Measuring Coupling and Cohesion to Evaluate the Quality of a Remodularized Software Architecture Result of an Approach Based on Formal Concept Analysis

Lala Madiha Hakik†, Rachid El Harti††

† Faculty of Science and Techniques, University Hassan I, BP 577, Settat, Morocco
†† Faculty of Science and Techniques, University Hassan I, BP 577, Settat, Morocco

Summary
Existing approaches of remodularized software systems often change the initial structure of software packages. These approaches do not take into account the original organization of classes or package size, and it can be difficult for a software expert to understand the result and know the way back to the initial state structure. Initial remodularized architecture by exploring the redistribution classes of a package with an approach based on formal Concept Analysis, it allows the software expert to know the way back and it offers an alternative remodularization. This paper presents the metrics concerning our approach in [13] and reveals interesting results for the cohesion and the coupling measuring.

Keyword:
Metrics of cohesion, Metrics of coupling, Remodularization, Software architecture, FCA.

1. Introduction

For the calculation of metrics coupling and cohesion, we conducted the remodularization architecture comprising 5 packages A, B, C, D and E by redistribution classes of package E using formal concept analysis techniques which resulted in two possible remodularized architectures (the package E is removed during this operation).
Initial architecture (figure 5) and the two architectures (figure 6) result from the remodularization obtained by applying our approach, which has been the object of the article [12][13].
The calculation of metrics of coupling and cohesion was tested on other examples, but for simplicity, we limited in one case that we treat throughout this paper.
Section 2 presents our example, then we describe the approach in Section 3.
In Section 4 of this paper, we present the validation of the metric of cohesion and coupling measure, the metrics of the new packages to support the implementation of remodularization was inspired by [1]. In Sections 5, we discuss our main results. Related work is presented in Section 6, and then we conclude in Section 7.

Manuscript received January 5, 2014
Manuscript revised January 20, 2014

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.

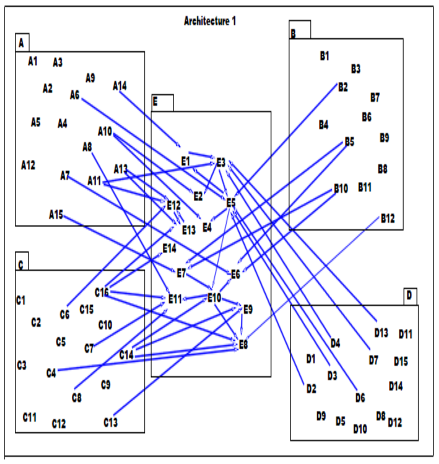


Figure 1. An initial architecture composed of classes and packages.

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) [12][13] [16] 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 [12][13] [17].

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) $\in$ R2 if e is an access target from p, for example (E2, A) $\in$ R2.

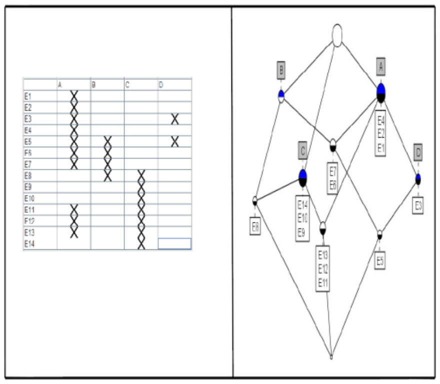


Figure 2. Formal context C2 and lattice T(C2) –Architecture 1-[12][13].

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) $\in$ R5 if there is an arrow e1 to e2 or e1 to e2, for example (E4, E5) and (E5, E4) belong to R2.

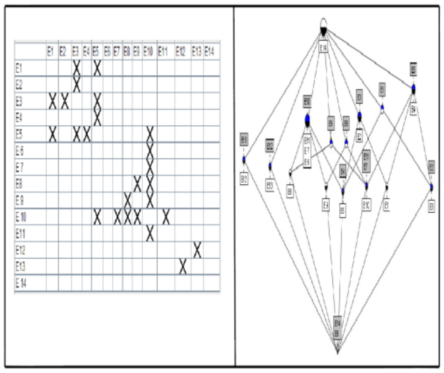


Figure 3. Formal context C5 and lattice T(C5) -Architecture 1-[12][13].

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},{})

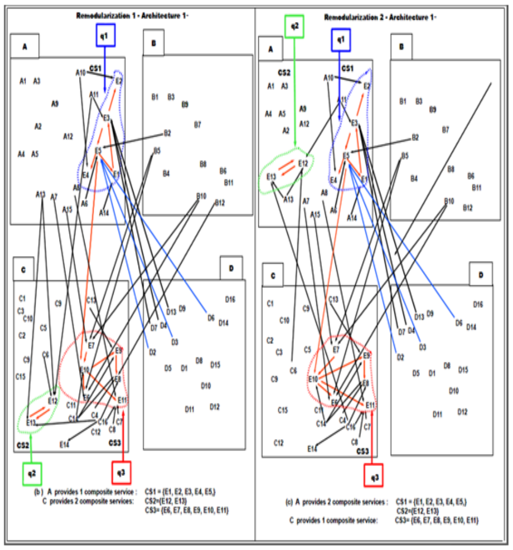
Example of exploration The exploration is to navigate the two lattices T (C2) and T (C5) to identify opportunities for redistribution of classes and submit to an expert. We partially detail an example of analysis to explain the principle.

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.

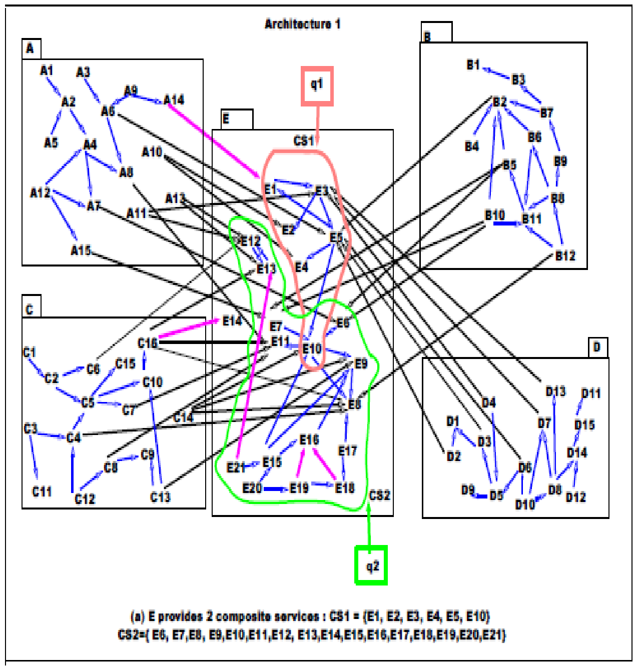
2. Analysis of the concept ({E3}, {E2, E5}) the right of T (C5) three classes are in full extension of the concept of intension {A} of T (C2), the expert can still choose to put them in A. The subsystem {E1, E2, E3, E4, E5} can be put into A.
3. Analysis of the concepts of right ({E12}, {E13}) and ({E13}, {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.

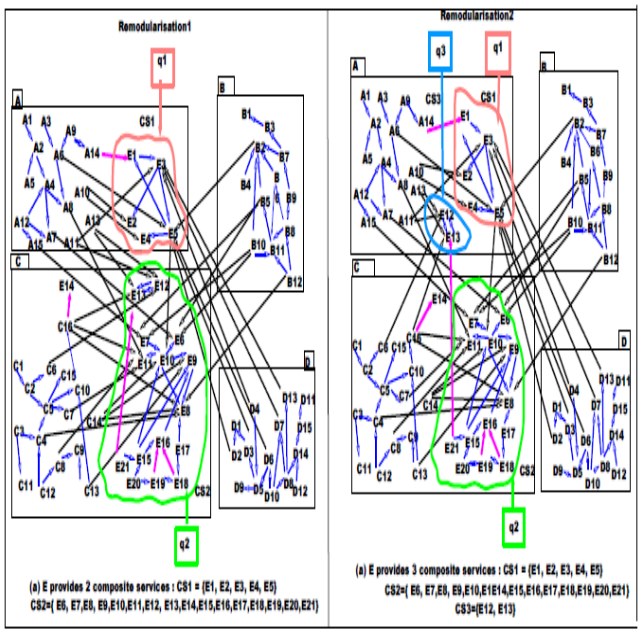
Figure 4 shows two possible results. The concepts of T (C5) have informed us on internal cohesion to package E, while the structure of redistribution classes of E is accessed in T (C2) and informs us about the potential coupling.


Remodularization 1 - Architecture 1-
Remodularization 2 - Architecture 1-
(b) A provides 1 composite service: CS1 = {E1, E2, E3, E4, E5}
C provides 2 composite services: CS2={E12, E13} CS3= {E6, E7, E8, E9, E10, E11}
(c) A provides 2 composite services: CS1 = {E1, E2, E3, E4, E5} CS2={E12, E13}
C provides 1 composite service: CS3= {E6, E7, E8, E9, E10, E11}
Figure 4. two possibilities of remodularization[12][13].

4. Validation metrics

As a reminder, for validatiton of metrics cohesion and coupling, our calculations were based on figures 1 and 2 with an architecture comprising 5 packages A, B, C, D and E by redistribution classes of package E (figure 5) using formal concept analysis techniques which resulted into two possible architectures (figure6). The package E is removed during this operation. Initial architecture (figure 1) and the two architectures (figure 6) result from the remodularization obtained by applying our approach, which has been the object of the article [12][13].


Architecture 1
A
B
C
D
E
q1
q2
CS1
CS2
(a) E provides 2 composite services : CS1 = {E1, E2, E3, E4, E5, E10}
CS2={ E6, E7,E8, E9,E10,E11,E12, E13,E14,E15,E16,E17,E18,E19,E20,E21}
Figure 5. Architecture -1-


Remodularisation1
Remodularisation2
q1
q2
q3
CS1
CS2
CS3
(a) E provides 2 composite services : CS1 = {E1, E2, E3, E4, E5}
CS2={ E6, E7,E8, E9,E10,E11,E12, E13,E14,E15,E16,E17,E18,E19,E20,E21}
(a) E provides 3 composite services : CS1 = {E1, E2, E3, E4, E5}
CS2={ E6, E7,E8, E9,E10,E11,E14,E15,E16,E17,E18,E19,E20,E21}
CS3={E12, E13}
Figure 6. Remodularization 1 and 2 -Architecture 1-

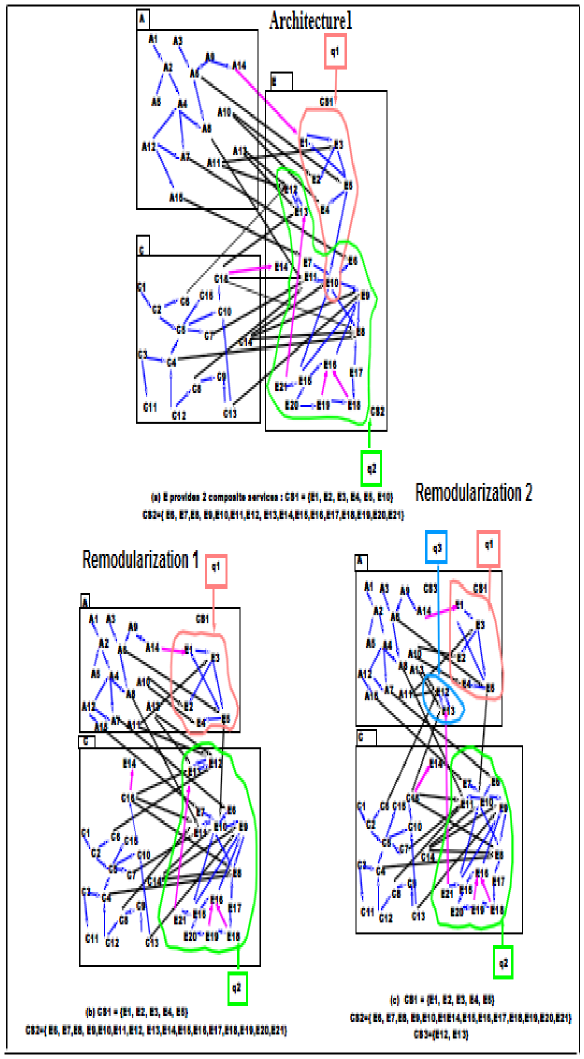

Architecture1
q1
CS1
q2
(a) E provides 2 composite services : CS1 = {E1, E2, E3, E4, E5, E10}
CS2={ E6, E7,E8, E9,E10,E11,E12, E13,E14,E15,E16,E17,E18,E19,E20,E21}
Remodularization 2
Remodularization 1
q3
(b) CS1 = {E1, E2, E3, E4, E5}
CS2={ E6, E7,E8, E9,E10,E11,E12, E13,E14,E15,E16,E17,E18,E19,E20,E21}
(c) CS1 = {E1, E2, E3, E4, E5}
CS2={ E6, E7,E8, E9,E10,E11,E14,E15,E16,E17,E18,E19,E20,E21}
CS3={E12, E13}

Figure 7. Explanation for Package Service Cohesion.

4.1 Cohesion metics

Table 1. Cohesion metrics: Index of Package Goal Focus and Index of Package Services Cohesion.

	PF	IPSC
Package E of the original architecture 1	0,5	0,0116
Package A of the original architecture 1	0	1
Package C of the original architecture 1	0	1
Package A of the remodularization 1	0,25	1
Package C of the remodularization 1	0,46	1
Package A of the remodularization 2	0,159	1
Package C of the remodularization 2	0,433	1

For the cohesion metrics contained in tables 1 , they are the result of the numerical implementation of formulas cited in [1].

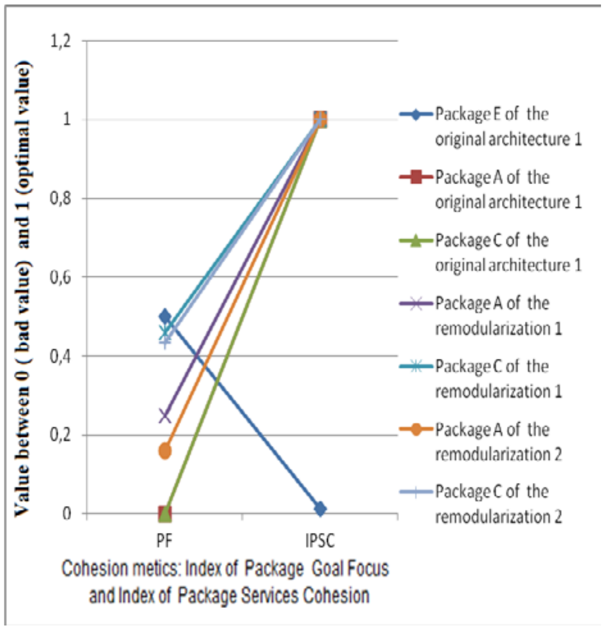


Figure 8. graphic representation of Cohesion metics: Index of Package Goal Focus and Index of Package Services Cohesion (table1

4.2 Coupling metrics

Table 2. Coupling metrics: Index of Inter-Package Interaction (IIPU and IIPE)

	IIPU	IIPE
The original architecture 1	0,588	0,333
Architecture of the remodularization 1	0,811	1
Architecture of the remodularization 2	0,811	0,75

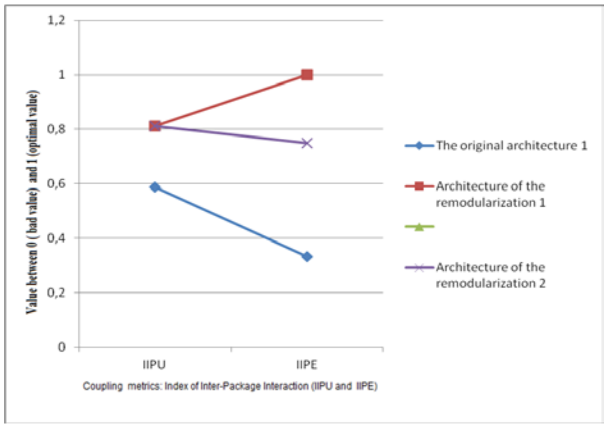


Figure 9. graphic representation of Coupling metrics: Index of Inter-Package IIPU and IIPE (table2).

Table 3. Coupling metrics: Index of Package changing Impact IPCI; Index of Package Communications Diversion (IIPUD and IIPED)

	IPCI	IIPUD	IIPED
Package E of the original architecture 1	0	0,271	1
Package A of the original architecture 1	1	1	1
Package B of the original architecture 1	1	1	1
Package C of the original architecture 1	1	1	1
Package D of the original architecture 1	1	1	1
The original architecture 1	0,8	0,854	1
Package A of the remodularization 1	0	0,38	0,38
Package B of the remodularization 1	1	0,583	0,583
Package C of the remodularization 1	1	0,541	0,541
Package D of the remodularization 1	1	1	1
Remodularization 1	0,75	0,626	0,626
Package A of the remodularization 2	0	0,384	0,38
Package B of the remodularization 2	1	0,583	0,583
Package C of the remodularization 2	1	0,541	0,538
Package D of the remodularization 2	1	1	1
Remodularization 2	0,75	0,627	0,625

For the coupling metrics contained in tables 2 and 3, they are the result of the numerical implementation of formulas cited in [1].

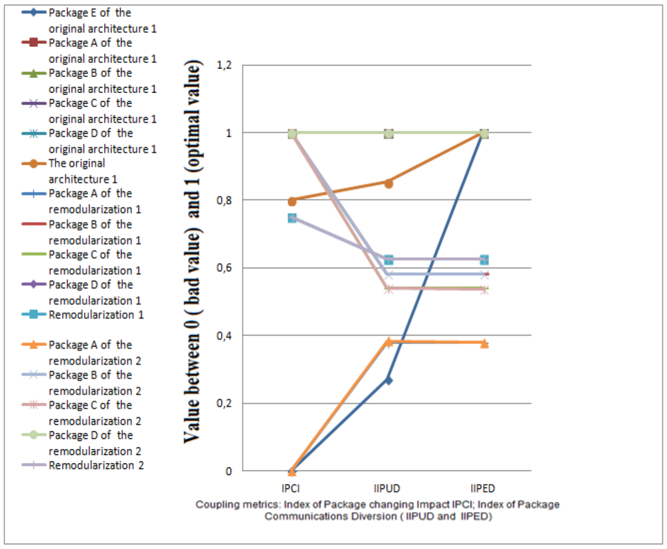


Figure 10. graphic representation of Coupling metrics: Index of Package changing Impact; Index of Package Communications Diversion (table3).

5. Results and discussion

The Cohesion metrics: Index of Package Goal Focus (**PF**) and Index of Package Services Cohesion (**IPSC**) take their values from 0 to 1, where 1 is the optimal value and 0 is the wrong value.

Figure 4 gives the values of indices PF and IPSC for:

- Package E of Original Architecture 1 whose indexes are bad values because they are lower than 1.
- Packages A and C respectively of remodularization 1 and 2 whose the index IPSC is optimal value 1 therefore very good.

The coupling metrics: Index of Inter-Package Interaction (**IIPU** and **IIPE**) object of the figure 5, it is observed an improvement indexes IIPU and IIPE at remodularizations 1 and 2 compared to indexes of the original architecture 1 therefore a trend to optimality .

Concerning the coupling metrics: Index of Package changing Impact (**IPCI**) and Index of Package Communications Diversion (**IIPUD** and **IIPED**) presented in figure 6, the results obtained for remodularizations 1 and 2 approximate from those of the original architecture 1 extend to a higher interesting value 0.6.

The results obtained at the level of the cohesion for the remodularization 1 and 2 provides an optimum value (with an advantage to the remodularization 1 remaining more performance for choosing a software expert).

The results of the coupling have an improvement at the level of remodularized architectures 1 and 2 compared to the original architecture 1.

6. 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[12] [13]. 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. Approaches to restructuring based on meta-heuristic algorithms 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 provide an algebraic derivation of hierarchies of abstractions from all entities of a system. In our approach, we add the dimension of exploration using the FCA[12] [13].

A large part of previous works related to oriented software metrics has focused on the issue of characterizing the class design, either looking internal complexity or relationship between a given class and other classes[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11].

In the literature, there is also a body of work that focus on object oriented metrics from the standpoint of their correlation with software changeability [1][14], or from the standpoint of their ability to predicate softwair maintenability [1][15]. Other reasearchers argue that the measures resulted by the cohesion and coupling metrics of the previous works are open to interpretation [1] [15].

In general, there are few metrics in the the literature devoted to packages.
Our cohesion and coupling metrics we provide in this work are similar to the metrics provided by Ducasse [1]. The results have an improvement at the level of remodularized architectures.

7. Conclusion

In this article, part of preparation a Phd thesis in software engineering, has been the subject of a communication in the conference [12] [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. This method guided and evaluated by metrics on two architectures results of a remodularization by exploring the redistribution classes of a package with an approach based on formal Concept Analysis, we proceed in this paper the calculation of cohesion and coupling metrics which have revealed to us indices tending to an improvement corresponding parameters providing an alternative choice for a software expert.

References

[1] S. Dcasse, N. Anquetil, M.U. Bhatti and A.C. Hora. Software metrics for package remodularisation. Research report, November 2011.
[2] S. R. Chidamber and C. F .Kemer. A metrics suit for object oriented design. IEEETSE, 20: 476-493, 1994.
[3] F.B. Abreu and R. Carapuca. Candidate metrics for objected-oriented software within a taxonomy framework. Journal of Sys, Sof. 26: 87-96, 1994.
[4] W. Li and S. Henry. Objected-oriented metrics that predict maintainability. Journal of Sys, Sof. 23:111-112, 1993.
[5] W. Li. Another metric suit for object oriented programming. Journal of Sys, Sof. 44:155-162, 1998.
[6] B.H. Selers. Object-Oriented Metrics: Measures of Complexity. Prentice-Hall, 1996.
[7] J.M. Bieman and B.K. Kang. Cohesion and reuse in an object-oriented system. In ACM Symposium on Software Reusability. April 1995.
[8] J.M. Bieman and B.K. Kang. Measuring design-level cohesion. IEEETSE, 24(2) :111-124, February 1998.
[9] L.C. Briand, S. Morasca and V. R. Basili. Defining and validation measures for object-based high-level design. IEEE TSE, pages : 722-743, 1999.
[10] L.C. Briand, J.W Daly and J. Wust. A Unified Framework for Cohesion Measurement in Objected-Oriented Systems. Empirical Software Engineering. An International Journal, 3(1):65-117, 1998.
[11] L.C. Briand, J.W Daly and J. Wust. A Unified Framework for Coupling Measurement in Objected-Oriented Systems. IEEETSE, 25(1):91-121, 1999.
[12] L. M. Hakik, M. Huchard, R. El Harti et A.D. 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.
[13] L.M. Hakik, R. El Harti. "*Exploring the Redistribution Classes of a Package with an Approach Based on Formal Concept Analysis* ", Vol.2 - Issue 12 (December - 2013), International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, www.ijert.org.
[14] H. Kabaili, R.K. Keller, F. Lustman. Cohesion as changeability *indicato*r in object- oriented systems. In Fifth Europ. Conf, on Sof. Maintenance and Reengineering. CSMR 01, pages39-46, Washington, DC, USA, 2001. IEEE Computer Society.
[15] R.K. Bandi, V.K. Vaishnavi and D.E. Tuk. Predicting maintenance performance using object- oriented design complexity *met*rics. IEEETSE, 29: 77-87, 2003.
[16] B. Ganter and R. Wille. Formal Concept Analysis. Mathematical Fondations. Spinge. 1999.
[17] 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.

Lala Madiha Hakik received the Maitrise in Computer Engineering, from Hassan 1st University, FST, Settat, Morocco in 2005, Specialized Master in Software Engineering, Montpellier -2- University, France in 2009, She is a PHD Stdent in Computer Science specialized in Software Engineering, University Hassan 1st , FST, Settat, Morocco, 2014.

Rachid El Harti received the PHD in Mathematics and applications from Mohammed V University, Morocco in 1993, Full professor, Hassan 1st University, Morocco