Comparison of Results Obtained by Application of Techniques Based on Formal Concept Analysis and Oriented Graph for a Remodulaisation Software Architecture Composed of Classes and Packages

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
In a previous study we proceeded to the remodularization architecture based classes and packages using the Formal Concept Analysis (FCA) [2] [13] [14], we then got two possible remodularized architectures. We tried the redistribution of classes by using a technique based on Oriented Graph to determine the packages that receive the redistributed classes, we then got one possible remodularized architecture. After, we have evaluate the quality of a remodularized Software Architecture by metrics for measuring Coupling and Cohesion of a Package for the two techniques adopted. This paper presents the comparison of results obtained by application of techniques based on formal Concept analysis and Oriented graph for a remodulaisation software architecture composed of classes and packages.
Keyword:
Remodularization, Software architecture, Classes and packages, FCA, Oriented graph, Classes and packages.

1. Introduction

Great software systems based on approaches, the object consist of classes grouped into packages, forming a modular structure[1].
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[1].
Many researchers make proposals on this subject using technical visualization, algorithms of remodularization or Exploring the Redistribution Classes of a Package with an Approach Based on Formal Concept Analysis. [13] [14] or solution using Oriented Graph based on the technique of shortest path[1].
In this paper, we provide *a comparison of results obtained by application of techniques based on formal Concept analysis* [13] [14] *and Oriented graph for a remodulaisation software architecture composed of classes and packages* [1] and we illustrate our proposal with a theoretical example.
Section 2 presents our example, then we describe the approaches in Section 3. Section 4 presents the comparison of results obtained by application of techniques based on formal Concept analysis and Oriented graph for a remodulaisation software architecture composed of classes and packages. Related work is presented in Section 5, and then we conclude in Section 6.

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[2] [13] [14].
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) and the Oriented Graph can bring interesting ways to solve this problem because this two technicals methods allows the

Manuscript received February 5, 2014
Manuscript revised February 20, 2014

group to connect classes identically. We are not looking here to propose a betters solutions; but offer to an expert different hierarchical solutions.

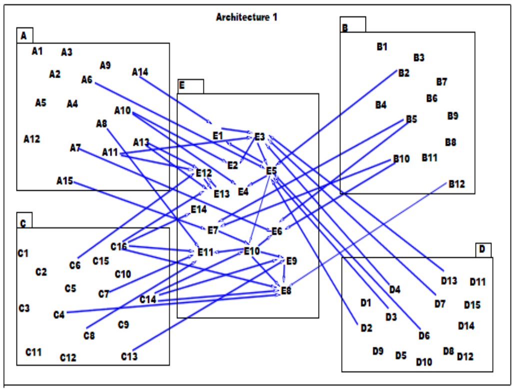


Figure 1. An initial architecture composed of classes and packages.

3. Proposed approaches

3.1. Technique based on Formal Concept Analysis

The Formal Concept Analysis (FCA) [2] [13][14] [12] 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 [2] [13][14] .

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.

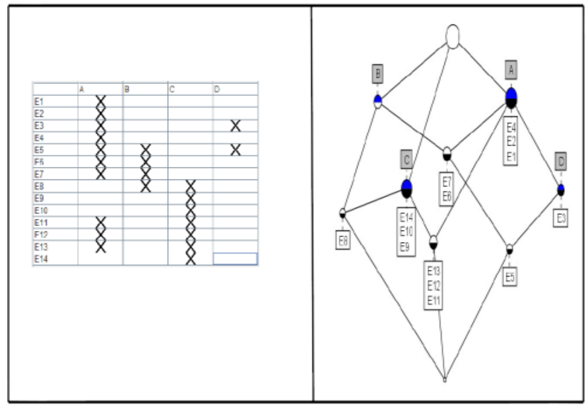


Figure 2. Formal context C2 and lattice T(C2) –Architecture 1-[2] [13][14].

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.

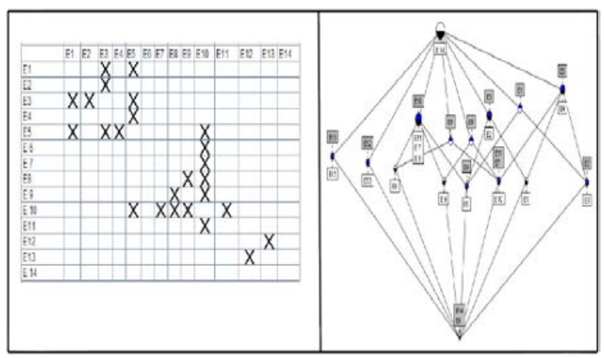


Figure 3. Formal context C5 and lattice T(C5) -Architecture 1-[2] [13][14]

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.

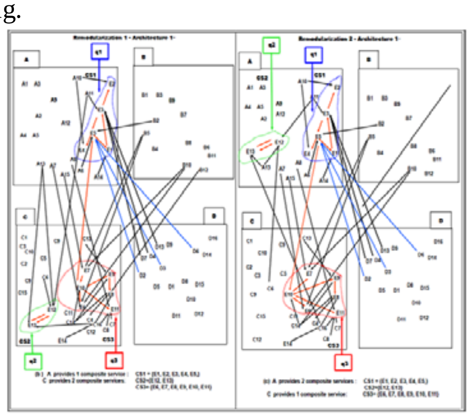


Figure 4. two possibilities of remodularization[2] [13][14].

3.2. Technique based on Oriented Graph

In our approach, we are inspired of the notion of graph to present the original architecture of Figure 1 as nodes relative to classes and arcs relative to the relationship between these classes. Figure5 illustrates this vision[1]

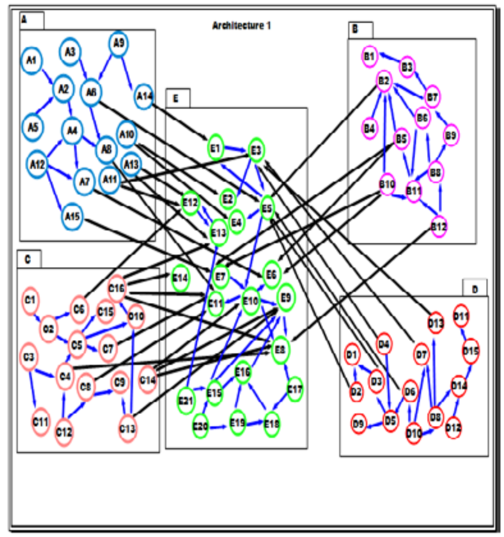


Figure 5. Original architecture of Figure 1 as nodes relative to classes and arcs relative to the relationship between these classes[1].

A. *Formalization*

In a second step we focus on the classes of package E, to be deleted related with classes of other packages considered in this case as nodes shown in figure 6 [1].

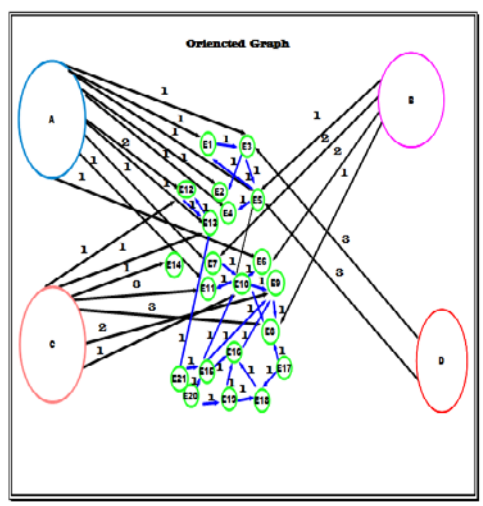


Figure 6. Oriented Graph result of figure 2 [1].

- The relationship between classes and packages are represented by edges connecting each pair of nodes as an example the nodes A and E1 are connecting by the edge (A,E1) image of couple (**Package**, **Class**) [1].

It is found that all the conditions are met to define a graph oriented, object of Figure 6 [1].

Definition 1 (Oriented Graph) [15]:
A graph G is a mathematical structure defined by a pair (N, E) where N is a set of objects called nodes or vertices and E part of N * N which represents a set of arcs (also called edges) each connecting a pair of nodes.
This general definition is a directed graph distinguishes two vertices s1 and s2 the edge (s1, s2) of the edge (s2, s1) [1].
The number of connections available to each class of package E with classes of packages A, B, C and D and mentioned on the arcs [1].
Example of procedure:
For the choice of the allocation of classes E Package to one of the other packages A, B, C and D, we adopted an approach advocating the use of directed graph and the technique based on the definition of the shortest path[1].
For examples:

1- In Figure 5, the class E1 of package E has an external connection with class A14 of package A, therefore the corresponding arc in figure 6 between class E1 and package A is the number 1. where the idea of cost of a shortest path [1]. By applying this principle class E1 of package E is affected into package A

2- the class E12 of package E has three external relations, both with package A (one with class A13 of package A, the other with the class A11 in the same package) and the third class C16 of package C (Figure 6). Under the definition of the shortest path the class E12 go to Package C [1].

Special case:

3- the class E8 of package E has three internal relations with the classes E9, E10 and E17 of package E and two external relations, one with class B12 of package B and the other with class C4 of package C, since the classes E9 and E10 will be affected by the principle of shortest path, to the package C therefore E8 will go also to the package C dominant [1].

Thus all the classes in the package E are redistributed according to the methodology listed above, and thereby the package E has been deleted to arrive on remodularized architecture (figure 7) [1].

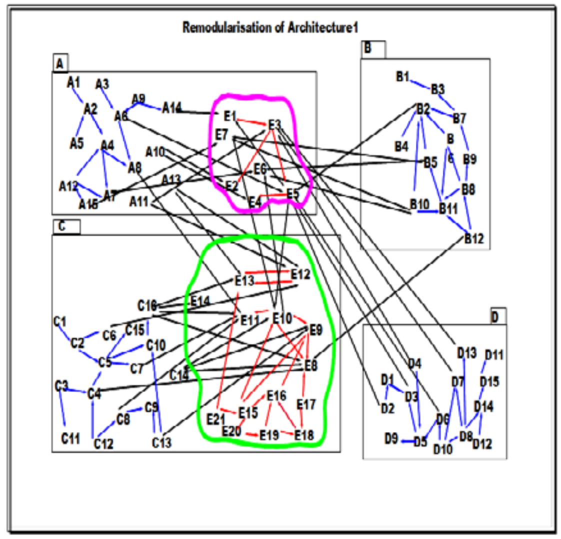


Figure 7 . one possibility of remodularization [1].

4. The comparison of results obtained by application of techniques based on formal Concept analysis and Oriented graph

4.1. Comparison of remodularized software architectures obtained

Concerning the technical of redistribution of classes based on formal concept analysis, we got two remodularized software architecture offering an alternative choice to a software expert on one hand and know the way back to the original architecture on the other hand [2] [13] [14]
As to the result of the redistribution of classes in a package to other package by using the graph-oriented, this technique has generated one and unique remodularized software architecture [1].

4.2. Comparison the results of the validation metrics coupling and cohesion

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 (using formal concept analysis techniques which resulted into two possible architectures (figure 4). The package E is removed during this operation. Initial architecture (figure 1) and the two architectures (figure 4) result from the remodularization obtained by applying our approach based on formal concept analysis, which has been the object of the articles [2] [13][14].
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 [2].
Furthermore the results obtained of the redistribution of classes in a package to other package by using the graph-oriented [1], at the level of the cohesion for the remodularized architecture 1 provides an optimum value 1 compared to the original architecture 1. The results of the coupling have an improvement at the level of remodularized architecture 1 compared to the original architecture 1.
So the two techniques adopted for the redistribution of classes we have revealed interesting results tending to optimization and limiting the number of remodularized software architectures proposed to the software expert.

5. 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[6]. 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-ori
3ented software reengineering. In previous work, we added the dimension of exploration using the FCA[13][14] and we explored the issue of redistributing classes of a package to other packages using an approach based on Oriented Graph to determine the packages that receive the redistributed classes and we have evaluate the quality of a remodularized Software Architecture by metrics for measuring Coupling and Cohesion of a Package [18].
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[16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26].
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 [16][27], or from the standpoint of their ability to predicate software maintainability [16][28]. Other researchers argue that the measures resulted by the cohesion and coupling metrics of the previous works are open to interpretation [16] [28].
In general, there are few metrics in the the literature devoted to packages.
Our cohesion and coupling metrics we provided are similar to the metrics provided by Ducasse [16] for validation[1] [2].
In this paper we proceed to the comparison of results obtained by application of techniques based on formal Concept analysis [2] and Oriented graph[1].
We think that the usefulness of this comparison is enable to the software expert to make a choice for one of the techniques to be adopted.

6. Conclusion

This article summarizes the methods used for the redistribution of classes in a package of software architecture consisting of 5 packages for its remodularization by using formal concept analysis as a first step and the oriented graph in a second step. These methods have been evaluated by metrics, the calculation of cohesion and coupling metrics which have revealed to us indices tending to an improvement corresponding parameters [1] [2].
We proceed in this paper to the comparison of results obtained by application of techniques based on formal Concept analysis and Oriented graph providing an alternative choice for a software expert.

References

[1] L.M. Hakik, R. El harti. Technique Of Redistribution Classes Of A Package With An Approach Based On Oriented Graph And Evaluation Quality of A Remodularized Software Architecture. International Journal of Innovative Research in Science, Engineering and Technology.ISSN:2319-8753. Vol. 3, Issue 1, January 2014.

[2] L.M. Hakik, R. El Harti. Measuring Coupling and Cohesion to Evaluate the Quality of a Remodularized Software Architecture Result of an Approach Based on Formal Concept Analysis. IJCSNS International Journal of Computer Science and Network Security .Vol. 14 No. 1 pp. 11-16. Journal ISSN : 1738-7906. January 2014..

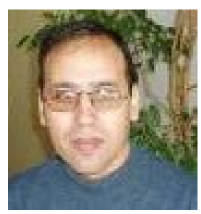
[3] F.B. Abreu, G. Pereira, and P. Sousa. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In Proceeding of the confeence on Software Maintenance and Reengineering. CSMR 'OO, pages 13-, Washington, DC, USA, 2000. IEEE Compter Society Press.

[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.
[14] Lala Madiha Hakik, Rachid 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.
[15] A. Anwar. Formalisation par une approche IDM de la composition de modeles dans le profil VUML. Thesis. Toulouse University. 2009.
[16] S. Dcasse, N. Anquetil, M.U. Bhatti and A.C. Hora. Software metrics for package remodularisation. Research report, November 2011.
[17] S. R. Chidamber and C. F .Kemer. A metrics suit for object oriented design. IEEETSE, 20: 476-493, 1994.
[18] F.B. Abreu and R. Carapuca. Candidate metrics for objected-oriented software within a taxonomy framework. Journal of Sys, Sof. 26: 87-96, 1994.
[19] W. Li and S. Henry. Objected-oriented metrics that predict maintainability. Journal of Sys, Sof. 23:111-112, 1993.
[20] W. Li. Another metric suit for object oriented programming. Journal of Sys, Sof. 44:155-162, 1998.
[21] B.H. Selers. Object-Oriented Metrics: Measures of Complexity. Prentice-Hall, 1996.
[22] J.M. Bieman and B.K. Kang. Cohesion and reuse in an object-oriented system. In ACM Symposium on Software Reusability. April 1995.
[23] J.M. Bieman and B.K. Kang. Measuring design-level cohesion. IEEETSE, 24(2) :111-124, February 1998.
[24] 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.
[25] 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.
[26] 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.
[27] R.K. Bandi, V.K. Vaishnavi and D.E. Tuk. Predicting maintenance performance using object- oriented design complexity metrics. IEEETSE, 29: 77-87, 2003.
[28] H. Kabaili, R.K. Keller, F. Lustman. Cohesion as changeability indicator in object- oriented systems. In Fifth Europ. Conf, on Sof. Maintenance and Reengineering. CSMR 01, pages39-46, Washington, DC, USA, 2001. IEEE Computer Society.
[29] R.K. Bandi, V.K. Vaishnavi and D.E. Tuk. Predicting maintenance performance using object- oriented design complexity metrics. IEEETSE, 29: 77-87, 2003.

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 Student 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 iversity, Morocco