Retrieving Model for Design Patterns

Sarun Intakosum and **Weenawadee Muangon**, Non-members

ABSTRACT

The purpose of this research is to develop a retrieving model for design patterns, based on problem domain context. The research aims to provide a convenient way for developers to access to the right design patterns that can solve their design problems. The proposed model is composed of two major parts, the analysis of design pattern documents to create search indexes, and the calculation of index weight. Vector space model is used to calculate similarity between queries and documents. The result of this research shows that precision of the proposed model, in retrieving correct design patterns, is about 70 percents in average.

Keywords: Keywords: Design patterns, Information retrieval systems, Retrieving design patterns.

1. INTRODUCTION

Design is one of important steps in software development. Software design goals are correctness, robustness, flexibility, reusability, and efficiency [1]. One approach to reach these design goals is to use design patterns. Experienced software developers and researchers develop and record solutions to design problems, which occurred again and again in many contexts, as reusable common design solutions and call them design patterns. There are many catalogues of design patterns available today in the forms of books [2- 4] or software tools such as MVCASE tool [5]. These catalogues, however, are always searched by a pattern name. Searching for patterns using their name is useful for only experienced developers who already know what patterns they need. Inexperienced developers, on the other hand, may have to spend time to read throughout each pattern description before they can find the right patterns.

One solution to this problem is to develop a model that can retrieve design patterns based on problem domain that developers are currently working. Information retrieval techniques are proven to be a useful technique to retrieve information based on users need. This research, therefore, proposes a model to retrieve design pattern using information retrieval techniques, and develops a prototype system based on the model. Developers can then search for the appropriate design patterns by entering keywords from a problem domain context, the system responses with the patterns that are related to the keywords, ranking by their similarities to the keywords.

The remainder of this paper is organized as follows. The fundamental knowledge of design patterns is described in section 2. Section 3 describes the basic concept of information retrieval systems. Section 4 describes the proposed retrieving model along with the information retrieval techniques applied in this research. Section 5 is about the experiment and its results. Finally, section 6 gives conclusion as well as an undergoing research.

2. DESIGN PATTERNS

In software engineering terminology, design patterns are common solutions to design problems. Generally in software design, some software developers found many problems that can be solved using similar solutions. The software developers then record those solutions in the term of reusable artefacts that is called patterns. As a result, all developers can gain the proper software design by not having to spend much time to find the solutions to the problems that are already solved.

There are many researchers and developers who developed and recorded design patterns that can be used to solve the problems in some specific problem domains, for examples, [2, 4]. However, design patterns that are considered fundamentals and can be applied to various problem domains are those described in [3], which are known as GOF design patterns. This research focuses on GOF design patterns because of their generality. There are 23 design patterns described in GOF; these patterns are divided into three categories, creational, structural, and behavioral. Creational design patterns aims to solve the problem of crating objects. Structural design patterns are for creating collection of related objects. Behavioral patterns are used to capture behavior among related objects.

3. INFORMATION RETRIEVAL SYSTEMS

Information retrieval system is a technique that allows users to search for information based on their needs. Before correct documents can be retrieved the indexing and term weighting must be prepared first. Indexing is the process to find the terms that can

Manuscript received on December 15, 2006; revised on March 15, 2007.

The authors are with the Software Systems Engineering Laboratory, Department of Mathematics and Computer Science,Faculty of Science, King Mongkuts Institute of Technology Ladkrabang, Bangkok 10520, Thailand; E-mail: kisurun@kmitl.ac.th, weenawadee@hotmail.com

be used to retrieve documents. Term weighting is the process to calculate weighted-value to each index term. This value shows how important of index term in the documents. Indexing and term weighting are very important for the retrieval performance. The index and its weight will be used in the retrieval process along with the user query to retrieve the required documents.

Many information retrieval models are available today [6]. However, the model that is considered classic, simple, and provides an acceptable retrieval performance is vector space model. Vector space model is a mathematical model that uses vector concept to represent documents and user queries. Documents and queries are represented as n-dimensional vector spaces. A document will be retrieve based on its similarity to the query. One way to determine the similarity between document and query is to compute the cosine value of the angle between document and query vectors [6]. Index weight is an important factor for calculating the similarity. The inverse document frequency (idf) is one of useful tools to specify the weight of an index. The idf shows distribution of an index in documents. If an index appears in many documents its quality to retrieve the right documents may be low.

There are many techniques to evaluate the retrieval performance such as those stated in [6,7]. Among them, two most accepted measures are precision and recall. Precision is the ratio of the retrieved documents that are relevant. Recall is the ratio of the relevant documents that are retrieved. The high precision ratio at the low recall level indicates that the require documents have been ranking at the high level.

4. RETRIEVING MODELS FOR DESIGN PATTERN

This research proposes the models for indexing and term weighting. Both models are achieved by analyzing and restructuring GOF design patterns documents. The details can be described as follows:

4.1 Model For Indexing

After carefully analyzing the structure of GOF design patterns document, we decide to categorize the indexes into three groups. The first group is the indexes that can refer to a design pattern by its name and alias name (Also known as section in GOF design patterns catalogue). The second group is the indexes that can refer to a design pattern category. The indexes in this group are category name and the terms that can refer to the category functions. The last group is the indexes that refer to the functions and/or solutions provided by a design pattern. The intent section of GOF design pattern catalog contains this information. Almost all indexes in this group, therefore, come from this section. In addition, we also divide the intent section into three parts, entity [8], keyword, and function, which will be described in more detail later. Based on the above information, we can restructure a GOF design pattern document as is shown in figure 1.

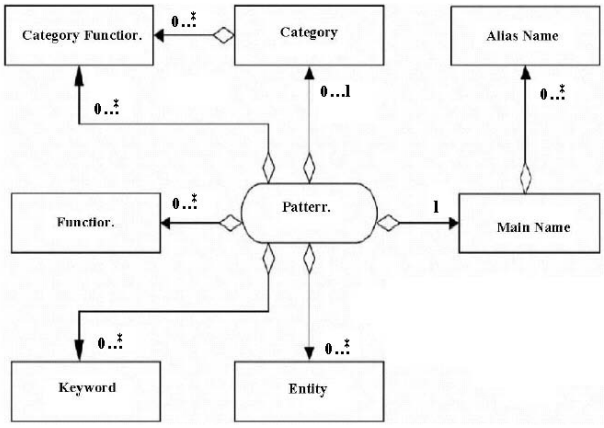


Fig.1: *Structure of design patterns description*

Each component can be explained as follows:

1.)Main name contains an index that represents a design pattern name.

2.)Alias name contains indexes that refer to other names for the particular design pattern.

3.)Category contains an index that represents a category of a design pattern.

4.)Category function contains indexes that can refer to major function of a design patterns category.

5.)Function contains indexes that refer to major functions or actions stated in the intent section of a design pattern description.

6.)**Entity** contains indexes to objects, classes, and interfaces those are described in the intent section of a design pattern description. In this research we apply the method described in [8] to specify the indexes in this part.

7.)**Keyword** contains indexes that show specific features of a design pattern.

The multiplicity in the diagram specifies how many indexes that each component can have for one design pattern. This information will be used as one criterion to specify priority level of indexes. The priority level shows quality of indexes to identify proper design patterns. It is used to calculate index weight, discussed later. The multiplicity for each component can be described as follows:

1.Main name, a design patterns can have only one main name.

2.Alias name, in addition to main name, a design pattern may be known by one or more alias names, for example the Abstract Factory pattern is also known as Kit [3].

3.Category, each design pattern can fall in only one category.

4.Category function, Function, Entity and Keyword, for these components more than one term may identify the same component, for example, the Function component of observer design pattern can be identified by one of the following terms "Update" and "Notify".

The search indexes in our research are created based on the above structure. We divide the indexes into two categories, single-word, and phrase indexes.

4.1.1 Single-word Index

A single-word index, as its name, contains only one word. Words that will not be considered as a single-word index are that refer to normal object oriented terminologies such as "Object", "Instance", "Class", and "Subclass". The major reason is that they are always appeared in almost all design patterns descriptions. However, the composition of these words to other singleword indexes may result in a better search index; therefore, they can be used as a part of a phrase index, discussed next.

4.1.2 Phrase Index

A phrase index is a composition of two or more single-word indexes, and/or object-oriented terms, "create family object" is an example of this type of index.

4.2 Model For Term Weighting

The model to calculate the index weight is divided into two parts. The first part is to rank the indication ability of indexes from the GOF design patterns document structures described in previous section. The second part is to calculate index weight based on the number of keywords found from the query statements.

In the first part, the indication ability is based on how well the indexes can narrow the search scope, and the number of indexes that can refer to the same component. For example, there is only one index that points to each Category, while two or more indexes can refer to a Category Function, therefore, Category has higher priority than Category Function. In summary, the rank of the indication ability of the indexes based on our GOF design patterns document structures can be shown in figure 2.

The priority value, which we call priority constant, can be assigned to each level as is shown in table 1. These numbers come from our experiment, which will be discussed in the later sections.

4.3 Term Weighting For Single-Word Index

Weight calculation for single-word index is done by adding inverse document frequency (idf) to priority constant. There are four cases, based on the type of an index, to consider as follows:

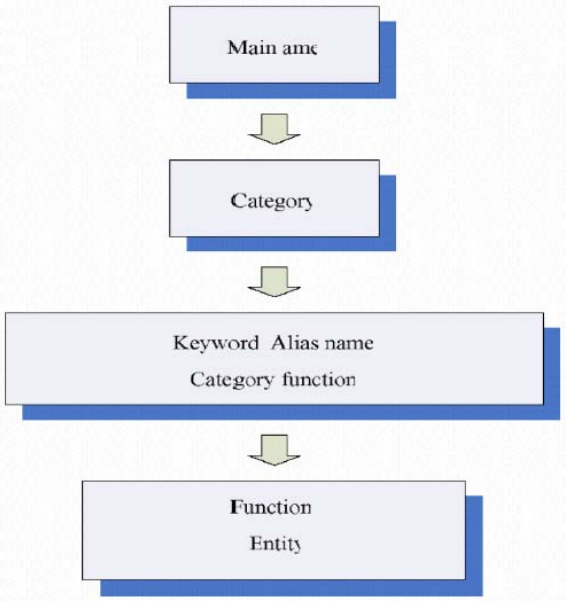


Fig.2: *Rank of indication ability of the indexes based on document structures*

Table 1: *priority constant of design patterns components*

Design pattern component	Priority constant
Main name	0.5
Category	0.3
Alias name, Keyword and Category function	0.2
Function and Entity	0.1

Case 1: an index is in Main name.

$$weight(t_i, d_k) = idf(t_i) + PC\_Name \quad (1)$$

Case 2: an index is in Category.

$$weight(t_i, d_k) = idf(t_i) + PC\_Category \quad (2)$$

Case 3: an index is in Alias name, Keyword, or Category Function.

$$weight(t_i, d_k) = idf(t_i) + PC\_Keyword \quad (3)$$

Case 4: an index is in Function and Entity.

$$weight(t_i, d_k) = idf(t_i) + PC\_Function \quad (4)$$

Where ; $t_i$is index i such that i $\in \{1, 2, 3, ..., M\}$
where M is number of indexes
d_k is document k such that$k \in \{1, 2, 3, ..., N\}$
where N is number of documents
idf(t_i)$i$is inverse document frequency of index i
weight(t_i, d_k)is weighted value of index i in document k

Pc_Name is priority constant of name Pc_Category is priority constant of category Pc_Keyword is priority constant of alias name, keyword and category

function Pc_Function PC is priority constant of function and entity

4.4 Term Weighting For Phrase Index

Calculation of weight for phrase index can be done by adding together weight of each single-word index in a phrase index. In this research, however, a phrase index can have the maximum length of only three words. The reason is based on our experiment in that using longer phrase index does not help to improve the retrieval performance. The weighting formula for a phrase index that is the combination of single-word indexes only can be formalized as show in equation 5.

$$\text{phrase\_ weight } (t_i + t_k + [t_l], d_k) = \text{weight}(t_i, d_k) + \text{weight}(t_j, d_k) + [\text{weight}(t_l, d_k)] \quad (5)$$

where $i \neq j \neq l$
$t_i, t_j, t_l$is index$i, j$and$l$respectively, such that
$i, j, l \in \{1, 2, 3, ..., M\}$where M is number of indexes
$d_k$is document k such that$k \in \{1, 2, 3, ..., N\}$ where
N is number of documents
weight$(t_i, d_k)$is weighted value of single-word index i of document k
weight$(t_j, d_k)$is weighted value of single-word index j of document k
weight$(t_l, d_k)$is weighted value of single-word index l of document k
phrase_weight$(t_i + t_k + [t_l], d_k)$is weighted value of phrase index that compose of single-word i,j and/or l in document k

As stated previously that a phrase index may contain an object-oriented terminology, in this case the formula to calculate the weight of the phrase index is as shown in equation 6.

$$\text{phrase\_weight } (t_i + [t_j] + [t\_Object], d_k) = \text{weight}(t_i + d_k) + [\text{weight}(t_j + d_k)] + PC\_Object \quad (6)$$

where i$\neq j$
t_object is an object-oriented terminology used as part of a phrase index.It can be at any location s in the phrase index.
PC_Object is the weighted value for object-oriented terminology.
The remaining terms are the same as the previous equation.

The next important point is how to define the value of PC_Object. The problem is that the object-oriented terminology tends to be appeared in almost all design patterns documents. Therefore, using the normal idf technique may result in too small value, which may not have a major impact as an indicator to the specific design patterns. On the other hands, if the value is too large it may dominate the search result. Based on our experiment, the best value for the PC_Object can be calculated using equation 7.

$$PC\_Object = [\frac{MAXidf}{2}] \quad (7)$$

where idf Max is maximum value of idf. The Max idf is calculated by assuming that the object-oriented term occurs in only one document. The major idea behind this decision is to find the value that lies in between the normal idf and the Max idf.

5. EXPERIMENT AND RESULTS

The experiment begins by creating document indexes from GOF design patterns descriptions. In this research, we create 280 single-word indexes, 40 twoword phrase indexes, and 30 three-word phrase indexes. The weight of each index is, then, calculated. The indexes and their weights are stored in our program database. We test our system based on 105 queries that cover all GOF design patterns. We also test our system using single-word indexes only, single-word indexes with two-word phrase indexes, and combination of single-word, two-word, and three-word phrase indexes, to compare their retrieval performances. The result of the experiment can be shown in the form of graph of precision/recall at eleven standard recall levels in figure3.

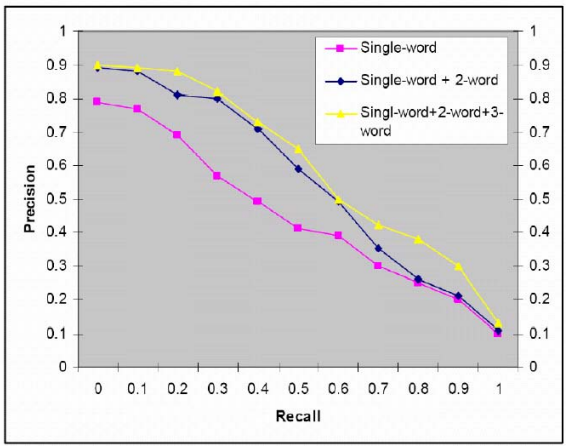


Fig.3: *Precision/Recall results for the proposed model*

The average precision is summarized in table 2.

We can conclude from the table that our system precision to retrieve the correct design patterns, when applying with the combination of single-word, two-word, and three-word indexes, is about seventy percent in average.

Table 2: *Average precision for the proposed model.*

Type of index	Average Precision	percentage
Single	0.548182	54.8
Single + 2 words of phrase	0.671429	67.1
Single + 2 + 3 words of phrase	0.705120	70

6. CONCLUSION AND FUTURE RESEARCHES

Design patterns are common solutions to design problems. Applying design patterns can help software developers to develop applications in flexible, manageable, and reusable manners. There are many design patterns catalogues available today. However, almost all of those catalogues are searched by only design patterns names. This means that developers must already know that what design patterns could be applied for their projects, and search the catalogues for just only the detail information of the patterns. This approach may not be suitable for developers who do not have much knowledge about design patterns.

This research aims to solve this problem by proposing the information retrieval model to allow developers to enter keywords to search for the proper design patterns for the design problems that they are going to solve. The proposed model is based on vector space model. In order to apply this model, the document indexes must be provided. These indexes are developed from GOF design patterns catalogue [6]. The index weight is calculated by analyzing and restructuring design patterns description, and gives the priority to each structure component based on its ability to indicate to the specific patterns. 105 queries that cover all of GOF design patterns are used to test the proposed model. The result shows that the retrieval precision is about seventy percents in average.

In this introductory research, the result is limited to the matching between document and query indexes, and index weight. We plan to improve our model by performing query analysis using the techniques such as syntactic, semantic analysis, and case-based reasoning.

References

[1] E. Braude, Software Design: From Programming to Architecture, John Wiley & Son, Inc., 2004.

[2] J. O. Coplien and D. C. Schmidt, Pattern Languages of Program Design, Addison-Wesley,1995.

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

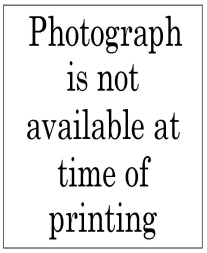
[4] J. Vlissides, J. O. Coplien, and N.L. Kerth, Pattern Languages of Program Design. Vol. 2, Addison Wesley. 1996

[5] L. Daniel, A. Alexandre, S. Eduardo, and F. Antonio, "MVCASE Tool-Working with Design Patterns." The Third Latin American Conference On Pattern Languages Of Programming. Porto de Galinhas PE. 2003

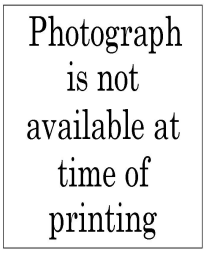
[6] R. Baeza-Yates, and B. Ribiero-Neton, Modern Information Retrieval, Addison Wesley, 1999.

[7] G. Salton, and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

[8] G. Andreas, and E. Mattias,,"Formalizing the Intent of Design Patterns." An Approach Towards a Solution to the Indexing Problem Technical report 1999-006, Uppsala University. 1999



Sarun Intakosum



Weenawadee Muangon