Exploring the Redistribution Classes of a Package with an Approach Based on Formal Concept Analysis.

Lala Madiha Hakik
Faculty of Sciences and Techniques,
University Hassan I, BP 577. Settat, Morocco,

Rachid El Harti
Faculty of Sciences and Techniques,
University Hassan I, BP 577. Settat, Morocco,

Abstract

During software evolution, the software structure, which mainly consists of modular abstractions (such as packages) erodes more or less slowly. Modernizing the software structure, with remodularization approaches, is an important issue with several variants.

In this paper we explore the issue of redistributing classes of a package to other packages. We use an approach based on Formal Concept Analysis to determine the packages that receive the redistributed classes.

Keyword: Software Modularization, Formal Concept Analysis.

1. Introduction

Great software systems based on approaches, the object consist of classes grouped into packages, forming a modular structure.

The dependency relationships between classes in the same package (internal dependencies), and between classes of different packages (external dependencies generate complexity making it difficult to understand and maintain the system.

In addition, the modular structure tends to degrade over time, making necessary an expert intervention for modernization.

Many researchers make proposals on this subject using technical visualization or algorithms of remodularization.

In this paper, we study a particular declination, cf. the problem presented by H. Abdeen et al. [2] [1], which is about the redistribution of classes from one system to existing packages. Namely, we consider in this paper more precisely the redistribution of classes in a package to other packages.

This package may be a very small and in fact we want to balance the sizes of packages in the system, or it was artificially created to contain added classes to the system and the designer considers that there is no consistency semantics.

In a context of restructuring architecture or development, it can also be linked to the disappearance of a node in a distributed architecture or recentralization and / or refactoring for maintenance reasons.

We explore a solution using the Formal Concept Analysis (FCA) and illustrate our proposal with a theoretical example.

Section 2 presents our example, then we describe the approach in Section 3. Related work is presented in Section 4, and then we conclude in Section 5.

2. Illustration

This section presents the problem of software architectures remodularization on an example. We will use the architecture shown in Figure 1 consists of five packages A, B, C, D and E. Packages A, B, C, D, E are expected to contain more classes that are not shown for simplicity. Dependencies linking classes: they correspond for example to call a method or use of a type. External dependency relationships link classes of package E to classes of other packages. Internal dependency relationships connect classes E between. Internal dependencies of A, B, C and D are not presented.

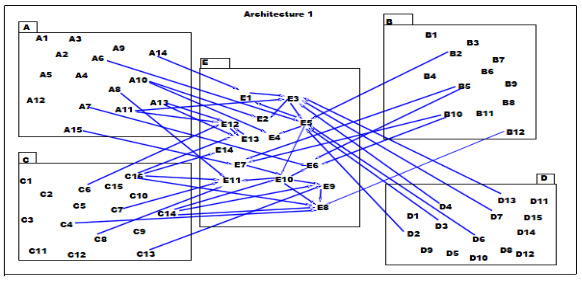

Architecture 1
A
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
B
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12
C
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16
D
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
E
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13 E14

Figure 1. An initial architecture composed of classes and packages.

We are interested in the redistribution of classes E to other packages with an exploratory method, whose proposals for redistribution are then presented to an expert. These proposals are based on the idea that the expert, while checking the semantic classes, could search for the increase of the cohesion (within the meaning of the coupling of classes in a package) and reduce the coupling between classes in different packages. To do this, we believe it is appropriate to encourage the following two trends:

- Classes in a package attract them to classes of E,
- If classes of E are interconnected, it is better to redistribute in the same package.

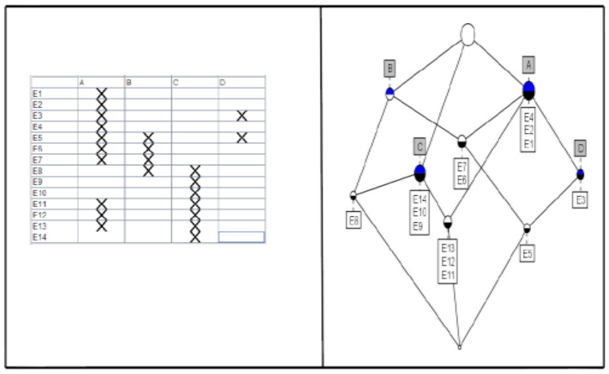
We believe that the Formal Concept Analysis (FCA) can bring interesting ways to solve this problem because this technical method allows the group to connect classes identically. We are not looking here to propose a better solution; but offer to an expert different hierarchical solutions.

3. Proposed approach

The Formal Concept Analysis (FCA) [6] is a technical data analysis that allows you to group entities with common characteristics. A concept is a maximal set of entities (extension of the concept) sharing a maximal set of characteristics (intension of the concept). The AFC is used in software engineering for solving several problems [11].

Configurations In the context of our problem, we studied five different configurations with FCA.
We present two of them.
The configuration with FCA is to define a formal context C: the set O of entities studied (or formal objects) Set A of characteristics (or formal attributes) and the relationship
$R \subseteq O \times A$.
The first formal context associates a class c of a package E to the packages that access to this class c (see Figure 2, left panel).

Context (formal context C2).
- O2 is the set of classes of E in relation to the outside.
- A2 is the set of packages A, B, C, D (which has a relation to a class of E).
- R2 is the relation "is a target for external access".
- (e, p) ∈ R2 if e is an access target from p, for example (E2, A) ∈ R2.

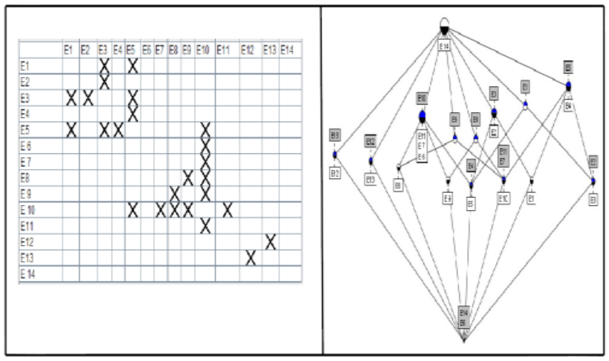


	A	B	C	D
E1	X			
E2	X			
E3	X			X
E4	X			
E5	X	X		X
F6	X	X		
E7	X	X		
E8		X	X	
E9			X	
E10			X	
E11	X		X	
F12	X		X	
E13	X		X	
E14			X	

Figure 2. Formal context C2 and lattice T(C2) - Architecture 1-

The second formal context can refine the results and redistribute the same package into two classes that are interconnected in E). It combines a class of package E another class that is connected (see Figure 3, left panel).

Context (formal context C5).
- O5 is the set of classes of E in relation to the outside.
- A5 = O5: E classes in relation to the outside.
- R5 is the relation "is connected to".
- (e1, e2) ∈ R5 if there is an arrow e1 to e2 or e1 to e2, for example (E4, E5) and (E5, E4) belong to R2.



	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14
E1			X		X									
E2			X											
E3	X	X			X									
E4					X									
E5	X		X	X						X				
E6										X				
E7										X				
E8									X	X				
E9								X		X				
E10					X		X	X	X		X			
E11										X				
E12													X	
E13												X		
E14														

Figure 3. Formal context C5 and lattice T(C5) - Architecture 1-

The concept lattice is the classification structures that expose concepts (their nodes) and link by specialization.
For example, the concept lattice T(C2) associated with context C2 (see Figure 2, right), contains eight concepts outside the top and bottom. The shaded part of the labels (upper part) corresponds to the simple intension of the concept, while the white portion of the label (lower part) is a simplified extension. Labeled extensions are inherited backwards in the lattice while labels intensions are inherited in descending.

For example the lattice T (C2) contains the concepts:
- ({E6, E7, E8}, {B}) at the top left, simplified in ({}, {B})
- ({E11, E12, E13}, {A, C}) in the middle at the bottom, simplified ({E11, E12, E13},{})

Example of exploration The exploration is to navigate the two lattices T (C2) and T (C5) to identify opportunities for redistribution of classes and submit to an expert. We partially detail an example of analysis to explain the principle.
The lattice T (C5) can be divided into three large blocks in which we will choose concepts.

1. Analysis of the concept ({E1, E3, E4}, {E5}) the right of T (C5): the extension of the concept is in the extension of the concept (Simplified) ({E1, E2, E4}, {A}) T (C2), and E5 is also connected to A, the expert can choose to put three classes E1, E3, E4 in A.
2. Analysis of the concept ({E3}, {E2, E5}) the right of T (C5) three classes are in full extension of the concept of intension {A} of T (C2), the expert can still choose to put them in A. The subsystem {E1, E2, E3, E4, E5} can be put into A.
3. Analysis of the concepts of right ({E12}, {E13}) and ({E13}, {E12}) T (C5). In T (C2) {E12, E13} is in the extension of the concept of intension {A, C} which indicates us the two possible solutions. The expert can choose of place all E12 and E13 in A or C, but it will avoid of place E12 to E13 in A and C. This will lead to two possible architectures of Figure 4.
4. In the center very interspersed of T (C5), the expert chooses a concept of low ({E10}, {E5, E7, E8, E9, E11}). Analysis of T (C2) shows that the majority of these classes is drawn in C.
5. The expert examines the concept ({E8}, {E9, E10}) T (C5). Its intension is in the extension of the concept of intension {C} which tends to place also the class E8 in C.

Figure 4 shows two possible results. The concepts of T (C5) have informed us on internal cohesion to package E, while the structure of redistribution classes of E is accessed in T (C2) and informs us about the potential coupling.

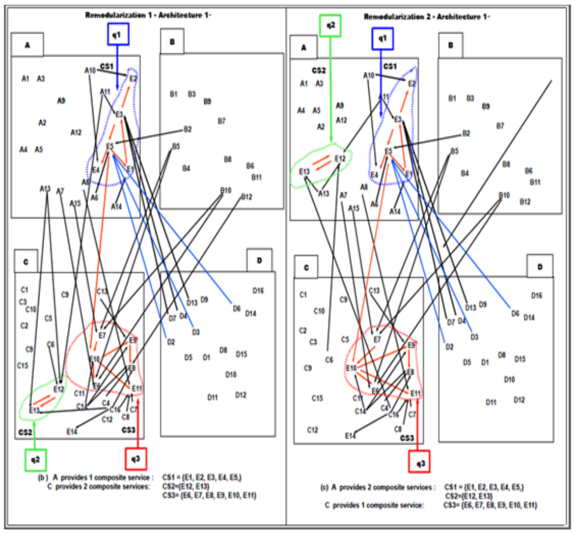

Remodularization 1 - Architecture 1-
Remodularization 2 - Architecture 1-
q1
q2
q3
CS1
CS2
CS3
A
B
C
D
(b) A provides 1 composite service : C provides 2 composite services:
CS1 = {E1, E2, E3, E4, E5}
CS2={E12, E13}
CS3= {E6, E7, E8, E9, E10, E11}
(c) A provides 2 composite services : C provides 1 composite service:

Figure 4. two possibilities of remodularization

4. Related Work

Different automated approaches have been proposed to restructure object systems. We cite three: the clustering algorithms, algorithms based on meta - heuristics and those based on the FCA. The first aim to restructure system by the distribution of some elements (eg classes, methods , attributes) in groups such that the elements of a group are more similar to each other with elements of other groups [3] [7] [5]. Approaches to restructuring based on meta-heuristic algorithms [9] [8] are generally iterative stochastic algorithms, progressing towards a global optimum of a function by evaluating a certain objective function (eg characteristics or quality metrics). Finally, the approaches based on FCA [10] [12] provide an algebraic derivation of hierarchies of abstractions from all entities of a system. Reference [4] presents a general approach for the application of the FCA in the field of object-oriented software reengineering. In our approach, we add the dimension of exploration using the FCA.

6. Conclusion and discussion

In this article, part of preparation a Phd thesis in software engineering, has been the subject of a communication in the conference [13], where we presented and illustrated a theoretical case and proposed to explore the redistribution of classes in a package, based on the FCA. There are still many issues to this first reflection. Lattices contain a lot of information to exploit: can be observed in T (C2) that all classes of E connected to classes of D must also be connected with classes of A. Arcs could be valued to refine the forces of attraction of a class on another class. Otherwise some classes of the package E cannot be connected to the outside. In this case we can repeat the analysis. The method can be guided and evaluated by metrics and visualization technique as proposed in [1].

7. References

[1] H. Abdeen, Visualizing, Assessing and Re-Modularizing Object-Oriented Architectural Elements. PhD thesis, Lille University, 2009.
[2] H. Abdeen, S. Ducass, H.A. Sahraoui, and I. Alloui. Automatic package coupling and cycle minimization. In International Woking Confence on Reverse Engineeing (WCRE), pages 103-112, Washington, DC, USA, 2009. IEEE Compter Society Press.
[3] F.B. Abreu, G. Pereira, and P. Sousa. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In Proceeding of the confeence on Software Maintenance and Reengineering. CSMR 'OO, pages 13-, Washington, DC, USA, 2000. IEEE Compter Society Press.
[4] G. Arévalo, S. Ducass, and O. Nierstrasz. Lessons leaned in appling fomal concept analysis to reverse engineering. In Proceeding of the Third international conference on Fomal Concept Analysis, ICFCA'05, pages 95-112, Berlin. Heidelberg, 2005. Spinge-Velag.
[5] M. Bauer and M. Trifu. Architecture-aware adaptive clustering of oo s ystems. In Poceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR 'O4), CSMR 'O4, pages 3-, Washington, DC, USA, 2004. IEEE Compter Society.
[6] B. Ganter and R. Wille. Formal Concept Analysis. Mathematical Fondations. Spinge. 1999.
[7] B.S. Mitchell and S. Mancoridis. Compaing the decompositions produced by software clustering algoithms using similarity measurements. In ICSM, pages 744-753, 2001.
[8] M.O'Keeffe and M. i Cinneide. Seach-based refactoring fo software maintenance. J. Syst. Softw., 81(4): 502-216, April 2008.
[9] O. Seng, J. Stammel and D. Burkhart. Search- based determination of refactorings for improving the class structure of object-oriented systems, In Mike Cattolico, edito. GECCO, pages 1909-1916. ACM, 2006.
[10] G.Snelting. Software reengineering based on concept lattices. In CSMR, pages 3-10, 2000.
[11] T. Tilley, R. Cole, P. Becker, P.W. Eklund. A survey of formal concept analysis support for software engineering activities. In Int. Conf. Fomal Concept Analysis (ICFCA 2005), pages 250-271, 2005.
[12] P. Tonella.Concept analysis for module restructuring. IEEE Trans. Software Eng..27 (4): 351-363, 2001.
[13] Lala Madiha Hakik, Marianne Huchard, Rachid El Harti et Abdelhak Djamel Seriai. Exploration de la redistribution des classes d'un package par des techniques d'Analyse Formelle de Concepts. The first conference in software engineering (CIEL 2012), France, 2012.