and F. Buschmann, Security Patterns: Integrating Security and Systems Engineering. John Wiley and Sons. 2006.

[38] W. Zimmer, Relationships between Design Patterns. Journal of Pattern Languages of Program Design, 1, p. 345-364, 1995.

[39] G. J. Kim and J.S. Han, Clustering Algorithm of Design Pattern using Object-Oriented Relationship. Proceeding of Computational Science and Its Applications LNCS, Springer, ICCSA, p. 997-1006, 2007.

[40] W.F. Tichy, A catalogue of general-purpose software design patterns, Proceedings of Technology of Object-Oriented Languages and Systems, p. 330 -339, 1997.

[41] S. J. B. Castro, R. G. Cresp, V. H. M. Garcia, Patterns of Software Development Process, International Journal of Artificial Intelligence and Interactive Multimedia, 4(1), 2011

[42] N. Russell, A. H. M. ter Hofstede, D. Edmond, W. M. P. van der Aalst, Workflow Data Patterns, Proceedings of the 24th international conference on Conceptual Modeling, p. 353-368, 2005.

[43] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, S.T. Halkidis, Design pattern detection using similarity scoring, IEEETrans. Softw. Eng. 32(11) (2006)896-909.

[44] A. K. Uysal, An Improved Global Feature Selection Scheme for Text Classification, Expert System with Applications, 43, p. 82-92, 2016.

[45] C. Alexander, S. Ishikawa, M. Silverstein, I. Jacobson and S. Angel, A Pattern Language: Towns, Building, Constructions. Oxford University Press, New York, 1977.

[46] A. Hotho, A. Nurnberger and G. Paab, A Brief Survey of Text Mining, Journal for Computational Linguistics and Language Technology, 20, p. 19-62, 2005.

[47] M. F. Porter, An algorithm for Suffix Stripping, Journal of Program-Electronic Library and Information Systems 40, p. 211-218, 2006.

[48] P. D. Turney and P. Pantel, From Frequency to Meaning: Vector Space Models of Semantics, Journal of Artificial Intelligence Research, 37, p. 141-188, 2010.

[49] G. Forman, An Extensive Empirical Study of Feature Selection Metrics for Text Classification, Journal of Machine Learning Research, 3, p. 1289-1305, 2003.

[50] C. D. Manning, P. Raghavan and H. Schutze, Introduction to Information Retrieval, Cambridge University Press. 2008.

[51] M. Ricardo et al, PreText: A Simple Text Preprocessing Tool, http://sites.labic.icmc.usp.br/torch/msd2011/jpretext/

[52] E. Alpaydın, E. Introduction to Machine Learning, Second Edition. The MIT Press Cambridge, Massachusetts, London, England, 2010.

[53] A. Huang, Similarity Measures for Text Document Clustering, Proceedings of NZCSRSC, 2008.

[54] F. Sebastiani, Machine Learning in Automated Text Categorization, ACM Computing Surveys, 34(1), p. 1-47, 2002.

[55] N. X. Vinh, J. Epps and J. Bailey, Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance, Journal of Machine Learning Research, 11, p. 2837-2854, 2010.

[56] P. Rousseeuw, Silhouettes: A Graphical aid to the Interpretation and Validation of Cluster Analysis, Journal of Computational and Applied Mathematics, 20(1), p. 53-65, 1987.
[57] A. Silberschatz, P.B. Galvin,G. Gagne, Operating System Concepts, 6 edition, 2002.
[58] A.S Tanenbaum and H. Bos, Modern Operating Systems, Fourth edition. Pearson Education Limited, 2015.
[59] A. Shalloway and R. Trott, Design Pattern Explained: A new Perspective on Object Oriented Design, Addison Wesley, 2001.
[60] J. M. Santos and M. Embrechts, On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification, Proceeding of the 19th International Conference on Artificial Neural Netwrokds, p. 175-184, 2009.
[61] G. Saporta and G. Youness, Comparing Two Partitions: Some Proposals and Experiments, Chapter of COMPSTAT, Springer, p. 243-248, 2002.
[62] A. Shvets, Design Pattern Explained Simply, pages 117, 2010, https://sourcemaking.com/design-patterns-ebook
[63] S. Hussain et al., Implications of deep learning for the automation of design patterns organization, Journal of Parallel and Distributed Computing, 2017.

Appendix A: Cosine Similarity (CS) Measures Values for 9 Design Problems (Section 5.4)

Table 6. Cosine Similarity Results for Design Problem 1 (DP-1)

Structural Design Patterns	Cosine Similarity (CS)
Adapter	**0.52**
Bridge	0.11
Composite	0.23
Decorator	0.22
Fascade	0.36
Flyweight	0.23
Proxy	0.26

Table 7. Cosine Similarity Results for Design Problem 2 (DP-2)

Behavioral Design Patterns	Cosine Similarity (CS)
Chain of Responsibility	0.47
Command	0.53
Interpreter	0.30
Iterator	0.38
Mediator	0.24
Memento	0.32
Observer	0.37
State	**0.64**
Strategy	0.51
Template	0.12
Visitor	0.37

Table 8. Cosine Similarity Results for Design Problem 3 (DP-3)

Behavioral Design Patterns	Cosine Similarity (CS)
Chain of Responsibility	0.35
Command	0.18
Interpreter	0.23
Iterator	0.26
Mediator	0.35
Memento	0.38
Observer	0.56
State	0.34
Strategy	0.56
Template	0.42
Visitor	**0.72**

Table 9. Cosine Similarity Results for Design Problem 4 (DP-4)

Memory Design Patterns	Cosine Similarity (CS)
Fixed Size Buffer	**0.67**
Garbage Collection	0.28
Garbage Compactor	0.45
Pool Allocation	0.28
Smart Pointer	0.47
Static Allocation	0.32

Table 10. Cosine Similarity Results for Design Problem 5 (DP-5)

Resource Design Patterns	Cosine Similarity (CS)
Order locking	0.16
Priority ceiling	0.42
Priority inheritance	0.24
Simultaneous locking	**0.67**
Critical Section	0.30
Highest locker	0.23

Table 11. Cosine Similarity Results for Design Problem 6 (DP-6)

Distribution Design Patterns	Cosine Similarity (CS)
Broker	0.21
Data Bus	0.47
observer	0.26
proxy	0.32
Remote method call	0.16
Shared memory	**0.69**

Table 12. Cosine Similarity Results for Design Problem 7 (DP-7)

Access Control Models (ACM) Design Patterns	Cosine Similarity (CS)
Authorization	0.14
Multilevel Security	0.20
Reference Monitor	0.31
Role Rights Definition	0.45
Role Based Access Control	**0.61**

Table 13. Cosine Similarity Results for Design Problem 8 (DP-8)

Accounting Design Patterns	Cosine Similarity (CS)
Audit Requirements	0.18
Audit Trails and Logging Requirement	**0.69**
Intrusion Detection Requirements	0.28
Non-Repudiation Requirements	0.31
Security Accounting Requirements	0.29

Table 14. Cosine Similarity Results for Design Problem 9 (DP-9)

Enterprise Security and Risk Management (ESRM) Design Patterns	Cosine Similarity (CS)
Enterprise Partner Communication	0.36
Enterprise Security Approaches	0.39
Enterprise Security Services	0.41
Risk Determination	0.33
security needs identification for enterprise asset	0.25
Threat Assessment	**0.65**
Vulnerability Assessment	0.21

Appendix B: Unsupervised Learning Techniques used in the Proposed Study

K-means:

The K-means unsupervised learning technique is widely used for cluster analysis in certain domains such as data mining. The primary objective of k-means is to partition the N observation (i.e. $\{x_1,x_2,\ldots,x_N\}$) into the k number of clusters by minimizing the sum of the distance of each observation to the k centers.

$$\underset{S}{\arg\max}\ \sum_{i=1}^{k}\sum_{x\in S_i}\left\|x-\mu_i\right\|^2 \tag{14}$$

The association of an observation with a cluster is decided on the base of nearest mean μ_i . The first step of k-means algorithm is assign each observation to a cluster with least mean evaluated through the equation 14. In next coming steps of k-means algorithm, the new means to the observations centroid are calculated using the equation 15, and this process remains continue till no further converging is required [48].

$$m_j^{t+1} = \frac{1}{\left|S_j^t\right|}\sum_{x_i\in S_j^t} x_i \tag{15}$$

Agglomerative

The Agglomerative algorithm is commonly used type of hierarchical clustering, which normally occurs in two fashions that is top-down and bottom-up. The agglomerative algorithm follows the bottom-up fashion and take start by considering each document as a single cluster, and successfully merge the cluster's pairs in subsequent iterations until the a single cluster of sample documents. The adjustment of the number of clusters is not required as prerequisite of Agglomerative algorithm. The primary objective of Agglomerative clustering algorithm can be explained through equation 16.

$$K = \underset{K'}{\arg\min}\ \left[RSS(K') + \lambda K'\right] \tag{16}$$

In the equation 16, K, K', λ and RSS (Residual Sum of Squares) present the number of clusters (in consecutive iterations), penalty for additional cluster, and the residual sum of squares.

Fuzzy c-means

The Fuzzy c-means algorithm is used to describe the relationship of an observation (i.e. Data point) with two or more clusters. The main objective of Fuzzy c-means algorithm can be achieved using the equation 17.

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m\left\|x_i - c_j\right\|^2 \tag{17}$$

In the equation, m, u_{ij} , x_i , and c_j represents the real number (i.e. m>1), degree of membership of an observation, ^{i}th observation, center of ^{j}th cluster respectively. The membership grade for an observation indicate its association with corresponding clusters. The fuzzy c-means algorithms is similar to k-means in terms of a number of chosen clusters, calculating the coefficient for an observation in the clusters and compute centeroids [49].

Partitioning Around Medoids (PAM)

The Partitioning Around Medoids (PAM) algorithm is the realization of k-medoid which used greedy search with two main assumptions a) optimum solution can not find, and b) faster than an exhaustive search. Like k-means, *k*-medoids algorithms are also partitional and minimize the distance between labelled observations. The objective function of PAM is presented in equation 18.

$$F(x) = \min\sum_{i=1}^{n}\sum_{j=1}^{n} d(i, j)z_{ij} \tag{18}$$

Since, the matrix of dissimilarity is used as prerequisites for PAM algorithm, consequently, two well-known metrics are used. The first metric is named as Euclidean distance which is used to evaluate the root sum of squares of differences. The Euclidean distance between two observations (i.e. Data Points) such as p and q can be computed using equation 19.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p2)^2 + \ldots + (q_n - pn)^2} \tag{19}$$

The second metric is named as Manhattan distance which is used to evaluate the sum of absolute distances. The Manhattan distance between two observations (i.e. Data Points) such as p and q can be computed using equation 20 [50].

$$d(p,q) = \sum_{i=1}^{n} \left| p_i - q_i \right| \tag{20}$$

Appendix C: Example to describe the implementation of proposed framework Gang-of-Four (GoF) design pattern catalog and a related design problem.

Implementation of the proposed framework for the organization of GoF design patterns:

Step 1. The removal of the stopwords, words stemming and indexing activities are performed to create the vector space model for the N design patterns (i.e. N=23) and M unique words (i.e. M=1465). Though, we retrieved M=1465 features (Step-1), however, due to space limitation, we are describing the vector space model with M=10 as follows (Table 15).

Table 15. Vector Space Model for GoF Design Pattern Collection

N Patterns	Client	Class	Application	Extend	Hierarchy	System	Structure	Behavior	Differ	Execution
1	0	0	0	1	1	1	0	0	1	0
2	1	0	0	0	0	0	0	0	1	0

23	0	0	1	0	0	1	0	0	1	1

Step 2. We applied the weighting methods (Section 4.1). However, in the table 15, we present the VSM with the binary weighting method.

Step 3. Though we assess the performance of each unsupervised learner with all weighting methods, however in Table 16, we only describe the analysis results of outperform unsupervised learner that is fuzzy c-means. The performance of fuzzy c-means with different weighting methods is assessed using the evaluation criteria described in section 5.1. According to the evaluation criteria, the TFIDF is the finest weighting method for fuzzy c-means in the case of the organization of the GoF design patterns.

Step 4. The same procedure is followed to find the best weighting for other unsupervised learners in the context of target design pattern groups [3, 37, 39]. The Figure 4-(a-c) depict the performance of all unsupervised learner under study with each weighting method for the GoF [3], Douglass [37], and Secuirty [39] design pattern collection respectively. In case of the GoF design patterns, from the Figure 4-a, we can observe the best weighting methods for the unsupervised learners. Such as TFIDF for Fuzzy c-means, PAM-Euclidean, and PAM-Manhattan, Entropy for Agglomerative, and Binary for K-means.

Step 5. In case of the organization of the GoF design patterns, the performance of unsupervised learners with their finest weighting method are assessed using the evaluation criteria (Section 5.1). Though we evaluated the effectiveness of the proposed framework with different number of clusters (Figure 5-a), however, with 3 clusters (i.e. k=3, according to expert's opinion), in terms of ARI, the performance of Fuzzy c-means (ARI=0.93) remain better than other unsupervised learners such as K-means (ARI=0.87), Agglomerative (0.83), PAM-Euclidean(ARI=0.76), and PAM-Manhattan (ARI=0.73).

Table 16. Performance assessment of outperform Fuzzy c-means with weighting methods

GoF Design Patterns			Fuzzy c-means with Weighting Methods				
Pattern	Category (By Expert)	Cluster Number	Binary	Entropy	LTC	TFC	TFIDF
Abstract Factory	Creational	1	1	2	1	1	2
Adapter	Structural	2	2	2	3	2	2
Bridge	Structural	2	2	2	3	2	2
Builder	Creational	1	1	1	1	3	1
Chain of Responsibility	Behavioral	3	3	1	2	3	1
Command	Behavioral	3	3	3	2	3	1
Composit	Structural	2	2	1	1	3	1
Decorator	Structural	2	2	2	2	3	1
Factor Methods	Creational	1	1	1	1	2	2
Fascade	Structural	2	2	1	1	2	2
Flywieght	Structural	2	2	2	2	3	1
Interpreter	Behavioral	3	3	3	1	2	2
Iterator	Behavioral	3	3	2	2	3	1
Mediator	Behavioral	3	3	3	2	3	1
Memento	Behavioral	3	3	3	3	3	1
Observer	Behavioral	3	3	1	1	3	1
Prototype	Creational	1	1	1	2	3	2
Proxy	Structural	2	2	1	2	3	1
Singleton	Creational	1	1	1	1	1	2
State	Behavioral	3	3	1	3	3	1
Strategy	Behavioral	3	3	3	1	2	3
Template	Behavioral	3	3	1	1	2	2
Visitor	Behavioral	3	3	3	3	2	3

Implementation of the proposed framework for determination of a pattern class for the Design Problem (DP) in the context of GoF design patterns:

The description of a Design Problem (DP) [8] described in the context of GoF pattern collection is as follows. The keywords which are used in the table 15 are highlighted in the description of DP.

Design Problem: "Design a system enabling to display on a screen some empty windows (no button, no menu...). A window can have several **different** styles depending on the platform used. We consider two platforms, XWindow and Presentation Manager. The **client** code must be written independently and without knowledge of the future **execution** platform. It is probable that the system evolves in order to display specialized windows by '**application** windows' (able to manage applications) and 'iconised windows' (with an icon)".
Though the structural pattern class has been mapped for the DP [8], however, in order to determine the appropriate pattern class for DP using the proposed framework, the following steps are used.
Step 1. The removal of the stopwords, words stemming and indexing activities are performed to create the vector space model for the N design patterns (i.e. N=23), one design problem DP, and M unique words (i.e.

M=1471). Though, we retrieved M=1465 features (Step-1), however, due to space limitation, we are describing the vector space model with M=10 and binary weighting method, shown in Table 17.

Table 17. Vector Space Model for GoF Design Patterns and Design Problem (DP)

N+1 Patterns + Problem	Client	Class	Application	Extend	Hierarchy	System	Structure	Behavior	Differ	Execution
1	0	0	0	1	1	1	0	0	1	0
2	1	0	0	0	0	0	0	0	1	0

23	0	0	1	0	0	1	0	0	1	1
24	1	0	1	0	0	1	0	0	1	1

Table 18. Performance assessment of unsupervised learners with their best weighting methods

GoF Design Patterns		Unsupervised Learners with Weighting Methods					
Pattern	Category (By Expert)	Cluster Number	PAM-Euclidean + TFIDF	PAM-Manhattan + TFIDF	Agglomerative + Entropy	k-means + Binary	c-means + TFIDF
Abstract Factory	Creational	1	1	3	1	1	1
Adapter	Structural	2	1	2	3	2	2
Bridge	Structural	2	2	2	2	2	2
Builder	Creational	1	1	1	1	3	1
Chain of Responsibility	Behavioral	3	2	1	2	3	1
Command	Behavioral	3	3	3	3	3	1
Composit	Structural	2	2	2	1	2	2
Decorator	Structural	2	2	2	2	3	2
Factor Methods	Creational	1	1	2	1	3	1
Fascade	Structural	2	2	1	3	2	2
Flywieght	Structural	2	2	2	2	2	2
Interpreter	Behavioral	3	3	3	1	2	3
Iterator	Behavioral	3	3	2	1	3	1
Mediator	Behavioral	3	3	3	3	3	3
Memento	Behavioral	3	3	3	3	3	1
Observer	Behavioral	3	3	1	1	3	1
Prototype	Creational	1	1	1	2	3	3
Proxy	Structural	2	2	2	2	2	2
Singleton	Creational	1	1	1	1	1	3
State	Behavioral	3	3	2	3	3	1
Strategy	Behavioral	3	3	3	2	3	3
Template	Behavioral	3	1	1	1	1	1
Visitor	Behavioral	3	3	3	3	2	3
Design Problem-1	Structural	2	2	2	1	2	2

Step 2. We applied the weighting methods (Section 4.1). However, in the table 17, we present the VSM with the binary weighting method.

Step 3. The unsupervised learners (under study) employed in the proposed framework are assessed with their best weighting methods (e.g. TFIDF for Fuzzy c-means, PAM-Euclidean, and PAM-Manhattan) to determine the appropriate pattern class for the DP. The results are shown in Table 18. The results of Table 18 depict that unsupervised learners Fuzzy c-means, k-means, PAM-Euclidean, and PAM-Manhattan have the capacity to determine the right pattern class (i.e. Structure according to expert opinion [5]), which present the effectiveness of the proposed framework. The list of patterns (Adapter, Bridge, Composite, Decorator, Fascade, Flyweight, and Proxy) belongs to the pattern class which is determined via outperform fuzzy c-means for the design problem (DP).

Step 4. The Step 1 to Step 3 can be repeated for the rest of the problems [55, 56, 57, 59] related to target design pattern collections [3, 37, 39].

Implementation of proposed framework for the selection of right design pattern(s):

The right pattern class and list of correlated design patterns for Design Problem (DP) is determined through the outperform unsupervised learner (Fuzzy c-means) of the proposed framework. The main steps to select the right design pattern(s) for the Design Problem (DP) as follows.

Step 1. The cosine similarity between the feature vectors of DP and patterns of the corresponding pattern class (determine through proposed framework) is computed using the equation 3.

Step 2. We applied the equation 5-7 to suggest the right design patter(s) for the DP. The thresholds Θ_1 and Θ_2 are suggested on the base of cosine values retrieved via the equation 3.

a) The first similarity option (i.e Equation 5) aid to recommend the right design pattern (i.e. The Adapter pattern with highest cosine value 0.52) for the DP.

b) In case of the second similarity option, the suggested threshold value Θ_1 is 0.36 to 0.51 (Table 15) to recommend the zero, one or more than one design pattern(s).

c) In case of the second similarity option, the suggested threshold value Θ_2 is 0.0 to 0.16 (Table 15) to recommend design patterns contiguous to the best design pattern (with CS_{max})

Step 3. The same procedure is repeated for the rest of design problems.

Step 4. Finally, the equation 13 is used to select the best similarity option (Section 4.4) by considering the ratio of the number of right suggested and total suggested design patterns. In the case of Design Problem (DP), the first similarity option is suggested to select the right design pattern.