I.J. Education and Management Engineering, **2015, 3, 1-8**
Published Online July 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2015.03.01

Available online at http://www.mecs-press.net/ijeme

Change Impact Identification in Object-Oriented System: Dependence Graph Approach

*Abdi Mustapha Kamel, Dinedane Mohammed Zoheir

Department of Computer Science, University of Oran 1 Ahmed Ben Bella BP 1524 El Mnaouer, Oran, Algérie

Abstract

The development of software products consumes a lot of time and resources. On the other hand, these development costs are lower than maintenance costs, which represent a major concern, specially, for systems designed with recent technologies. Systems modification should be taken rigorously, and change effects must be considered. The need to offer tools allowing on the one hand, to explain the mechanisms of changes at the source code level, and on the other hand, to reduce the effort as well as the cost of maintenance, is affirmed more and more.

We propose in this article an approach based on dependence graph to identify change impact of object-oriented systems. The analysis of change ripple effect is made on the dependence graph representing the considered system. The identification at graph level of maximum range of change ripple effect will enable us to know consequences change at code source of system. This knowledge will enable us to estimate the change cost and to make a compromise between the various suggested changes. Finally, to concretize this study we produced a tool and tested our approach on certain systems.

Index Terms: Object-Oriented Systems, Change Impact, Ripple Effect, Analysis, Prediction, Dependence.

1. Introduction

Maintenance is the last phase of the software life cycle. It is defined as the modification process of software in operation to allow it to always satisfy current and future specifications [8]. According to Pfleeger [17], maintenance cost depends in large part (40 %) on the modification of software architecture, interactions between components, procedures /methods, and variables. Systems modification should be taken seriously; changes effects must be considered [19]. A small change can have considerable and unexpected effects on the system. Risks incurred during the modification are related to the consequence of the impact of a given change. When modularity is adequately used, it limits the effects relating to changes. Nevertheless, change impacts are

* Corresponding author
Email: abdimk@yahoo.fr, din_danos@hotmail.fr

subtle and difficult to discover; designers and maintainers need mechanisms to analyze changes and to know how they are propagated in the whole system.

The main motivation of our work is to improve the maintenance of object-oriented systems, and to intervene more specifically on change impact analysis. By identifying the potential impact of a modification, one reduces the risk to deal with expensive and unpredictable changes. For that, we try to give more explanations on real and responsible factors for this change impact and its evolution.

Among several representation models, the dependence graphs constitute an interesting approach for identification study of change impact of object-oriented system at least on class level. Section 2 presents various works done in the topic of change impact analysis. Our proposition is presented in the third section. We start by defining the change impact concept in our context. Then we explain our problematic and we give the solution inspired of graph theory. In section 4, we present our tool produced within the framework of this work and we show through an example how to locate the change impact in an object-oriented system. Finally, section 5 concludes this study and presents the principal perspectives of our work.

2. Related Works

Several studies were done concerning change impact. Thus, Han [10] developed an approach for computing change impact on design and implementation documents. This approach considers the original representation of software artifacts (classes) rather than an extracted system model separately. The dependencies artifacts involve inheritance, aggregation and association. The approach is both automatic, based on rules of change propagation, and manual, requiring the intervention of the user. However, no change model is formally defined. On another side, Lindvall [16] identified the most common and frequent changes in C++ so that change models can be built to help developers make predictions regarding future requirements.

In [4], the authors predicted evolving object-oriented systems size starting from the analysis of the classes impacted by a change request. They predicted changes size in terms of added/modified lines of code. Kung et al [13] interested by regression testing, developed a change impact model based on three links: inheritance, association, and, aggregation. They also defined formal algorithms to calculate all the impacted classes including ripple effects. Li and Offutt [14] and [15] examined the effects of encapsulation, inheritance, and, polymorphism on change impact; they also proposed algorithms for calculating the complete impact of changes made in a given class. However, some changes, implying for instance inheritance and aggregation, were not completely covered by their algorithms.

Briand et al in [5] tried to see if coupling measures, capturing all kinds of collaboration between classes, can help to analyze change impact. Strategy adopted in this study is different from other strategies since it is purely empirical. This study, (i) showed that some coupling metrics, related to aggregation and invocation, are connected to ripple effect, and, (ii), it allows performing dependence analysis and reducing impact analysis effort. In [6], [11] and [12], a change impact model was defined at an abstract level, to study the changeability of object-oriented systems. The adopted approach uses characteristic properties of object-oriented systems design (coupling, cohesion,, etc.), measured by metrics, to predict changeability.

According to a different perspective, Sahraoui and al. studied in [18] the impact of refactoring on structure and thus on structural metrics. This study made it possible to determine the refactoring scenarios that can improve or deteriorate certain structural properties. In [1], [2], [3], [20] and [21], the authors also showed that coupling, measured by some metrics, influences change impact. Recently, the authors in [22], [23] and [24] try to apply Bayesian networks to study and predict the behavior of software systems (usually coded in Java) to determine the key factors that influence the more the change impact in such systems. In this work, we locate the change impact in object-oriented using dependence graph approach. In the following section, we present our proposition starting by defining change impact.

3. Proposition

3.1 Impact Concept

A class "A" has an impact on a class "B" if one change in "A" can involve that "B" does not compile any more or that the behavior of B is affected. To this impact concept, we associate the dependence concept between two classes. When a class A can have an impact on a class B, we say that B depends on A. There is a direct relationship (or direct dependence) between two classes A and B, if these classes are linked by at least one of the three following relations: inheritance, aggregation, use.

The three relations above (noted R) have from an impact point of view the following properties: reflexivity, non-symmetrical and transitivity. We insist on the fact that this is valid only from the impact point of view. Aggregation and the use, contrary to inheritance, are not transitive relations. Indeed, if the class A is an aggregation of B and B an aggregation of C; then A is not inevitably an aggregation of C. Because it may be that no attribute of A is an instance of C. But on the other hand, C can have an impact on A via this aggregation relation. Therefore, it is the impact which propagates these relations which is transitive, but not these relations.

3.2 Indirect Relations Between Classes

A class A has an indirect relationship with a class B, if there exist $B_1, B_2, \ldots, B_n$ such as:

$$A\ R\ B_1,\ B_1\ R\ B_2,\ B_2\ R\ B_3,\ \ldots,\ B_n\ R\ B \text{ and } n > 0 \quad (1)$$

This is noted: $A\ R^+\ B$.

The R^+ relation is called the closing of *R*, it has as a property to be always transitive, from where its usual name of transitive closing. This relation includes all the classes which are connected directly or indirectly by relation *R*. If the relation *R* is defined as follows:

$$R = \{(A, B) \text{ such as: A can have an impact on B}\} \quad (2)$$

$$\text{et } R^+ = \{(A, B) \text{ such as: } \exists\ B_1,\ldots,\ B_n \text{ et } A\ R\ B_1,\ B_1\ R\ B_2,\ \ldots,\ B_n\ R\ B\} \quad (3)$$

Then for a class C, the following set: {X such as: C R^+ X } contains all classes connected directly or indirectly to C and on which C can thus have an impact. We call this set the ***C firewall***, then we give its definition in following sub-section.

3.3 Firewall of Class

We define the firewall of a class C like the set of classes which can be affected by a modification in C. In other words, they are all classes on which C can have an impact, and we note it: *CFW(C)*. It is important to keep in mind that the firewall includes the classes, possibly but not necessarily, affected by a change. For a given modification, a class of the firewall can not be affected. But on the other hand any affected class is necessarily in this set.

To have a certain confidence degree in a modified program, we must re-test all classes of firewall, because it constitutes the maximum range of the consequences of a modification. No edge effect can escape the firewall. An equivalent definition of the firewall, consists in saying that they are all classes which must be re-tested after a class was modified. For the modification of several classes, the firewall is the reunion of the firewalls of each class taken individually.

Suggested solution

A traditional problem when we work with a directed graph is to find all nodes which we can reach starting from a given node. In other words, for a node A, find all nodes X of graph for which there exists a way from A to X. This problem amounts calculating the transitive closing of relation which corresponds to graph.

The calcul of transitive closing is traditionally solved by the famous Warshall algorithm [7]. It provides a matrix (connectivity matrix) indicating for each node all the others which are connected to it. In the context of impact analysis for an object code, this algorithm will enable us to calculate all the classes which can be affected by a modification in a given class.

3.4 Warshall Algorithm

Procedure Warshall (A: BoolMatrix; var P: BoolMatrix; n: integer,);
/* A is the adjacent matrix of graph, P the connectivity matrix which must be calculated and n the nodes number of graph */

```
int i, j, k;
begin
for i: =1 to n do
     for j: = 1 to n do
          P[i, j]: = A[i, j];
for k: = 1 to n do
    for i: = 1 to n do
         for j: = 1 to n do
            P[i, j]: = P[i, j] or (P[i, k] and P[k, j])
end;
```

Other algorithms were proposed in the literature to calculate transitive closing in more reasonable times for very large graphs. They are inspired more or less almost all from Warshall algorithm [7].

4. Change Impact Identification

We point out that we are interested in the systems (software) coded in Java. We start by extracting the design diagram (diagram of classes) of system in entry. Then we check its coherence before extracting the corresponding dependence graph. Thereafter, we can select any class of system and we will make the desired change. By the means of Warshall algorithm (section 3.4), we can estimated change impact on the remainder of system. The following figure 1 presents the various stages of our approach.

In this work, we produced a tool allowing to support the stages of our approach (figure 1). Except for the extraction stage of design diagram (diagram of classes), which for the moment, is done manually, all the other stages are achieved by the tool. Figure 2 shows the result of the coherence verification stage of the classes diagram in entry which corresponds to a system coded in java and composed of 8 classes. The classes are numbered from class 1 to class 8.

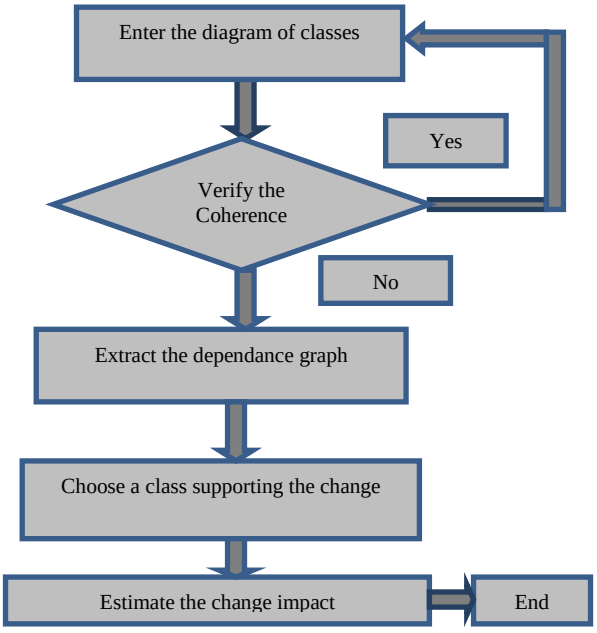

Fig.1. Stages of our Approach

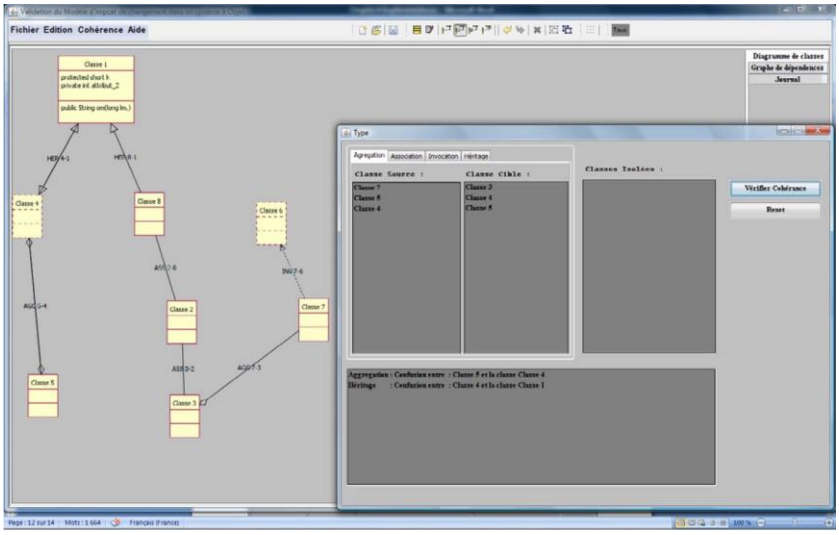

Fig.2. Example of coherence verification

We select for example class 7 like class supporting the change (change of variable, method or class). The Warshall algorithm execution on the graph enables us to locate the ripple effect of this change on all the remainder of the system. In our case, the set of classes which are supposed being impacted following this change is the set of all classes colored in green (figure 3). There are finally 3 classes. The nonaffected classes by ripple effect are colored in yellow and the class supporting the change in blue.

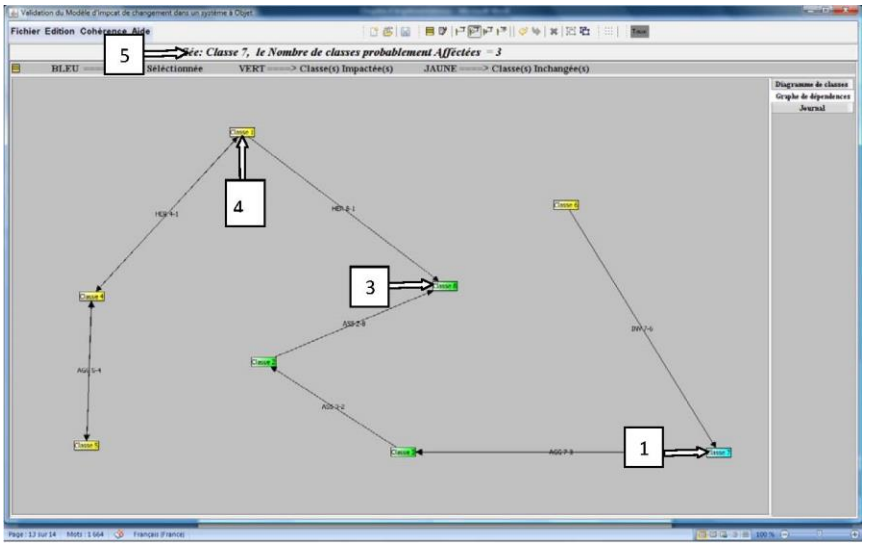

Fig.3. Localization example of change impact

5. Conclusion

We proposed in this article an approach based on dependence graph to identify change impact in object oriented system. A thorough study and a general synthesis of various former works dealing with this subject were initially essential. We explained our problematic and we proposed a solution in graph theory. Then, we produced a tool to support our approach. Finally, we showed the results of this tool on an example of small object-oriented system with 8 classes.

Like perspectives for this work, we plan mainly to extract automatically the design diagram (diagram of classes) from system in entry (coded in java) as well as to validate our approach on systems of average (or large) size.

Acknowledgements

This work was supported in part by the Industrial Computing and Networking Laboratory of Oran University.

References

[1] M.K Abdi, H. Lounis, H. Sahraoui: "Using Coupling Metrics for Change Impact Analysis in Object-Oriented Systems" In QAOOSE 2006 Proceedings, 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 3 July 2006 - Nantes, France.

[2] M.K Abdi, H. Lounis, H. Sahraoui: "Analyzing Change Impact in Object-Oriented Systems " In proceedings of the 32nd EUROMICRO Software Engineering and Advanced Applications Conference, Cavtat/Dubrovnik (Croatia), August 29-September 1, 2006.

[3] M.K Abdi, H. Lounis, H. Sahraoui, M.K Rahmouni: "Vers une approche d'analyse de l'impact du changement dans un système à objets", dans revue "L'Objet", volume 13 – N ° 1/2007, Pages 147-169, Éditions Hermès.

[4] Antoniol G., CANFORA G., LUCIA A. D., "Estimating the size of changes for evolving Object-Oriented Systems: a Case Study" *in Proceedings of the 6th International Software Metrics Symposium*, pages 250-258, Boca Raton, Florida, Nov 1999.

[5] Briand L. C., WÜST J., LOUNIS H., "Using Coupling Measurement for Impact Analysis in Object-Oriented Systems" *in proceedings of the International Conference on Software Maintenance ICSM'99*, Oxford, England, August 30 – September 3, 1999.

[6] M. A. Chaumun, H. Kabaili, R. K. Keller and F. Lustman. *"A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems"*. In Proceedings of the Third Euromicro Working Conference on Software Maintenance and Reengineering CSMR'99, pages 130-138, Amsterdam, The Netherlands, March 1999.

[7] Aho A. V., J. E. Hopcroft and J. D. Ullman, "Data Structures and Algorithms", Addison-Wesley Publ. Comp., 1983.

[8] Computer Society Press, Standards Collection –Software Engineering, The Institute of Electrical and Electronics Engineers, Inc., 1993.

[9] Alikacem EL Hachemi, Hicham Snoussi, *"BOAP 1.1.0: Manuel d'utilisation",* CRIM, Janvier 2002.

[10] HAN J., "Supporting Impact Analysis and Change Propagation in Software Engineering Environments" *in Proceedings of the STEP'97*, London, England, pages 172-182, July 1997.

[11] Hind Kabaili, Rudolf K. Keller, François Lustman, and Guy Saint-Denis. Class Cohesion Revisited: An Empirical Study on Industrial Systems. In *Proceedings of the Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 29-38, Cannes, France, June 2000.

[12] H. Kabaili, *"Changeabilité des logiciels orientés objet: propriétés architecturales et indicateurs de qualité"*, PhD thesis, Université de Montréal, Canada, Janvier, 2002.

[13] Kung D. C., GAO J., HSIA P., LIN J., TOYOSHIMA Y., "Class firewall, test order, and regression testing of object-oriented programs" *in Journal of Object-Oriented Programming*, Vol. 8, No. 2, pages 51-65, May 1995.

[14] LEE M., OFFUTT A. J., "Algorithmic Analysis of the Impact of Changes to Object-Oriented Software" *in Proceedings of the ICSM'96*, pages 171-184, 1996.

[15] Lee M., Change Impact Analysis for Object-Oriented Software, PhD thesis, George Mason University, Virginia, USA, 1998.

[16] Lindvall M., "Measurment of change: Stable and Change-Prone Constructs in a commercial C++ System" *in Proceedings of the 6th International Software Metrics Symposium*, pages 40-49, Boca Raton, Florida, Nov 1999.

[17] Pfleeger S. L., "A Framework for Software Maintenance Metrics" *in IEEE Transactions on Software Engineering*, pages 320-327, May 1990.

[18] SAHRAOUI H. A., GODIN R., MICELI T., "Can metrics help to bridge the gap between the improvement of OO design quality and its automation?", *in Proceedings of the International Conference on Software Maintenance (ICSM'00)*, 2000.

[19] WILDE N., HUITT R., "Maintenance support for object-oriented programs" i*n IEEE Transactions on Software Engineering*, Vol. 18, Issue 12, Pages 1038–1044, Dec 1992.

[20] Abdi, M. K (2007) "Analyse et Prédiction d'impact de Changement dans un système à objets", Thèse de Doctorat d'Etat en Informatique, Université d'Oran, Es-Sénia, Avril 2007.

[21] Cheikhi, L (2004) *"Estimation de l'impact de changement dans les programmes à objet"*. Thèse de Master, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal.

[22] C. Cherif, M.K. ABDI, «Change Impact Study by Bayesian networks», Modeling Approaches and Algorithms for Advanced Computer Applications Studies in Computational Intelligence, Volume 488, Springer, pp 429-438, may 2013, ISSN: 1860-949X, ISBN: 978-3-319-00559-1.

[23] C. Cherif, M.K. ABDI, "Étude de l'impact de changement dans les systèmes à objet par les réseaux bayésiens" dans la Revue COST, Communication Science et Technologie, N 15, Janvier 2015.

[24] M.A. Zehouane, M.K. ABDI, "Simulation de comportement de Systèmes logiciels par les réseaux bayésiens", Mémoire de Master en Informatique, Juin 2015, Université d'Oran 1 Ahmed BenBella.
[25] Chérif Chahira, "Étude de l'impact de changement dans les systèmes à objets par les réseaux bayésiens" Mémoire de Magister, Février 2013, Université d'Oran 1 Ahmed BenBella.
[26] C. Cherif, M.K. ABDI, "Étude de l'évolution des systèmes à objet par les réseaux bayésiens" dans les actes des 3éme Journées des Étudiants à l'École Supérieure d'Informatique, ESI, Oued Smar, Alger, JEESI'14, 19 mai 2014.

Authors'Profiles

M.K. ABDI holds a master degree and a Ph.D. degree in computer science from Department of Computer Science at the University of Oran 1 Ahmed Ben Bella, Algeria. He is currently, professor for the same Department. His research interests include the application of artificial intelligence techniques to software engineering, software quality, formal specification, Systems analysis and simulations, Data-Mining and Information Research.

M.Z. Dinedane is Phd Student at the Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria. He received his Engineering degree in Computer Science and his Master from the Computer Science Department in 2008 and 2011. His research interests include software engineering, software Decision Support Systems and Multi-Agents Systems.

How to cite this paper: Abdi Mustapha Kamel, Dinedane Mohammed Zoheir,"Change Impact Identification in Object-Oriented System: Dependence Graph Approach", IJEME, vol.5, no.3, pp.1-8, 2015.DOI: 10.5815/ijeme.2015.03.01