Primary inter-patterns relationships analysis

Asma HACHEMI

Computer science department
USTHB
BP 32 El Alia, Bab Ezzouar 16111 Algiers
ALGERIA
ashachemi@usthb.dz

Mohamed AHMED-NACER

Computer science department
USTHB
BP 32 El Alia, Bab Ezzouar 16111 Algiers
ALGERIA
anacer@mail.cerist.dz

ABSTRACT. The relationships between patterns allow to combine several patterns, in order to solve complex problems that are not treated by a single pattern. Unfortunately, it is difficult to identify these relations if they are not explicit in each pattern. In this case, and taking into consideration the growing number of patterns, the manual analysis of inter-patterns relationships is a daunting activity. Thus, we present in this paper our approach that improves an existing method of automatic relations analysis. This automatic method treats relationships between patterns, from different catalogs and from different forms. We extend this method so that it can treat the primary inter-patterns relationships Uses and Refines.

KEYWORDS: Inter-patterns relationship, primary relationships, relations analysis, automatic analysis, Refines, Uses.

1. Introduction

A software pattern offers a proven solution to solve a recurrent problem in the software engineering field. Concerning the complex problems which are not treated by a single pattern, the relationships between patterns allow to combine several patterns, in order to get a solution to this kind of problems. Unfortunately, it is difficult to identify inter-patterns relationships if they are not explicit in each pattern. This difficulty puts patterns into a disadvantage, since it discourages the reuse of patterns. Thus, patterns will be better served by an automatic approach that can treat any number of patterns (belonging to different collections or expressed in different forms), in order to analyze relationships between them.

We present in this paper our improvement of an automatic method which analyzes inter-patterns relationships. That method is the first automatic approach treating inter-patterns relationships on patterns from different catalogs or from different forms. We extend that method so that it can treat the primary inter-patterns relationships Uses and Refines. So, the present paper is structured as follows : we introduce the primary inter-patterns relationships in section 2. Section 3 presents a method for automatic analysis of relations. Our approach for the automatic analysis of the primary relations Uses and Refines is explained in section 4. Finally, we conclude this paper and give some research perspectives in section 5.

2. Primary relationships between software patterns

Inter-patterns relationships represent the links that may exist between patterns. They are on the basis of patterns composition. Their role is to indicate which patterns can function together and in which manner, so one can combine patterns in order to solve a problem which is not treated by a single pattern. Primary relationships between patterns [1] are ubiquitous relations in the literature, are on the basis of the definition of other relations and their definitions are straightforward [1]. Thus, primary relations must have priority (over other relations) in any work dealing with inter-patterns relationships.

Many researchers focused their interests on inter-patterns relationships :

– W. Zimmer [2] is the first to define relations between object oriented design patterns ; he used in his work the GoF patterns [3]. Other studies have followed, like [4], [5], [6], [7], [8], [9] and [10], which were also interested in inter-patterns relationships. However, all those works define the relationships but cannot extract them if they are not explicitly expressed in patterns.

– Prabhakar *et al.* propose a graphical model called DDTM [11] based on Design Decisions [12] [13], in order to represent design patterns and analyze relationships

between them. However, this work is limited to design patterns only (it cannot be applied on other kind of software patterns).
– The method of Kubo *et al.* [14] is the first automatic approach able to analyze relations between patterns, without being limited to a particular kind of patterns. We are particularly interested in this method (and present it in the following section), because it is the only approach able to analyze relations on patterns belonging to different catalogs and on patterns expressed in different forms. Moreover, it is the only method dealing with any kind of patterns and treating process patterns [15].

3. A method for automatic analysis of relations

Kubo *et al.* method [14] is based on a new pattern model, uses many text processing techniques, and uses the cosine similarity to analyze relations between patterns. It represents a pattern application as a context transition from a starting context to a resulting context, and includes the pattern forces in the model (because patterns that differ only in term of forces are considered as different patterns). It targets patterns described with HTML, and considers seven types of relations between patterns [14] [16], namely : Starting-Starting, Resulting-Resulting, Resulting-Starting, Same, SubInStarting, SubInResulting, SimilarForce.

However, the method [14] presents the following main drawback : it analyzes little known relationships like *Resulting-Resulting* [14], *SubInStarting* [16], *SubInResulting* [16], *SimilarForce* [16], whereas the two primary relationships Uses and Refines can't be treated.

– *Uses* is a very important relationship which is defined in many works like [2], [5], [6], [8], [9] [10], [11], [17], [19]. It links many patterns, for instance : *Distinguish Identities* pattern uses *Component Proxy* pattern, *Component Home* pattern and *Managed Resource* pattern [17]. The process pattern *Review* uses three other process patterns, namely *Introductional Session* pattern, *Review Session* pattern and *Release* pattern [8]. Unfortunately, the relation *Uses* can't be analyzed by [14].

– *Refines* relates many patterns, for example : Two refinements of the process pattern *Capture A Common Vocabulary* exist, namely the *Capture Vocabulary Centrally* pattern and the *Capture Vocabulary Participatorily* pattern [8]. The *Iterator* pattern [3], the *Type-safe Session* pattern [20] and the *Accumulator* pattern [21] all refine [1] the *Curried Object* abstract pattern [22]. The *Refines* primary relationship is defined in several works like [6], [7], [8], [9] [10], [11], [17], but can't be analyzed by [14].

4. Our approach

Our approach towards an automatic way to analyze the primary relations between patterns is based on the Kubo *et al.* method [14]. This latter is based on several text processing techniques (stop word removal [23], stemming [24], the TFIDF term weighting method [25], vector space model [23] and the cosine similarity). Unfortunately, the analysis of the primary relations *Uses* and *Refines* is not possible with [14], as explained earlier. Thus, an added value of our approach is the possibility to analyze the primary relations *Uses* and *Refines*.

Our approach uses the pattern model of [14], that enables us to treat most heterogeneous patterns (expressed in different forms) automatically, and uses an auxiliary which is the analysis of the *Inclusion*.

The existence of *Inclusion* between two texts means that in addition to being similar, one of these texts contains the other. Let's have two texts T1 and T2 consisting of one or more terms. T1 *is Included in* T2 means that the following two conditions are true :

– T1 is similar to T2, to signify that the terms of T1 exist in T2, which means that T2 treats of T1 subject.

– T2 is larger than T1, to signify that T2 contains T1 in addition to another content.

For example, let's have these two texts : *"Provide an interface for creating families of related or dependent objects without specifying their concrete classes"* and *"Providing an interface to create objects"*. We calculate the *Inclusion* between them as follows : First of all, we calculate the size of each text after eliminating stop words. We obtain Size of Text1 = 9 terms, Size of Text2 = 4 terms. Then, we compare the sizes and we notice that the first text is larger than the second one by 0,550 (the ratio of the sizes difference to the size of the largest text). After that, we calculate the cosine similarity between the two texts. We obtain the value 0,537. Finally, given that those two texts are similar (their similarity value is larger than the Similarity threshold) and the first one is larger than the second (their difference of sizes is larger than the Sizes Difference threshold), then we conclude that the first text *Includes* the second.

4.1. Analysis of the relation Uses

The *Uses* relationship is defined as follows. P1 and P2 are two patterns expressed in the model of Kubo *et al.*. P1 *Uses* P2 means that the following two conditions are true :

– The P2 Starting Context *is Included in* the P1 Starting Context, to mean that the problem addressed by P2 is a sub problem of that treated by P1.

– The P2 Resulting Context *is Included in* the P1 Resulting Context, to mean that the result produced by applying the pattern P2 is a sub set of the result produced by the application of pattern P1.

For example, we consider the pattern *Code Ownership* [26] (called P2) which *Uses* the pattern *Review* [26] (called P1). This relationship is given by the author of these patterns, so we consider it as correct and process the analysis using our method.
First, we compare the different elements of the two patterns P1 and P2. We note SC the element Starting Context and RC the element Resulting Context of a pattern. We obtain the following results :

Compared Elements	Results
SC of P1 and SC of P2	Similarity = 0,095
	SC of P1 includes SC of P2 = False
	SC of P2 includes SC of P1 = True
RC of P1 and RC of P2	Similarity = 0,240
	RC of P1 includes RC of P2 = False
	RC of P2 includes RC of P1 = True
Forces of P1 and Forces of P2	Similarity = 0,127
	Forces of P1 includes Forces of P2 = False
	Forces of P2 includes Forces of P1 = False
RC of P1 and SC of P2	Similarity = 0,040
RC of P2 and SC of P1	Similarity = 0

Table 1. Results of comparisons between patterns elements

Then, we calculate from Table 1 the value of each relationship between those patterns. For the relationships Same, Starting-Starting and Resulting-Starting, we uses the analysis method of Kubo *et al.*. Whereas for the relationships Uses and Refines, we exploit our propositions given respectively in the previous and following paragraphs. We obtain the following results :

Relationship	Its value
P1 Uses P2	0
P1 Refines P2	0
P2 Uses P1	0,167
P2 Refines P1	0
Same	0,167
Starting-Starting	0,095
Resulting-Starting (P1 then P2)	0
Resulting-Starting (P2 then P1)	0

Table 2. Results of relations analysis

Finally, as in [14] the strongest relation among the eight types (P1 Uses P2, P1 Refines P2, P2 Uses P1, P2 Refines P1, Same, Starting-Starting, Resulting-Starting (P1 then P2), Resulting-Starting (P2 then P1)) is assumed as the representative relationship. So the pattern P2 *Uses* the pattern P1.

4.2. Analysis of the relation Refines

The *Refines* relationship is defined as follows. P1 and P2 are two patterns expressed in the model of Kubo *et al.*. P2 *Refines* P1 means that the following tree conditions are true :

– The P2 Starting Context is similar to the P1 Starting Context, to mean that both patterns P1 and P2 deal with the same problem.
– The P1 Starting Context *doesn't Include* the P2 Starting Context, to insure that the pattern P1 (the pattern being refined) doesn't deal with a problem larger than the one dealt by the pattern P2.
– The P1 Forces *is Included in* the P2 Forces, to mean that constraints imposed on pattern P1 represent a sub set of the constraints imposed on the pattern P2.

For example, we consider the pattern *Head-Body* [18] (called P1) which *Refines* the pattern *Separate Metadata and Data* [18] (called P2). This relationship is given by the author of these patterns, so we consider it as correct and process the analysis using our method. We compare the elements of P1 and P2 and obtain the following results:

Compared Elements	Results
SC of P1 and SC of P2	Similarity = 0,342
	SC of P1 includes SC of P2 = False
	SC of P2 includes SC of P1 = False
RC of P1 and RC of P2	Similarity = 0,040
	RC of P1 includes RC of P2 = False
	RC of P2 includes RC of P1 = True
Forces of P1 and Forces of P2	Similarity = 0,746
	Forces of P1 includes Forces of P2 = True
	Forces of P2 includes Forces of P1 = False
RC of P1 and SC of P2	Similarity = 0,056
RC of P2 and SC of P1	Similarity = 0,070

Table 3. Results of comparisons between patterns elements

After that, we calculate the value of each relationship between those patterns using Table 3. The results are as follows :

Relationship	Its value
P1 Uses P2	0
P1 Refines P2	0,554
P2 Uses P1	0
P2 Refines P1	0
Same	0
Starting-Starting	0,342
Resulting-Starting (P1 then P2)	0
Resulting-Starting (P2 then P1)	0,070

Table 4. Results of relations analysis

Finally, considering the strongest relationship, we conclude that the pattern P1 *Refines* the pattern P2.

5. Conclusion and perspectives

We presented in this paper our approach that analyzes primary relationships between patterns. Our contribution is based on the automatic method of Kubo et al. and on the analysis of *Inclusion*, in order to extract the primary relationships.

Some improvements are under work to ameliorate our approach of relations analysis. Mainly :

– The vector space model used for the calculation of similarity assumes independence between terms, which is not always true because two different terms may be synonymous. So the method can be extended to recognize synonyms.

– Our contribution can be extended to offer the functionality of Patterns Retrieval, which offers for a particular problem all available patterns that treat it.

6. References

[1] J. Noble, “Classifying Relationships Between Object-Oriented Design Patterns,” *Australian Software Engineering Conference, Adelaide*, South Australia, November 09-13, 1998.

[2] W. Zimmer, “*Relationships between design patterns*,” In Pattern Languages of Program Design, Addison-Wesley, 1994.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “*Design patterns: elements of reusable object-oriented software*,” Addison-Wesley, 1995.

[4] R. Deneckere, C. Souveyet, “Organising and selecting patterns in pattern languages with Process Maps,” *OOIS'01*, Canada, 2001.

[5] M. Gnatz, F. Marschall, G. Popp, A. Rausch, W. Schwerin, “The Living Software Development Process,” *Journal Software Quality Professional*, Volume 5, Issue 3, June 2003.

[6] G. Meszaros, J. Doble, “A Pattern Language for Pattern Writing,” *PLoP*, 1997.

[7] K. Henney, “Patterns Inside Out,” Talk presented at Application Development, London, 1999.

[8] M. Hagen, V. Gruhn, “Process patterns - A means to describe processes in a flexible way,” *International Workshop on Software Process Simulation and Modeling*, Edinburgh, U K, 2004.

[9] D. Rieu, J.P. Giraudin, C. Saint Marcel, A. Front-Conte, « Des opérations et des relations pour les patrons de conception, » *Inforsid’99*, 1999.

[10] A. Conte, M. Fredj, J.P. Giraudin, D. Rieu, « P-Sigma : un formalisme pour une représentation unifiée de patrons, » *Inforsid*, Genève, 2001.

[11] T.V. Prabhakar, K. Kumar, “Design Decision Topology Model for Pattern Relationship Analysis,” *Asian PLOP* 2010 Tokyo, 16-17 March 2010.

[12] L. Bass, P. Clements, R. Kazman, “*Software Architecture in Practice*,” AddisonWesley, 2003.

[13] P. Kruchten, “An ontology of architectural design decisions in software intensive systems,” In *2nd Groningen Workshop on Software Variability*, pp. 54-61, December 2004.

[14] A. Kubo, H. Washizaki, A. Takasu, Y. Fukazawa, “Analyzing Relations among Software Patterns based on Document Similarity,” Proc. *IEEE Internationale Conference on Information Technology : Coding and Computing*, 2005.

[15] H. Washizaki, A. Kubo, A. Takasu, Y. Fukazawa, “Relation analysis among patterns of software development process,” *PROFES 2005*, LNCS 3547, pp. 299-313, 2005.

[16] H. Washizaki, A. Kubo, A. Takasu, Y. Fukazawa, “Extracting Relations among Security Patterns,” *1st International Workshop on Software Patterns and Quality* SPAQu07, 2007.

[17] M. Volter, “Server-side components - A pattern language,” In proceedings of *EuroPLoP*, 2000.

[18] “Develop effective XML documents using structural design patterns,” http://www.xmlpatterns.com/

[19] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, M. Stal, “*Pattern-oriented software architecture - A System of Patterns*,” Volume 1, Wiley and Sons, 1996.

[20] N. Pryce, “Type-safe session,” In *EuroPLOP* Proceedings, 1997.

[21] P. M. Yelland, “Creating host compliance in a portable framework: A study in the use of existing design patterns,” In *OOPSLA* Proceedings, 1996.

[22] J. Noble, “Arguments and results,” In *PLOP* Proceedings, 1997.

[23] G. Salton, M. McGill, *Introduction to Modern Information Retrival*, McGraw-Hill Inc., N Y, 1983.

[24] C. Paice, “Another stemmer,” *SIGIR Forum*, Vol. 24, No. 3, pp. 56–61, 1990.

[25] G. Salton, C. Yang, “On the specification of term values in automatic indexing,” *Journal of Documentation,* Vol. 29, pp. 351–372, 1973.

[26] J.O. Coplien, “A Development Process Generative Pattern Language,” In Proceedings of the *Annual Conference on the Pattern Languages of Programs* PLoP’94, 1994.