# Analyzing Relations among Software Patterns based on Document Similarity

Atsuto Kubo[1], Hironori Washizaki[2], Atsuhiro Takasu[2] and Yoshiaki Fukazawa[1]
'Department of Computer Science, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
a.kubo@fuka.info.waseda.ac.jp, fukazawa@waseda.jp
[2]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{washizaki, takasu}@nii.ac.jp

## Abstract

*In software development, many kinds of knowledge are shared and reused as software patterns. Howevel; the relation analysis among software patterns by hand is difficult on the large scale. In this paper, we propose a techniquefor the automatic relation analysis among the patterns. Our technique is based on a new pattern model to treat various patterns, and utilizes exiting text processing techniques to extract patterns from documents and to calculate the strength of pattern relations.* **As** *a result of experiments, the system that implements our technique has extracted appropriate relations among patterns without information on relations described in original pattern documents. Moreover, our* sys*tem has the ability to suggest relations among patterns that the author has not noticed.*

## 1 Introduction

In software development, we use various types of knowledge such as algorithms, data structures, techniques of analysis and design, project management and so on. If we can share and reuse such knowledge widely, we will be able to achieve more efficient software development. One of sharing and reusing methods to express knowledge is to describe the knowledge used in software development as a pattern. Many software patterns are available in public on the World Wide Web (WWW) and in other resources.

The software pattern usually includes the context, forces, and solution [3]. Context refers to the situation and problem that appear repeatedly in each phase of the software development. Forces describe constraints that should be considered when the pattern is applied. Solution refers to a concrete procedure to solve the corresponding problem.

The relation analysis among patterns involves determining the relation among two or more patterns, and obtaining the set of patterns which relates mutually. The purpose of the analysis is to obtain a combination of patterns. A combination of patterns can deal with larger problems than individual patterns can.

In software development, two or more solutions with different constraints might exist for the same problem. Therefore, a variety of patterns giving a solution to a certain problem can exist. In this case, the user should select the best pattern according to the constraints. At this time, the user wants to obtain many patterns that can be applied to the target problem. Therefore, it is important to analyze the relations among patterns.

There are several approaches for analyzing relations among patterns, such as [5] [7] [1 13. However, these approaches have only used the small number of patterns. In general, there are the following problems associated with the manual analysis.

- Analyzing the relations among many patterns.
- Directly comparing patterns in different pattern forms with each other.
- Directly comparing patterns published by different authors with each other.

Thus, as the number of patterns increases, difficulty to analyze the relations among patterns increases.

Therefore, the relation analysis among a large number of patterns by hand is not realistic. An automatic approach that can be applied to a large number of patterns is required; however, to the best of our knowledge, there is no approach for automatic relation analysis.

In this paper, we propose a technique for automatic relation analysis that can be applied to a large number of patterns. There are various problems associated with the automatic analysis technique.

1. How to identify a pattern document from a collection of documents.

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

2. What kind of models are appropriate for treating pattern by computer.
3. How to analyze the relations among patterns.

First, in this paper, we do not propose any solution to problem 1. The pattern documents are collected manually. Second, we solve problem **2** by proposing a common pattern model which can treat most patterns. We design the pattern model as a context transition, which is defined in the following section. Third, we solve problem **3** using the pattern model and several text processing techniques. We analyze the relations among patterns by considering the flow of pattern application.

Our technique utilizes several text processing techniques, such as stop-word removal[9], stemming[8], the TF-IDF term weighting method[10], and vector space model[9].

## 2 Preparation

In the following, we define the terms and the concepts used in this paper.

The ***pattern document*** is the document in which a pattern is described. The ***pattern form*** specifies the kind of information described in the pattern document and the structure of the pattern document. The ***pattern catalog*** is a set of pattern documents which concerns the same area and written in a same format. The ***pattern application*** is solving the problem in according to the solution.

In the pattern document, according to the examples of a number of well known pattern forms, the headings and bodies appear alternately. **A** section is a combination of a heading and a body that appears next to the heading. We denote $H$ as a set of headings, and $B$ as a set of bodies. Then, Section s as defined as follows:

$$s = (h, b), h \in H, b \in B.$$

For a set S of sections, a pattern document d is defined as follows:

$$d = \{s_1, s_2, \ldots, s_n\}, s_1, s_2, \ldots, s_n \in S.$$

For example, Figure 1 is a pattern document which describes the ***Command*** pattern of the Gang of Four (GoF) 's design patterns[5] using HTML. This pattern document contains six sections, such as ***Intent, Motivation, Applicability,*** and so on.

Table 1 shows two different pattern forms. This pattern document follows the GoF form. In the pattern document **of** Figure 1, the context is described in the section ***"Applicability",*** the force is described in the section ***"Motivation",*** and the solution is described in each of the sections of ***"Structure", "Participants",*** and ***"Collaborations".***

```
 1: <html><head>
 2: <title>Command Pattern [GoF]</title>
 3: </head><body>
 4:  <h1>Command Pattern [GoF]</h1>
 5:  <h2>Name</h2>
 6:  <p>Command</p>
 5:  <h2>Classification</h2>
 6:  <p>Behavior, Object</p>
 7:  <h2>Motivation</h2>
 8:  <p>Encapsulate a request as a parameterized …</p>
 9:  <h2>Applicability</h2>
10:  <p>More flexibility on managing an execution…</p>
11:  <h2>Collaborations</h2>
12:  <ol>
13:     <li>Client creates commands as needed, …</li>
14:  </ol>
15:  <h2>Consequences</h2>
16:  <p>Decouples an object from the …</p>
17:  <h2>Related Patterns</h2>
18:  <p>Commands may be assembled using …</p>
19:</body></html>
```

**Figure 1. Example of a pattern document**

**Table 1. Well known pattern forms**

| Pattern form | Set of headings |
|---|---|
| GoF Form [5] | $f_{GoF}$ = { Pattern Name, Classification, Also Known As, 'Motivation, Applicability, Structure, Particiants, Collaborations, Consequences, Implementation, Sample Code, Known Uses, Related Pattern } |
| PoSA Form [3] | $f_{PoSA}$ = { Name, *Also* Known As, Examples, Context, Problem, Solution, Structure, Dynamics, Consequences, Implementation, Sample Code, Known Uses, Related Pattern } |

Thus, the kind of information described in a given pattern form is determined by the particular set of headings of that pattern form. Therefore, a pattern form $f$ containing $m$ headings is defined as follows:

$$f = \{h_1, h_2, \ldots, h_m\}, h_1, h_2, \ldots, h_m \in H$$

## 3 Proposed technique

### 3.1 Pattern model

We discuss the pattern model from the viewpoint of treating the pattern automatically.

The context obtained because of applying a pattern sometimes includes a problem supported by another pattern[3]. Here, we call the context before the pattern application "starting context". Similarly, we also call the context after the pattern application "resulting context". Therefore, we assume that a pattern application is a context transition from a starting context to a resulting context.

In addition, we include a force in the model. because two patterns which differ only in terms of forces are considered

different patterns. For example, the ***Adapter Class*** pattern and the ***Adapter Object*** pattern[5] are similar in terms of starting context and resulting context. The starting contexts are similar in terms of mismatched interfaces, and the resulting contexts are similar in terms of adaptation of the interface by interface conversion. However, there are the following differences in terms of force.

- In the ***Adapter Class*** pattern, combination among classes and reusability of the code increases.
- In the ***Adapter Object*** pattern, combination among classes and reusability of the code decreases.

We model the pattern with a labeled directed graph that illustrates a flow of the pattern application (Figure 2). The starting node is the starting context, the terminal node is the resulting context, and the label at the edge indicates force.

Formally, for a context set $C = \{c_1, c_2, \ldots, c_n\}$, pattern $p$ is defined as follows:

$$p = (c_i, c_j, \lambda_k), c_i, c_j \in C, i \neq j, \lambda_k \in \Lambda.$$

The graph shown in Figure **3** illustrates a context transition system. We call this graph "Pattern Relation Graph" (PRG). **A** PRG visualizes the related pattern set. For a context set $C$, a pattern set $P$, and a force set **A,** PRG is defined as follows:

$$PRG = (CP, \mathrm{A}).$$

If a node in a PRG is shared in multiple edges, it means that multiple patterns are sharing the same context. We consider three types of relations between two patterns as follows:

**Starting contexts:** When the starting contexts of two patterns are similar, the two patterns share the same node as a starting context in the PRG. Thus, they provide different solutions to the same problem.

**Resulting contexts:** When the resulting contexts of two patterns are similar, the two patterns share the same node as a resulting context in the PRG. Thus, they provide similar results for pattern application.

**Resulting context and starting context:** When the resulting context of a certain pattern $p_1$ and the starting context of another pattern $p_2$ are similar, we can apply $p_2$ after $p_1$. The node that is the resulting context of $p_1$ and the node that is the starting context of $p_2$ are mapped to the same node in the PRG.

## 3.2 Analysis procedure

Figure **4** shows an overview of the analysis procedure in our technique. Many pattern documents that exist on the World Wide Web (WWW) are described using HTML. Therefore, our technique targets pattern documents described with HTML.

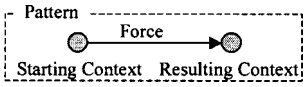


**Figure 2. Our pattern model**

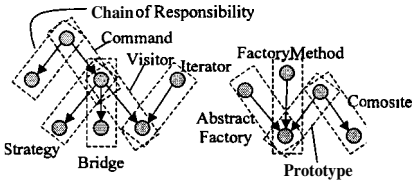


**Figure 3. Example of Pattern Relation Graph (Result of Experiment 1)**

The outline of the proposed analysis procedure is as follows. First, the input pattern document is analyzed, and sections are extracted from the document in the ***HTML Analysis*** block. Next, the form of the input pattern document is judged in the ***Pattern Form Judgment*** block. Third, the pattern is obtained from the sections according to the judged pattern form in the ***Pattern Extraction*** block. Finally, the relations between patterns are analyzed in the ***Relation Analysis*** block.

### 3.2.1 HTML document analysis

In the HTML analysis, an HTML analyzer is used to analyze the structure of the pattern document, and it obtains sections. In the pattern document described with HTML, headings and bodies are often clearly marked up. Then, the document is automatically analyzed by defining the tag set corresponding to the markup policy for the document.

- Tags that mark headings up (for example: <font size="+1">,<h2>)
- Tags that mark bodies up (for example:<p>, <li>, <dt>, and <dd>, etc.)

The HTML analyzer is a finite state machine that has three states: **empty, heading,** and **body.** Initial state of the analyzer is ***empty*** state. The appearance of a specified HTML tag changes the state of the analyzer. In the ***heading*** state, the analyzer treats an appeared partial document as a heading. Similarly, in the ***body*** state, the analyzer treats an appeared partial document as a body. In the ***empty*** state, the analyzer discards an appeared partial document. By the above-mentioned procedures, sections (pairs of a heading and a body) can be extracted from the HTML document.

For example, we specify the <h2> tag for headings and the <li> and <p> tag for bodies, and input the pattern document of Figure 1. First, the analyzer discards from the first

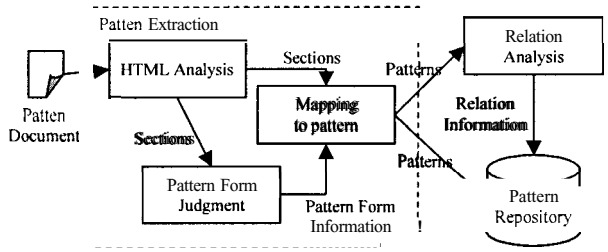


**Figure 4. Procedure overview**

to the third line. Next, the analyzer goes into the heading state, in the <h2> tag of the fourth line. When the partial document *"Name"* appears, the analyzer treats that as a heading. Then, because the <p> tag appears in the fifth line, the analyzer goes into the *body* state. The analyzer treats the partial document *"Command"* as a body, at the <h2> tag in the sixth line. The analyzer again goes into the *heading* state, and treats the appeared partial document *"Motivation"* as a heading. The analyzer goes into the body state at the <p> tag in the seventh line. The analyzer treats the partial document of *"Encapsulate a ..."* as a body. Finally, the analyzer obtains six sections.

### 3.2.2 Pattern form judgment

Because the extracted sections follow an uncertain pattern form, it is necessary to judge the pattern form. In our technique, we design the measurement that expresses the adaptability between the pattern document and pattern forms.

The pattern form is defined as a set of headings. We define $h(d)$ as a set of headings of the pattern document $d$. We define $ad(d, \mathrm{f})$ as the form adaptability of d for the pattern form f.

$$ad(f,d) = \frac{|f \cap h(d)|}{|f \cup h(d)|}.$$

To equate the inflected words such as *"Zntent"* and *"Intention"*, we perform stemming previously, using the algorithm proposed by Paice [8].

For example, we calculated form adaptability between the Command Pattern document (Figure 1) and each pattern form of Table 1. We obtained the following results: $ad(d_c, f_{PoSA}) = 0.105$ and $ad(d_c, f_{GoF}) = \mathbf{0.538.}$ Therefore, this pattern document is most likely to be GoF Form.

### 3.2.3 Pattern extraction

The extracted sections are converted into a pattern.

The pattern form specifies the kind of information described in the pattern document. Therefore, each section is mapped to the element of pattern, or discarded. If the correspondences are defined in each pattern form, the pattern can be obtained from the sections.

For example, the section *"Consequences"* in the GoF form corresponds to the resulting context, and the section *"Motivation"* corresponds to the starting context. Correspondence is defined in Table **2.**

### 3.2.4 Analysis of the relation between patterns

To analyze the relations among patterns, we obtain the relations among the partial documents in the pattern.

The technique for document similarity analysis is matured. Therefore, we obtain the relation among patterns using the similarity among the partial documents of the patterns.

We proposed three kinds of relation in section 3.1. First, partial documents are taken out of two patterns being analyzed according to the kind of relation. Next, the similarity between two partial documents is calculated. We consider that similarity indicates the strength of the relation among two the patterns. All of those relations are sorted by the strength of the relations. We consider that the relations higher than a certain specific rank are realistic. We call this rank *threshold.*

The degree of similarity between two partial documents is calculated by using the vector space model of the weighting by the TF-IDF method [10]. We suppose $P$ is a set of patterns which contains $N$ patterns. Then, stemming [8] and stop word removal [9] are applied to the words included in the pattern of $P$. The list of the words $T$ is obtained because of processing. Because the pattern is a combination of three partial documents, the total number of partial documents is $3N$. Let $ts(s,t)$ denote the frequency of the word $t \in T$ in a partial document $s$, and $df(t)$ denote the number of partial documents that contain the word $t$. At this time, the weighting of the word $t$ in the partial document $s$ is defined as follows.

$$w(s,t) = tf(s,t)(\log_2 \frac{3N}{df(t)} + 1).$$

**Table 2. Correspondence between sections and patterns**

| Section | Corresponding element of pattern |
|---|---|
| Pattern Name | |
| Intention | Starting Context |
| Motivation | Starting Context |
| Applicability | Force |
| Solution | |
| Consequences | Resulting Context |
| Related Patterns | |

Using the word weight, the document vector is designed as

$$\vec{tv}(s,T) = (w(s,t_1), w(s,t_2), ..., w(s,t_m)),$$

w here $t_1, t_2, \ldots, t_n$ are words in $T$. Then, we define the similarity between the partial documents $s_1$ and $s_2$ as follows.

$$sim(s_1,s_2) = \frac{\vec{tv}(s_1,T)\cdot\vec{tv}(s_2,T)}{|\vec{tv}(s_1,T)||\vec{tv}(s_2,T)|}.$$

We calculate the similarity of any pair of patterns, and extract the pairs whose similarity is more than pre-designed threshold as related patterns. Then, we let the corresponding nodes share the same node. Finally, we obtain a PRG like that shown in Figure 3.

## 4 Experiment

We implemented a system that automatically executes this analysis process, and performed the following experiments and evaluations.

**Experiment 1:** We used the *22* pattern documents in ref. [4] represented with the GoF's design pattern.

**Experiment 2:** We used the 23 pattern documents in ref. [6] represented with the GoF's design pattern.

**Experiment 3:** We used the pattern documents of experiments 1 and 2.

**Experiment 4:** We used the pattern documents of experiments 1 and 2; however, we only use HTML tag removal, the TF-IDF method, and the vector space model.

In the experiments 1 and *2,* we discarded the content of the section ***"Related Patterns",*** to evaluates whether our technique can analyze the relations among patterns without explicit information. We assume that the relations specified in the section ***"Related Patterns"*** are the correct relations. The strongest relation of the three relations per combination was assumed the representative relation.

For experiment 3 and **4,** we compared the efficiency of the two methods: our technique and a simple similarity-based technique. We assume that the relations between the documents that describe the same pattern are the correct relations.

We use two metrics. One is recall that is defined as a ratio of the number of system outputs to the number of all correct answers. **A** large recall value indicates that the leakage in the result is small. Therefore, the larger the value, he more excellent the performance of the system. The others is precision which is defined **as** a ratio of the number of correct system outputs to the number of all system outputs. **A** large value indicates that the number of unnecessary results is small. Therefore, the larger the value, the more excellent the performance of the system. The recall and the precision are in a trade-off relationship. For this reason, it is necessary to evaluate the overall result on the basis of the recall-precision graph like Figure 5.

We change the threshold from the most significant rank to the least, and obtain all pairs of recall and precision. Plotting these values to the **XY** axis enables **us** to obtain the recall-precision graph. In general, we equates the maximum precision in all pairs whose recall is higher than $x$ with the precision whose recall is $x$. Moreover, we use the 11-point average precision to summarize the obtained results. The 11 point average precision is the average of the precision in 11 points where the value of the recall is 0.0 from 1.0, that is one of the evaluation indices which is used to synthesize the recall and the precision. Using 11 point average precision, we can perform a simple comparison of recall and precision.

### 4.1 Results

The result of experiment 1 indicated 0.333 in 11 points average precision, and the result of experiment 2 indicated 0.301. Therefore, it is evident that our technique enables to analyze the relation between patterns with a certain degree of precision.

The results of experiments 1 and 2 are shown in Figure 5 as a recall-precision graph. For the relations extracted in experiment 1, Table 3 shows the strengths of the relations and ten headings, the kind of the relation, and the relation is correct or not.

We show a part of PRG in Figure 3 on the result of experiment 1. That PRG visualizes the result in Table 3. Figure 3 shows that the ***Abstract Factory, Factory Method,*** and ***Prototype*** patterns derive similar application results, and the ***Bridge*** and ***Visitor*** patterns are applicable after the *Command* pattern has been applied.

In Table 3, the relation between ***Chain*** of ***Responsibility*** and *Command* patterns is not explicitly described in original pattern documents, however, we think that this relation is valid since the following reasons. The purpose of the ***Chain*** of ***Responsibility*** pattern is to allocate the limited responsibility to each object and to achieve the responsibility by delegating the responsibility between the objects. In addition, the pattern enables the dynamic change of the processing order. The ***Command*** pattern encapsulates the responsibility of the command object. The range of responsibility of each object is reduced. **A** complex process is expressed by combining the command objects. Therefore, these patterns are similar in the purpose of reducing the responsibility.

The results of experiments 3 and **4** are shown in Figure 6 as a recall-precision graph. The result of experiment 3 indicated 0.789 in 11 points average precision, and the result of experiment 4 indicated 0.514. Therefore, it is evident that our technique is more effective in analyzing of the relations among patterns than the technique using simple similarity

| Rank | Pattern $p_1$ | Pattern $p_2$ | Strength | Kind of relation | explicitly described? |
|---|---|---|---|---|---|
| 1 | ObjectAdapter | ClassAdapter | 1.0 | starting context | Yes |
| *2* | ChainofResponsibility | Command | 0.44 | starting context | No |
| 3 | Visitor | Iterator | 0.43 | resulting context | Yes |
| 4 | FactoryMethod | Prototype | 0.42 | resulting context | No |
| 5 | State | Memento | 0.42 | resulting and starting | No |
| *6* | Command | Visitor | **0.38** | resulting and starting | No |
| 7 | AbstractFactory | FactoryMethod | 0.37 | resulting context | Yes |
| 8 | Command | Bridge | 0.35 | resulting and starting | No |
| 9 | Composite | Decorator | 0.35 | resulting context | Yes |
| 10 | Strategy | Bridge | ***0.26*** | starting context | No |

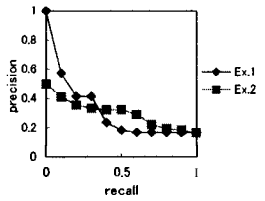


**Figure 5. Results of experiments 1 and 2**

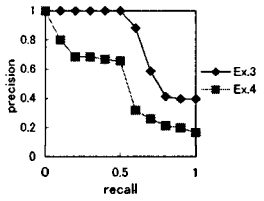


**Figure 6. Results of experiments 3 and 4**

among the documents.

## 5 Related work

Ong et al. designed the relations among forces, such as *"Helps", "Hurts"*[7]. They attempted to systematize patterns based on the relations between forces. Our technique can treat those relations automatically.

Borchers modeled a pattern as a list of sections[2]. This is common with our technique in that we design the pattern as a combination of sections. In addition, Acosta and Zambrano are trying to show a concrete pattern set as a directed graph based on the Borchers' model [1]. Borchers' model can carry precise information. However, we customized the pattern model by discarding a part of the information in order to obtain the flow of the pattern application.

Zimmer designed three types of relation among the GoF's design patterns[11]. Specifically, a consecutive application relation and a similar relation were indicated. The GoF's design patterns are systematized on a basis of the above-mentioned relation. Our pattern model can treat these relations automatically.

## 6 Conclusion

We proposed a technique for the automatic analysis of the relation of pattern documents. As a result of experiments, we succeeded in the analysis of the relations among patterns without using explicit information in the pattern document. In addition, a system which implements our technique can suggest the relations that authors have not noticed.

Our technique is expected to contribute the following objectives:

**Pattern research support:** Large-scale, automatic relation analysis and the suggestion of the relations that authors have not noticed can contribute to pattern research.

**Pattern use support:** The analysis results become a pattern repository. Highly accurate pattern retrieval becomes possible.

We plan to improve the precision of our system by applying text processing technology and natural-language processing technology in the future. Furthermore, relation analysis among many patterns is also planned.

## References


[1] A. E. Acosta and N. Zambrano. Patterns and objects for user interface construction. *Journal of Object Technology,* 3(3):75–90, 2004.

[2] J. O. Borchers. A pattern approach to interaction design. *AI & Society Journal of Human-Centred Systems and Machine Intelligence*, 15(4):359–376, 2001.

[3] F. Buschmann et al. *Pattern Oriented Software Architecture: A System of Patterns.* Wiley, New York, 1998.

[4] D. V. Camp. An object-oriented pattern digest. http://patterndigest.com/.

[5] E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[6] V. Huston. Huston design patterns. http://home.earthlink.net/~huston2/dp/patterns.html.

[7] H.-Y. Ong et al. Rewriting a pattern language to make it more expressive. ChiliPLoP2003, 2003.

[8] C. D. Paice. Another stemmer. *SIGIR Forum,*24(3):56–61, 1990.

[9] G. Salton and M. J. McGill. *Introduction to Modern Information Retrival.* McGraw-Hill, Inc., ,New York, 1983.

[10] G. Salton and C. S. Yang. On the specification of term values in automatic indexing. *Journal of Documenatation*, 29:351–372, 1973.

[11] W. Zimmer. *Relationships between Design Patterns.* Pattern Languages of Program Design, Vol.1, Addison-Wesley, 1995.