



Vers une vérification de l'étendu de l'impact de changement dans les systèmes à objets

DAHANE Miloud⁽¹⁾, ABDI Mustapha Kamel⁽²⁾

(1), (2) Laboratoire RIIR, Université d'Oran1 Ahmed BEN BELLA,
BP 1524 ELM_Naouer 31000 Oran, Algérie.

(1) dahane_miloud@yahoo.fr, (2) abdimk@yahoo.fr

Résumé. Un petit changement peut avoir des effets considérables et inattendus sur le reste du système. Le danger encouru lors de la modification, réside dans cette conséquence de l'impact d'un changement donné. Dans ce papier, on propose une démarche de vérification de l'étendu de l'impact de changement en utilisant des techniques qui se basent sur des modèles d'impact de changement afin de confronter les résultats des modèles en question. Nous abordons une démarche de vérification empirique qui permet de renforcer les affirmations des modèles, nous utilisons l'algorithme de Warshall qui donne la portée maximale d'un changement pour soutenir les résultats des tests empiriques. Cette démarche nous a permis de montrer que les résultats obtenus par les tests empiriques sont inférieures aux résultats obtenus par les modèles d'impact de changement et que ces derniers sont à leurs tours inférieurs aux résultats déduits par l'algorithme de Warshall.

Mots clés : Analyse d'impact de changement, orientés objets, algorithme de Warshall, modèle d'impact.

I. Introduction

La maintenance de logiciel est une partie intégrante du cycle de vie du logiciel. Cependant, elle n'a pas, historiquement, reçu le même degré d'attention qu'ont eu les autres phases. Historiquement, le développement a eu bien plus de considération que la maintenance dans la plupart des organisations. Ceci est maintenant en train de se changer vu que les organisations essaient d'obtenir le plus possible de leur investissement en développement en conservant les logiciels en opération le plus longtemps possible.



D'autre part le paradigme du logiciel libre a apporté encore plus d'attention sur la maintenance du logiciel développé par d'autres.

Les deux activités les plus coûteuses dans la maintenance de logiciels sont la compréhension du problème qui est généralement liée à la compréhension du logiciel à maintenir, et la maîtrise de la totalité des effets de propagation des changements proposés [1]. En effet, un petit changement peut avoir des effets considérables et inattendus sur le reste du système. Le danger encouru lors de la modification, réside dans cette conséquence de l'impact d'un changement opéré. La modularité en oriente objet, adéquatement utilisée, limite les effets relatifs aux changements. Néanmoins, ils sont subtils et difficiles à découvrir. Pour toutes ces raisons, les concepteurs et les mainteneurs ressentent le besoin de mécanismes pour analyser les changements et connaître comment ils sont propagés dans le reste du système. Ce processus s'appelle l'analyse de l'impact du changement.

Dans la suite du papier, nous définissons en section 2 l'impact de changement. La section 3 décrit les approches et modèles d'impact de changement tandis que dans la section 4 nous présentons l'algorithme de Warshall. La section 5 est consacrée à la présentation du modèle d'impact de changement. L'expérimentation ainsi que les tests réalisés sont détaillés dans la section 6. La section 7 conclut notre travail et esquisse les perspectives.

II. Analyse d'impact

L'identification des impacts possibles avant d'effectuer le changement, nous permettra de réduire les risques d'entreprendre des changements coûteux. En effet, plus les problèmes liés à un changement sont découverts tard, plus leur coût augmente [2].

Généralement, plusieurs changements sont proposés pour résoudre un même problème et satisfaire le même besoin. L'évaluation de l'impact de chaque changement permet de choisir le changement adéquat. Selon M. Lee [3], l'expérience a montré qu'effectuer des changements dans un code sans comprendre leurs implications peut conduire à une mauvaise estimation de l'effort à

produire, à prolonger les délais de livraison, à dégrader la conception d'un logiciel, à entamer la fiabilité du produit et enfin peut amener au retrait prématuré du logiciel.

III. Approches et modèles d'analyse d'impact de changement

La synthèse des travaux autour de la problématique d'analyse et prédiction d'impact de changement présentée dans [4] a permis de dégager les principales approches adoptées ainsi que les différents modèles (de base) utilisés dans ces travaux. La Table 1 résume cette synthèse. De plus, l'auteur distingue les cas où on peut avoir de l'analyse d'impact puis sa mesure (son calcul) des cas où en plus de l'analyse d'impact, on peut avoir aussi sa prédiction avant sa mesure.

Table 1 Synthèse des différentes approches [4]

Approches purement empiriques	Pas de Modèle			Analyse + Mesure
Approches basées sur modèles	Modèle			Analyse + Prédiction + Mesure
	Graphe de Dépendance	Graphe de Dépendance + Formalisme	Autre que Graphe de Dépendance	

IV. Algorithme de Warshall

L'algorithme de Warshall permet de calculer le firewall d'une classe C. c'est-à-dire l'ensemble des classes pouvant être affectées par une modification dans C. Il est important de garder l'esprit que



le firewall comprend les classes possiblement, mais pas nécessairement affectées par un changement [5].

Nous calculons le firewall de chaque classe en suivant les étapes suivantes :

- 1- Extraire le graphe de dépendance.
- 2- Etablir la matrice adjacente de ce graphe.
- 3- Calculer la fermeture transitive à l'aide de l'algorithme de Warshall.

L'algorithme de Warshall [6] est comme suit :

```
(A :BoolMatrix ; var P :BoolMatrix ;n :integer) ;  
/*A est la matrice adjacente du graphe, P la matrice de  
connectivité qui doit être calculée et n le nombre de nœuds  
du graphe*/  
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;  
    End;  
  End;  
End;
```

V. Modèle d'Impact de changement

Un système est vu comme un ensemble de classes connectées par différents liens. Une classe est définie comme un groupe de méthodes qui servent comme interface publique ou pour des opérations internes, et une section de variables qui définissent l'état des instances de la classe [7]

Quand un changement à un système est considéré, il est nécessaire d'identifier les composants du système qui seront impactés suite à ce changement. Cela permet de s'assurer que le système s'exécutera toujours correctement après que le



changement soit mis en œuvre. Notre intérêt est concentré sur comment le système réagit à ce changement et à d'autres changements en général [8].

Nous appelons impact d'un changement l'ensemble de classes qui exigent une correction suite à ce changement. Il dépend de deux facteurs : l'un est le type de changement, l'autre facteur est la nature des liens impliqués.

Nous tenons à signaler que le modèle d'impact que nous avons utilisé a été défini dans [9, 10] puis raffiné en JAVA dans [4]. La Table 2 suivante présente un extrait de ce modèle d'impact.

Table 2. Extrait des Résultats de l'impact des changements (Java)

Id Changement	Description de Changement	Expression d'impact
v.1.1	Changement de valeur de variable	-
v.1.2	Changement de type de variable	S+L
.	.	.
m.2.1	Changement de méthode (Static/Non-static)	
.	.	.
m.2.6.2	Public → Protected	
m.2.6.2.1	Non-abstract method	I ~H
.	.	.
c.3.1	Changement de classe (Abstract/Non-abstract)	
.	.	.
c.3.2	Suppression de classe	
c.3.2.1	Non-abstract class	S+G+H+I
.	.	.

Il est à noter que l'absence d'un opérateur entre 2 liens signifie une intersection, et l'opérateur "+" signifie une union tandis que l'opérateur "~" avant un lien signifie la négation. Enfin le pseudo lien L signifie « Local » et que les variables du modèle d'impact sont les liens entre les classes (Héritage « H », Agrégation « G », Association « S » et Invocation « I »).

Le modèle d'impact raffiné en JAVA comporte en tout 52 changements : 12 changements de variable, 25 changements de

méthode et 15 changements de classe [4, 11]. Nous avons présenté dans cet extrait seulement les changements que nous avons considérés par la suite dans notre étude empirique.

VI. Expérimentation

Afin de vérifier empiriquement ce modèle d'impact raffiné en JAVA (décrit en section 4), nous proposons d'effectuer plusieurs expérimentations sur divers systèmes (codés en JAVA) en procédant de la manière suivante [12] :

- Représentation graphique de la structure du système (pour visualiser les interactions entre classes)
- Extraction des constituants du système (à partir du code source)
- Calcul de la protée maximale « Algorithme de Warshall »
- Tests empiriques : Des changements sont portés concrètement sur le système (code) et après chaque correction, on effectue une compilation du système et on comptabilise le nombre de modifications apportées jusqu'à la disparition de toutes les erreurs de compilation.
- Confrontation des résultats.

1. Systèmes tests

Deux systèmes ont été considérés dans notre étude empirique (voir Table 3) mais nous avons préféré présenter dans le cadre de ce travail seulement les résultats du système le plus important du point de vue taille (nombre de classes), à savoir le système de "gestion de compagnie aérienne" qui comporte un total de 10 classes [13].

Table 3. Systèmes considérés dans l'étude empirique

Nom du système	Taille (nombre de classes)
Gestion de bibliothèque	07
Gestion de compagnie aérienne	10

2. Représentation en graphe

Dans cette représentation, les liens entre classes sont interprétés graphiquement : par exemple, le losange (◊) entre les classes « Vol » et « Vol_Générique » représente le lien d'Agrégation de même pour le triangle (▽) entre les classes « Passage » et « Individu » et les classes « Passage » et « Client » représente un lien d'Héritage tandis que le trait rouge qui lie les classes « Vol » et « Escale » représente un lien d'invocation.

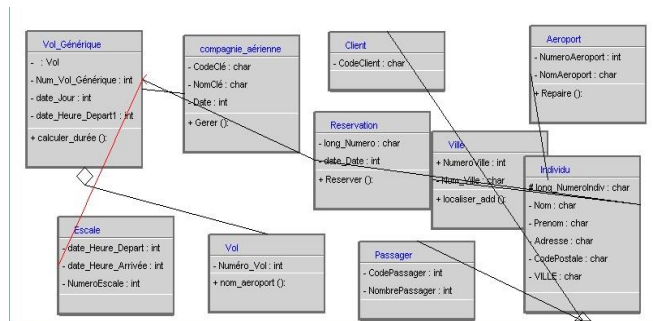


Figure 1. Représentation graphique du système test

3. Composants du système test

Dans la Figure 2, les composants du système (Code) sont ordonnés en (Classe, Attribut, Méthode, Paramètre) et les liens entre les classes, variables du modèle d'impact décrit en section 4 (Héritage « H », Agrégations « G », Association « S » et Invocation « I ») suivant leurs emplacements dans le système et sur la base des concepts de la programmation Orientée Objets, en particulier le langage Java.

Classe

nom_classe	Porté
Aeroport	public
Ville	public
Escal	public
Vol_Général	public
Vol	public
Individu	public
Reservation	public
Passager	static

Attribut

nom	Porté	Type	Var.	Oper	Valeur	Static
Aero	private	int	Name			
Aero	private	char	Name			
Ville	public	int	Numé			
Ville	private	char	Numé			
Escal	private	int	date			
Escal	private	int	date			
Vol	private	Vol	Numé			

Methode

nom_m	nom_c	type1	type2	type3
Aeroport	Prepare	void	static	public
Ville	localise	void	static	public
Vol_G	calcule	void	static	public
Vol	lance	void	static	public

Paramètre

nom_meth	nom_classe	type	valeur

Invocation

CL	CL	Me	Ps	St	Ab	Tai
Vol	Esc	cal	pub	static	1	

Figure 2. Différents tableaux des composants du système test

4 Calcul de la protéée maximale « Algorithme de Warshall »

Afin de déduire la porte maximale de l'impact de changement, on extrait du système (programme) en question le graphe correspondant dont les sommets sont les classes et on applique sur ce dernier l'algorithme de Warshall [5].

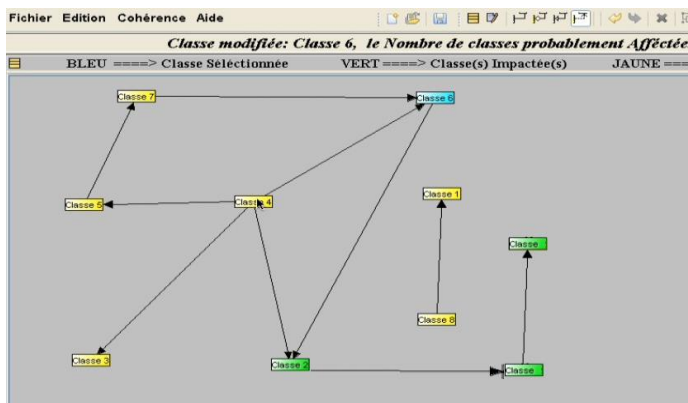


Figure 3. Graphe extrait du système considéré

On obtient la table 4 suivante après avoir porté les modifications sur l'ensemble des classes du système considéré.



Table 4. Résultats de l'impact de changement (Algorithme de Warshall)

Classes du système considéré	Algorithme de Warshall
Aéroport	2
Ville	4
Escale	2
Vol_Generique	5
Vol	2
Individu	6
Reservation	2
Passager	2
Client	2
compagnie_aerienne	2

Table 5. Résultats de l'impact de changement par le modèle

Classes du système considéré	Résultats sur la base du modèle		
	Changement de type de variable « S+L »	Changement de portée d'une method de "public" à "protected" « I~H »	Suppression de classe « H+G+S+I »
Aéroport	1	0	1
Ville	2	0	1
Escale	1	1	1
Vol_Generique	3	0	2
Vol	1	0	1
Individu	3	0	4
Reservation	1	0	1
Passager	1	1	1
Client	1	1	1
compagnie_aerienne	1	0	1

Table 6. Résultats de la vérification manuelle

Classes du	Résultats experimentation
------------	---------------------------

système considéré	Changement de type de variable	Changement de portée de "public" à "protected" d'une méthode	Suppression de classe
Aéroport	0	0	1
Ville	0	0	1
Escale	0	1	1
Vol_Generique	1	0	2
Vol	0	0	1
Individu	1	0	2
Reservation	0	0	1
Passager	0	1	1
Client	0	1	1
compagnie_aerienne	0	0	1

5. Résultat avec le modèle

Nous portons des changements au même système, et on calcule l'impact de changement selon le modèle. Le résultat est illustré dans la table 5 suivante.

6. Résultats avec vérification manuelle

Dans cette vérification, on porte les mêmes changements, et on corrige dans le code, puis on recompile jusqu'à ce que le système compile correctement et compte le nombre de classes corrigées, qui est en fait le nombre classes impactées. Dans cette étape, il se peut qu'on porte un changement et on n'effectue aucune modification dans le code [12].

7. Synthèse des résultats d'expérimentation

Afin de comparer les résultats d'impacts de changements générés par le modèle d'impact (Table 2) et ceux obtenus par

expérimentation (Table 6) ainsi que ceux obtenus en appliquant l'Algorithme de Warshall (Table 4), nous dressons la table de confrontation des résultats suivante :

Dans la Table 7, nous voyons bien que pour une classe donnée et un changement considéré, le nombre de classes impactées déduit par le modèle est toujours supérieur ou égale à celui obtenu manuellement (par expérimentation). Cela veut dire que l'expression booléenne générée par le modèle représente bien l'étendue de la propagation du changement, les résultats de l'algorithme de Warshall couvrent et confirment les affirmations en donnant une portée maximale de l'impact.

De même, si on considère le deuxième système il apparaît sur la Figure 4 que le nombre de changements prédits par le modèle est supérieur à celui obtenu par expérimentation. Aussi, la portée maximale est vérifiée pour le deuxième système.

Table 7. Table de confrontation des résultats

Classes du système considéré	Résultats sur la base du modèle			Résultats expérimentation			Porté maximale
	S+L	I~H	H+G+S+I	Changement de type de variable	Changement de portée de "public" à "protected" d'une méthode	Suppression de classe	Algorithme de Warshall
Aéroport	1	0	1	0	0	1	2
Ville	2	0	1	0	0	1	4
Escale	1	1	1	0	1	1	2
Vol_Generique	3	0	2	1	0	2	5
Vol	1	0	1	0	0	1	2
Individu	3	0	4	1	0	2	6
Reservation	1	0	0	0	0	1	2
Passager	1	1	1	0	1	1	2
Client	1	1	1	0	1	1	2
compagnie_aerienne	1	0	0	0	0	1	2

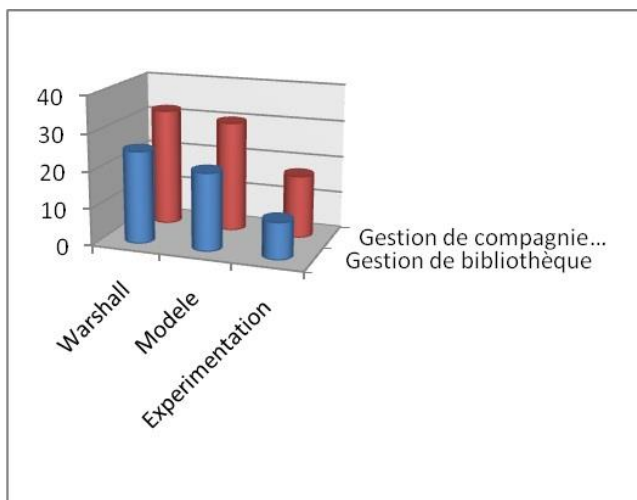


Figure 4. Résultats d'impact des trois approches

VII. Conclusion

Nous avons proposé dans cet article une étude comparative de trois approches de vérification d'impact de changement dans un système à objets afin de répondre à la problématique d'analyse et prédiction d'impact de changement. Nous avons comparé les résultats d'impacts de changements déduits par un modèle d'impact avec ceux obtenus manuellement (par expérimentation). Ensuite, afin de voir l'étendue de l'impact théorique, on a utilisé l'Algorithme de Warshall qui nous a permis d'avoir la portée maximale de l'impact. Les résultats obtenus montrent que la portée de l'impact réel est bien cadrée par celle donnée par le modèle d'impact faisant l'objet de la vérification, cela veut dire que l'impact déduit par le modèle représente bien la portée maximale de l'étendue de la propagation du changement réel. Nous envisageons d'autres études empiriques sur d'autres systèmes afin de généraliser les résultats obtenus.



Références

- [1] A.B 1. Baran, A.B., P.Brisebois,R.Côté, *La Ré ingénierie du logiciel :un Bref tour d'Horizon*. 1995(In le génie logiciel, reproduit des Actes de la conférence “Le génie Logiciel et ses applications“, huitièmes journées internationales (GL95), Paris).
- [2] 2. Vaucher, S., *Modelling Software Quality: A Multidimensional Approach*. 2010(Thèse présentée à la Faculté des arts et des sciences en vue de l’obtention du grade de Philosophiæ Doctor (Ph.D.) en informatique Université de Montréal).
- [3] 3. Michelle L.Lee, *change Impact Analsis of Object Oriented Software*. 1998. **PhD thesis, George Mason University Virginia USA**
- [4] 4. ABDI M.K, *Analyse et Prédiction d'impact de changement dans système à objet*. 2007(Thèse de doctorat d’état en Informatique, Université Es-Sénia d’Oran Algérie).
- [5] 5. DINEDANE, M.Z., *Prédiction de l'impact de changement dans un système à Objets*. 2008. **mémoire d'ingénieur d'état, Université d'oran Es-Sénia**
- [6] 6. Warshall S., *A theorem on Boolean matrices* Journal of the ACM, 1962. **9**(1): p. 11-12.
- [7] 7. Hind Kabaili, et al., *An Empirical StudyClass Cohesion Revisited: An Empirical Study on Industrial Systems*. 2000(In Proceedings of the Workshop on Quantitative Approaches in Object-Oriented Software Engineering,Sophia Antipolis and Cannes, France, Held in conjunction with the 14th European Conference on Object-Oriented Programming (ECOOP'2000).): p. pages 29-38.
- [8] 8. Abdeen, H., et al., *Learning dependency-based change impact predictors using independent change histories*. Information and Software Technology, 2015. **67**: p. 220–235.
- [9] 9. Chaumon, M.A., et al., *A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems*. 1998(in Proceedings of the Third European Working Conference on Software Maintenance and Reengineering,Amsterdam, the Netherlands): p. Pages 130-138.
- [10] 10. Hind Kabaili, Rouldof K.Keller, and F. Lustman, *Class Cohession as Predisctor of Changeability: An Empirical Study*. l'Objet, 2001.
- [11] 11. ABDI M.K, H. Lounis, and H. Sahraoui, *Analyse et prédiction de l'impact de changements dans un système à objets : Approche probabiliste*. 2009(In proc. of LMO 2009, Nancy, France).
- [12] 12. DAHANE, M., *Validation empirique d'un modèle d'impact de changement pour les systèmes à objets*. 2011. **mémoire de Magister, Université Es-Sénia d'Oran**.
- [13] 13. DAHANE M., A.M.K., *l'impact de changement dans un système à Objets : Verification empirique*. 2011(1st international conference on information systems and technologies ICIST'2011 - Tebessa, Algeria).
- [14]