UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN MATHÉMATIQUES
ET INFORMATIQUE APPLIQUÉES

PAR SAIDOU BALDÉ

SÉVÉRITÉ DES FAUTES ET COUPLAGE DES CLASSES DANS LES SYSTÈMES ORIENTÉS OBJET.

AOUT 2016

REMERCIEMENTS

Je rends grâce à la Providence de m'avoir guidé et inspiré dans toutes mes entreprises, et de m'avoir donné la force et le courage de mener à bout ce projet.

Je remercie mes professeurs encadrants Fadel Touré et Mourad Badri, sans qui ce travail n'aurait pas pu être réalisé. Vos conseils et votre soutien ont beaucoup été appréciés.

Je remercie profondément mes parents de m'avoir ouvert la porte de la vie, de m'avoir offert un foyer chaleureux, et d'avoir fait preuve de patience, de douceur et de bienveillance envers moi.

J'adresse un remerciement spécial à la famille Wade pour qui j'ai une affection profonde. Vous m'avez accueilli et soutenu durant toutes ces années et je vous suis très reconnaissant.

À toutes les familles qui m'ont offert leur accueil et leur soutien, je vous remercie. La famille Beloin, la famille Côté et la famille Brand, c'est grâce à vous si j'ai pu m'installer et mener à bien mes études et mon séjour au Québec. La famille Diallo et la famille Diao, je vous exprime ma profonde gratitude. Votre générosité et votre gentillesse sont pour moi une source d'inspiration.

Mes sincères remerciements à mes amis, pour leur soutien indéfectible et leurs encouragements, ainsi qu'à mes frères et sœurs, qui ont toujours été présents pour moi, surtout dans les moments difficiles. Je vous aime.

RÉSUMÉ

Les systèmes logiciels sont devenus incontournables dans la plupart des secteurs de l'activité humaine. Leur taille et leur complexité ne cessant de croître, assurer leur développement et leur maintenance est devenu un défi. Plusieurs défaillances sont décelées après la mise en production d'un logiciel. Les corrections nécessitent, une modification du code source. Les fautes qui en sont à l'origine des défaillances n'ont pas la même gravité. Les fautes ayant un impact jugé trop important, ou un coût de correction jugé trop élevé, seront catégorisées comme *sévères*, et les parties du code à leur origine seront considérées comme *critiques*. Maintenir minimal le taux de fautes de haut niveau de sévérité est une des préoccupations de l'assurance qualité du logiciel. Pour atteindre cet objectif, les modèles de qualité basés sur les métriques logicielles constituent une approche intéressante. Par ailleurs, le paradigme OO offre une variété de métriques qui sont liées à la plupart des attributs externes de qualité, dont la prédisposition des classes aux fautes. Plusieurs travaux ont investigué les liens entre les métriques et la prédisposition des classes aux fautes, mais très peu ont considéré l'aspect sévérité des fautes, encore moins, le lien entre le couplage et la sévérité. Dans ce mémoire, nous avons étudié les relations entre le couplage et la prédisposition des classes aux fautes sévères. Nous avons recueilli des données sur les fautes et leurs sévérités en analysant l'information issue des systèmes de suivi de bogues et de gestion de versions. Nous avons calculé les métriques de code source des systèmes étudiés. Nous avons conduit plusieurs expérimentations avec des méthodes statistiques et d'apprentissage automatique. L'objectif était d'identifier les différentes formes de couplage et leurs liens avec la présence de fautes sévères. Nous avons mis en évidence trois formes de couplage que sont le couplage entrant, le couplage sortant, et le couplage par héritage. Nous avons trouvé que le couplage sortant avait un lien plus significatif avec la prédisposition des classes aux fautes sévères. Nous avons aussi montré que les métriques de couplage, utilisées dans des modèles de prédiction, étaient capables d'identifier une grande partie des classes critiques.

ABSTRACT

Nowadays, software systems have become essential in all industries. Many failures are detected after the software release. Their corrections require the source code modification. However, the faults that cause these failures have not the same severity. The faults having an important impact or cost of correction will be categorized as *severe*, and the related parts of the source code will be considered as *critical*. Minimizing the rate of high-severity faults is an important quality assurance concern. To achieve this goal, using quality models based on software metrics is an interesting approach. Moreover, the object-oriented paradigm offers a large variety of metrics, which are implicitly related to the most of external quality attributes, including fault proneness.

In this context, several studies have been conducted to investigate the relationships between metrics and fault proneness of classes, but few of them have considered the severity dimension of the faults, even less the particular effect of OO coupling.

In this thesis, we empirically investigated the relationships between coupling and high-severity fault proneness of classes, through three systems developed in JAVA. For this, we collected data of faults and faulty classes by mining information on bug tracking and version control systems, and calculating source code metrics of the systems under study. With the resulting data, we conducted several experiments using statistical and machine learning techniques, in order to identify several kinds of coupling and how they could influence high severity fault proneness of classes. We found that the import coupling had a more significant effect on the probability of having high severity faults in classes. We also showed that coupling metrics, used in prediction models, were able to identify a large part of critical software classes.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1: Introduction

Dans le paradigme objet, un logiciel peut être considéré comme un ensemble d'objets qui collaborent de façon cohérente pour produire le comportement voulu et donner les résultats attendus. Ce comportement et ses résultats répondent à des spécifications bien définies. Le niveau de collaboration des classes augmente avec la taille et la complexité du logiciel et rend difficile sa maintenabilité. Or, de nos jours, l'omniprésence des logiciels fait que leur taille et leur complexité ne cessent d'augmenter alors que leur utilisation dans des domaines sensibles augmente les exigences de qualité. Cette réalité représente un vrai défi dans l'ingénierie logicielle.

1.1 Problématique

Dans le processus logiciel, les tests occupent plus de la moitié des ressources, et ne garantissent pas pour autant la fiabilité totale du système [Chauhan 12]. Pour les systèmes de grande taille (des milliers, voire des centaines de milliers de lignes de code), il est quasiment impossible de réaliser une couverture de test complète. Or, vu que certaines fautes sont plus tolérables que d'autres, il serait très utile d'avoir une approche permettant de prioriser les tests afin de cibler les classes les plus critiques, aussi bien pour des soucis d'efficience (concernant le déploiement des ressources de test), que de fiabilité (du produit final).

Le processus d'assurance qualité va au-delà de la phase de test. En effet, la maintenance constitue la phase la plus longue du processus logiciel. Après livraison, beaucoup de tâches sont effectuées pour assurer l'amélioration, la non-régression et l'évolution du produit logiciel afin de conserver son exploitation de façon durable.

La principale difficulté de la maintenance est la mise en œuvre des changements à apporter au logiciel tout en évitant une dégradation de sa qualité. Parmi ces changements, il y a la correction des fautes (dites fautes « post release »), qui se trouvent être plus difficiles à localiser et à corriger que celles détectées avant la sortie du logiciel, pendant la phase de test (fautes « pre-release ») [Shatnawi 08].

Ces fautes, qui n'ont pas été détectées lors de la phase de test (pour diverses raisons), engendrent des erreurs qui surviennent pendant l'exploitation du logiciel et peuvent conduire à des défaillances plus ou moins sévères. Ainsi, en partant de l'hypothèse selon laquelle la sévérité des fautes aurait une importance plus grande que leur nombre [Shatnawi 08], nous allons nous intéresser aux attributs qui pourraient influencer la présence de fautes sévères dans les systèmes orientés objet.
Dans le paradigme orienté objet (OO), le couplage est l'un des attributs logiciels mettant en évidence la collaboration entre les objets d'un système logiciel. Il serait alors intéressant de voir comment, et à quel point, il impacte la sévérité des fautes.
Plusieurs travaux dans le domaine se sont concentrés sur l'étude des liens entre les métriques OO et la prédisposition aux fautes (voir section 2.1), mais très peu d'entre eux ont pris en considération la dimension de la sévérité des fautes et encore moins l'effet tout particulier du couplage sur celles-ci.

1.2 Approche

Avec une approche basée sur les métriques, nous allons mener une étude empirique pour explorer les liens entre le couplage et la sévérité des fautes dans les systèmes OO en utilisant des modèles d'analyse statistique et d'apprentissage automatique.

1.3 Plan

Le travail effectué à travers ce mémoire sera présenté comme suit :
Dans la première partie, nous allons présenter quelques travaux connexes afin de donner un aperçu de l'état de l'art.
Dans la deuxième partie, nous allons présenter les métriques OO et plus particulièrement le couplage (sous ses différentes formes).
La troisième partie concernera la description de la collecte des données en passant en revue les outils utilisés et la procédure d'extraction des données.
Dans la quatrième partie, nous allons aborder les expérimentations, en présentant les concepts théoriques, notre démarche, puis nous présenterons et discuterons les résultats obtenus.

Les deux dernières parties nous permettront de faire un bilan sur les menaces à la validité (*threats of validity* : TOV), de présenter les conclusions générales par rapport aux résultats obtenus, et de dégager les perspectives de recherches futures comme suite à ce travail.

Chapitre 2: État de l'art

2.1 Introduction

L'étude de la prédisposition des classes aux fautes (PCF) avec une approche basée sur les métriques s'est fortement développée au cours des dernières années. Outre l'augmentation des outils de calcul des métriques, la mise à disposition de répertoires publics de données comme PROMISE[1] (ou le programme de métriques de la NASA), a beaucoup contribué au développement de cet axe de recherche. Les données en question sont à la fois des informations sur les fautes détectées et les métriques de code de systèmes logiciels. Elles sont utilisées pour mener des expérimentations dont le but est d'évaluer les relations (potentielles) entre les métriques et la prédisposition aux fautes grâce à diverses méthodes d'analyse.

2.2 Sources des données

Comme mentionné dans [Radjenović 13], les données utilisées dans les études peuvent être : (1) Privées, donc non disponibles, (2) Partiellement publiques, autrement dit disponibles en partie, ou à l'état brut et nécessitent des prétraitements pour recouper l'information, ou (3) Publiques, i.e. totalement disponibles et prêtes à l'usage. La disponibilité des données est importante, car elle permet la réplication des études et d'ainsi garantir la validité des résultats. Cet aspect a fait que les logiciels open source sont très largement utilisés comme sujets d'étude dans la prédiction de la prédisposition aux fautes, étant donné que leur code source ainsi que toutes les informations relatives à leur *Configuration Management* (CM) sont disponibles.

2.3 Les suites de métriques

Il existe une large variété de métriques OO qui peuvent être utilisées pour mener des études empiriques sur la prédisposition des classes aux fautes. Parmi celles qui reviennent le plus souvent, on peut citer : Abreu et Carapuca (MOOD : Metrics for

[1] (Predictor Models in Software Engineering)
http://promise.site.uottawa.ca/SERepository/index.html

Object Oriented Design) [Abreu 94,96], Bansiya et Davis (QMOOD : Quality Metrics for Object Oriented Design) [Bansiya 02], Bieman et Kang [Bieman 95], Briand et al. [Briand 97], Cartwright et Shepperd [Cartwrigth 00], Chidamber and Kemerer (CK) [Chidamber 94-96], Etzkorn et al. [Etzkorn 99], Halstead [24], Henderson-Sellers [Henderson 96], Hitz et Montazeri [Montazeri 95], Lee et al. [Lee 95], Li [Li 98], Li et Henry (LH) [Li 93a-b], Lorenz and Kidd (LK) [Lorenz 94], McCabe [McCabe 76], Tegarden et al. [Tegarden 95].

Plusieurs facteurs entrent en considération quant au choix d'une suite de métriques plutôt qu'une autre pour conduire des études empiriques sur la PCF (prédisposition des classes aux fautes). Le facteur le plus souvent évoqué est la validation empirique déjà effectuée sur celles-ci à travers des études antérieures. Olague et al. [Olague 07] ont trouvé que la suite QMOOD, contrairement à MOOD, était souhaitable pour la prédiction de la prédisposition aux fautes tout en mentionnant que la suite CK donnait de meilleurs résultats que QMOOD et MOOD. Les métriques de Briand et al. [Briand 97] ont été validées pour leur pertinence pour la prédisposition aux fautes dans plusieurs études [Briand 98, Briand 97, Briand 01, Briand 00, Briand 02, Aggarwal 07, Aggarwal 09]. Celles proposées par Henderson-Sellers semblent être moins efficaces [Radjenović 13].

2.4 Les méthodes d'analyse

On retrouve dans la littérature différentes approches pour étudier les liens entre les métriques OO, qui vont être les variables indépendantes ou explicatives, et la prédisposition aux fautes, qui sera la variable dépendante ou à expliquer. Parmi les plus communément utilisées, nous pouvons citer :

L'analyse de corrélation : Elle est utilisée pour mettre en évidence des dépendances linéaires entre des variables prises deux à deux. La corrélation de Spearman, utilisant les rangs des variables et requérant moins d'hypothèses sur les observations, est largement utilisée.

La régression logistique (RL) : Il s'agit de la technique statistique la plus largement utilisée dans la littérature [Chauhan 12]. Elle permet de construire un modèle de prédiction de la prédisposition des fautes. Elle peut être univariée (une

variable explicative) ou multivariée (plus d'une variable explicative). Dans le premier cas, on cherche à voir l'effet d'une métrique OO sur la prédisposition des fautes et dans le second l'effet combiné de plusieurs.

Les méthodes d'apprentissage automatique : Ce sont des techniques basées sur des algorithmes d'intelligence artificielle qui permettent de faire de la classification ou du regroupement. Une large variété de ces techniques est utilisée pour construire des modèles de prédiction de la prédisposition aux fautes : Decision Tree (DT), Artificial Neural Network (ANN), Radial Basis Function (RBF), Support Vector Machine (SVM), Baysian Networks (BN), k-nearest neighbor (K-NN ou IBK : instance-based K-NN), etc.

2.5 Les fautes dans les Systèmes OO – Travaux connexes

Dans cette partie, nous allons effectuer un survol de ce qui a été fait en termes de recherches empiriques sur les liens entre les métriques OO et la prédisposition des classes aux fautes. Nous allons citer quelques travaux marquants, dans un ordre chronologique, en mettant l'accent sur les métriques utilisées, les techniques d'analyse, les sources des données et les résultats obtenus.

En 1999, Tang et al. [Tang 99] ont étudié les corrélations entre les métriques OO (CK) et la prédisposition aux fautes à travers trois applications industrielles de temps réel, développées en C++, en utilisant l'analyse de régression logistique univariée. Ils ont trouvé que seules les métriques de complexité (WMC : Weighted Methods per Class) et de couplage (RFC : Response For a Class) étaient assez significativement liées à la prédisposition aux fautes.

En 2000, Briand et al. [Briand 00] ont extrait 49 métriques pour identifier un modèle idéal pour la prédiction de la prédisposition des classes aux fautes. L'étude a porté sur huit systèmes de taille moyenne développés en C++ par des étudiants (gradués et non gradués) contenant un total de 180 classes. Ils ont utilisé l'analyse de régression univariée et multivariée pour évaluer l'impact individuel puis combiné des métriques OO sur la prédisposition aux fautes. Les résultats ont

montré que toutes les métriques, excepté la métrique d'héritage NOC (Number of Children), étaient des prédicteurs significatifs.

En 2002, Yu et al. [Yu 02] sont partis d'une base de huit métriques, à savoir les six métriques de la suite CK, le couplage entrant (Fan-in), et la métrique de taille LOC (Lines Of Code), pour examiner leurs relations avec la prédisposition aux fautes. Ils ont introduit deux autres métriques CBOin et RFCin afin de distinguer le couplage par héritage (RFCin, CBOin) des autres formes du couplage (CBOout, RFCout). L'objet de l'étude était la partie cliente d'un large système de gestion de services réseau (développé en JAVA) composée d'un ensemble de 123 classes totalisant environ 34000 lignes de code. Ils ont utilisé l'analyse de régression (linéaire) et l'analyse discriminante afin d'identifier les métriques les plus significatives pour la prédiction de la prédisposition aux fautes. Ils ont trouvé que CBOin, RFCin, et DIT n'étaient pas assez significatives contrairement aux autres métriques qui étaient significatives, mais à des degrés différents. Ils ont aussi trouvé avec l'analyse multivariée que les métriques étaient plus efficaces pour la prédiction à la prédisposition aux fautes quand le nombre de ces dernières n'est pas pris en compte.

En 2005, Gyimothy et al. [Gyimothy 05] ont validé empiriquement les métriques OO (CK) sur un large logiciel open source (Mozilla) pour la prédiction des fautes. Ils ont employé les régressions (linéaire et logistique) et les méthodes d'apprentissage automatique (ANN et DT) afin de construire leurs modèles de prédiction. Ils ont conclu que seule la métrique NOC n'était pas un prédicteur assez significatif pour la prédisposition aux fautes et que le couplage (CBO) était particulièrement significatif.

En 2006, Zhou et al. [Zhou 06] ont été les premiers à considérer la dimension de la sévérité dans l'étude de la prédisposition des classes aux fautes. Ils ont utilisé les méthodes de régression logistique et d'apprentissage automatique pour montrer comment les métriques OO et la prédisposition aux fautes sont liées quand la sévérité des fautes est prise en compte. Ils ont divisé les classes en deux catégories : celles étant affectées de fautes sévères et les autres. L'étude a été

conduite sur des données publiques de la NASA (KC1) avec la suite de métriques CK. Les métriques WMC, CBO et la SLOC se sont révélées être de bons prédicteurs à la prédisposition aux fautes pour les différents niveaux de sévérité.

En 2008, Shatnawi et al. [Shatnawi 08] ont investigué la capacité des métriques à prédire la probabilité de présence de fautes dans les classes lors du processus d'évolution du logiciel en considérant cinq niveaux de sévérité. Ils ont utilisé les suites de métriques CK et LK, et les données ont été recueillies sur trois versions du projet Eclipse. En appliquant des modèles de régression logistique univariés et multivariés (binomiaux et multinomiaux), ils ont constaté que la précision diminuait de version en version et que des méthodes alternatives (autre que les modèles de prédiction basés sur les métriques) pourraient être nécessaires pour des prédictions plus précises. Leurs résultats ont montré que les métriques de complexité (WMC) et de couplage (RFC, CBO, CTA et CTM) se trouvaient être de bons prédicteurs pour chaque version et chaque niveau de sévérité. Ils ont souligné aussi qu'il serait plus pratique de se baser sur la sévérité des fautes plutôt que sur leur nombre pour ordonner les classes dans le but d'identifier celles qui sont les plus critiques.

En 2009, Aggarwal et al. [Aggarwal 09] ont fait une étude analogue à celle conduite par Briand et al. Ils ont étudié et validé des métriques capturant différents aspects de la conception OO dont le couplage, l'héritage, la cohésion, l'encapsulation, et le polymorphisme. Ces métriques ont été analysées à travers une application JAVA et la régression logistique a été utilisée comme méthode de prédiction. D'après leurs résultats, le couplage semble être un bon prédicteur de la prédisposition aux fautes, plus spécifiquement le couplage à travers l'invocation de méthodes (couplage sortant).

En 2010, Singh et al. [Singh 10] proposent une approche différente de celle de [Zhou 06] en catégorisant les fautes en trois niveaux de sévérité : Haut (Hight), Moyen (Medium), et Faible (Low). Outre les méthodes statistiques standards (analyses de régression), ils ont utilisé aussi les méthodes d'apprentissage automatique (DT et ANN) pour prédire l'effet des métriques OO sur la

prédisposition aux fautes. Les métriques CBO, WMC, RFC, et SLOC se sont révélées être très significatives pour les différents niveaux de sévérité, mais le modèle construit pour les fautes de haute sévérité avait une plus faible précision que les modèles construits avec les sévérités de niveaux moyen et faible. Ils ont constaté aussi que les techniques d'apprentissage automatique donnaient de meilleurs résultats que la régression logistique.

Toujours en 2010, Malhotra et al. [Malhotra 10] ont construit un modèle SVM (Support Vector Machine) pour trouver les relations entre les métriques OO (de CK) et la prédisposition aux fautes, à différents niveaux de sévérité, et en utilisant des données publiques de la NASA (projet KC 1). L'objet de l'étude était un système OO développé en C++ formé d'un ensemble de 145 classes. Les performances du modèle construit pour la prédiction des fautes de haute sévérité étaient plus faibles que celles pour la prédiction des fautes de faible et de moyenne sévérité. Le modèle SVM avait une précision et un rappel assez élevés. Ses performances surclassaient les méthodes basées sur les réseaux de neurones et étaient comparables à la méthode basée sur les arbres de décision dans [Singh 10].

En 2012, Rathore et al. [Rathore 12] ont évalué la capacité des attributs de conception OO reliés au couplage, à la cohésion, à la complexité, à l'héritage et à la taille, à travers leurs métriques respectives, pour la prédiction à la prédisposition aux fautes, sur une base indépendante et combinée. Ils ont utilisé les données (du projet sous le nom de code PROP) du répertoire public de données PROMISE pour investiguer dix-huit métriques de différentes suites (CK, Tang, QMOOD, HENDERSON-SELLERS, McCabe) avec les méthodes classiques de régression logistique et des méthodes d'apprentissage automatique (IBK, RF et NB). Dans leurs résultats, le couplage, la complexité et la taille étaient les attributs de conception qui pouvaient influencer de façon significative la prédisposition aux fautes. Les modèles de prédiction construits avec les métriques de couplage et de complexité avaient une meilleure précision que les modèles construits avec le reste des métriques.

En 2013, Oyetoyan et al. [Oyetoyan 13] ont évalué empiriquement deux systèmes non triviaux (Apache-ActiveMQ en JAVA et CommApp en C#), en utilisant les métriques de cycle définies à partir du graphe de dépendance des modules/composants. Ils ont abordé les composants selon qu'ils appartiennent ou non à une dépendance cyclique. Ils démontrèrent que les composants appartenant à une relation cyclique représentaient une part significative des composants à fautes sévères (SDC). D'après leurs résultats, entre 95% et 100% des fautes de sévérité critiques pouvaient être découvertes dans les classes appartenant à une relation de dépendance cyclique, ce qui représentait entre 82% et 90% des composants/modules comportant des fautes. Cela suppose que le couplage a un impact déterminant sur la sévérité des fautes, étant donné qu'un SDC est en fait un composant qui est couplé à lui-même de façon directe ou indirecte.

2.6 Conclusion

En comparant plusieurs études, nous avons remarqué que la suite de métriques de CK se trouve être la suite de métriques la plus fréquemment utilisée dans la littérature [Chauhan 12, Radjenović 13, Malhotra 16]. Elle a été validée par plusieurs études [Basili 96, Chowdhury 11, Elish 11, Ambros 11, Kpodjedo 11]. Ces métriques se sont révélées être les plus significatives dans la prédiction à la prédisposions aux fautes [Radjenović 13], et parmi les plus significatives d'entre elles nous retrouvons CBO RFC et WMC [Basili 96, Briand 00, Briand 01, Yu 02, Gyimothy 05, Zhou 06, Olague 07, Pai 07, Shatnawi 08, Aggrawal 07, Singh 10]. Le couplage apparaît donc comme un indicateur pertinent pour la prédisposition aux fautes, d'où l'intérêt que nous lui portons à travers cette étude.

Dans l'étude menée par Radjenović et al. [Radjenović 13], l'analyse de régression (logistique et linéaire) était largement la technique d'analyse la plus utilisée (68%), suivie des méthodes d'apprentissage automatique (24%) et de l'analyse de corrélation (8%). Les méthodes d'apprentissage automatique semblent être très prometteuses en termes de performances. Nous les inclurons les deux approches dans notre étude.

En parcourant les travaux effectués dans le domaine, on s'aperçoit aussi que le couplage s'est toujours révélé être significativement lié à la présence de fautes dans les systèmes OO. Cependant, aucune de ces études ne s'est intéressée à l'effet tout particulier des différents types de couplage sur la prédisposition des classes aux fautes sévères. Du moins, pas à notre connaissance au moment où nous avons entamé ce travail.

Chapitre 3: Les métriques OO

En génie logiciel, comme dans toutes autres disciplines d'ingénierie, la mesure est indispensable pour le contrôle des processus et l'assurance de la qualité. Il se trouve que les caractéristiques du paradigme OO offrent un grand nombre d'attributs de qualité du logiciel, que l'on regroupe en deux catégories : les attributs internes tels que le couplage, la complexité, la taille, et les attributs externes tels que la maintenabilité, la portabilité, la robustesse, la réutilisabilité, etc.

Les attributs externes étant difficilement mesurables de façon directe. Il est d'usage de passer par la mesure des attributs internes à travers les métriques logicielles, sachant qu'il existe des liens évidents entre ces dernières. D'ailleurs, la mise en évidence de ces liens est l'un des domaines de recherche les plus importants sur les métriques logicielles. Les métriques OO permettent, en effet, de mesurer directement les attributs internes du logiciel en prenant en considération les artéfacts propres au paradigme OO comme l'encapsulation, l'héritage, les classes, etc.

Les industriels portent un intérêt croissant aux métriques logicielles aussi bien pour des enjeux de productivité que d'assurance qualité. Il est possible, entre autres, de :

- Faire des études prédictives pour, par exemple, identifier les composants à risque et prendre des décisions en conséquence comme prioriser les tests afin de faire des économies en ressources (souci économique), ou augmenter la rigueur des tests sur ces composants (souci de qualité, de robustesse).
- Faire des études rétrospectives dans le but, par exemple, d'améliorer le processus de développement logiciel.

3.1 Définitions

Dans le paradigme OO, un objet est un concept logique permettant de représenter n'importe quelle entité concrète ou abstraite, en lui attribuant un identifiant, un état et un comportement. Chaque Objet est une instance d'une classe qui est un concept abstrait représentant les objets ayant les mêmes attributs et les mêmes méthodes

(mêmes propriétés). Les attributs (propriétés) d'un objet définissent son état, ses méthodes, son comportement. Deux objets de la même classe sont différenciés grâce à leur identifiant même s'ils ont le même état. Le comportement d'un objet est conditionné par son état, et cet état est modifié à travers ses comportements.

En partant de cette brève définition, nous allons présenter les métriques relatives aux concepts OO (attributs internes) qu'elles sont censées capturer. Nous nous limiterons aux métriques de code statiques mesurées au niveau classe.

3.1.1 La taille

Le nombre de lignes de code (LOC) est la métrique traditionnelle d'un code source logiciel indépendamment du paradigme. Le concept OO a permis d'introduire d'autres mesures de la taille comme NOA (Number of Attributes) [Lorenz 94], qui compte le nombre de variables (d'instance et de classe) pour une classe, NOM (Number of Methods) [Bansiya 02] qui compte le nombre de méthodes que contient une classe. Dans notre étude nous utiliserons :

NPM (Number of Public Methods) qui compte le nombre total de méthodes déclarées publiques dans une classe. Elle peut être utilisée pour mesurer la taille d'une API proposée par un paquetage (CIS : Class Interface Size dans QMOOD).

3.1.2 La complexité

Comme son nom l'indique, la complexité d'une classe révèle à quel point il peut être difficile de l'implémenter, de la comprendre, de la tester, etc. Il s'agit donc d'un concept assez large qui intègre plusieurs interactions internes et qui est très étroitement lié au couplage (notamment celui sortant). Il s'agit d'une des mesures les plus utilisées à l'instar de LOC (métrique traditionnelle). Plus une classe est complexe, plus elle est susceptible de présenter des fautes (probabilité).

WMC [Chidamber 94] : elle mesure la complexité en faisant la somme des méthodes pondérées par leur complexité cyclomatique (CC [McCabe 76]). Elle devient une simple mesure du nombre de méthodes quand la même pondération (1) est fixée pour toutes les méthodes.

3.1.3 La cohésion

La cohésion illustre à quel point les éléments d'une classe sont liés entre eux. Une classe est dite cohésive s'il y a une bonne collaboration entre ses composants (variables membres, méthodes). Une classe non cohésive est souvent investie de responsabilités disparates.

LCOM [Chidamber 94] : sa définition se base sur la notion de degré de similarité des paires de méthodes d'une classe, qui vaut 1 si les deux méthodes accèdent à une même variable membre et 0 sinon. Dans une classe, en notant P l'ensemble des paires de méthodes dont la similarité est nulle, et |Q| l'ensemble des paires de méthodes dont la similarité est non nulle :

$$LCOM = |P| - |Q| \text{ si } |P| > |Q|, \text{ 0 sinon}$$

3.1.4 L'héritage

Il s'agit de la mesure directe des relations parent-enfant dans la hiérarchie d'héritage du système OO. Elle caractérise les classes selon leur profondeur dans leur arbre d'héritage ou leur nombre de sous-classes ou encore leur nombre de parents (superclasse). Plusieurs métriques sont proposées dans la littérature :

DIT : Proposée dans la suite de CK, elle mesure la profondeur d'une classe dans l'arbre d'héritage en comptant le nombre d'ascendants (superclasses), de la classe à la racine de l'arbre. Dans le cas de l'héritage multiple, DIT mesure la branche qui compte le plus de superclasses.

NOC : Proposée dans la suite de CK, elle compte le nombre de classes qui héritent directement d'une classe donnée dans la hiérarchie d'héritage.

NOP : Proposée par Lorenz et Kidd [Lorenz 94], cette métrique compte le nombre de classes desquelles hérite directement une classe donnée dans la hiérarchie d'héritage.

NOD : Cette métrique est une autre proposition de Lorenz et Kidd [Lorenz 94] qui mesure le nombre de descendants d'une classe en comptant le nombre de classes qui héritent directement ou indirectement d'une classe.

NA : Cette métrique est une proposition de Tegarden et al. [Tegarden 92, Tegarden 95] qui compte le nombre d'ancêtres d'une classe, à savoir le nombre de classes dont elle hérite directement ou indirectement.

3.1.5 Le couplage

Le couplage est défini comme étant le degré d'interdépendance des composants logiciels. Cette interdépendance se manifeste à travers l'accès aux variables, les appels de méthodes et les liens d'héritage et d'interface. Ainsi, plusieurs formes de couplages existent. Nous les aborderons en détail dans la section 3.2.

Le couplage se mesure en comptant le nombre de messages passés entre les classes. Une classe communique avec une autre par voie de messages, par exemple les appels de méthodes. Cela conduit à une distinction du couplage, selon le sens des appels, entre :

- **Couplage entrant** : Pour une classe, il reflète le nombre de classes la référençant. Il montre le niveau d'utilisation d'une classe par le reste des classes du système. Un couplage entrant très élevé est signe de criticité. La classe/composant devrait être stable/fiable puisqu'étant largement utilisée par le reste du système. C'est aussi un indice de réutilisabilité et souvent signe d'abstraction élevée (Interfaces, bibliothèques externes …). Le Fan-in ou *afferent coupling* (Ca) est la métrique traditionnelle du couplage entrant.
- **Couplage sortant** : qui, par opposition à celui entrant, reflète le nombre de classes auxquelles fait référence une classe donnée. Il caractérise le niveau de dépendance d'une classe vis-à-vis du reste du système. Le fan-out ou *efferent coupling* (Ce) est la métrique traditionnelle du couplage sortant.

La figure 1 donne un exemple de calcul de Fan-in et de Fan-out dans un système de 5 classes (ABCDE).

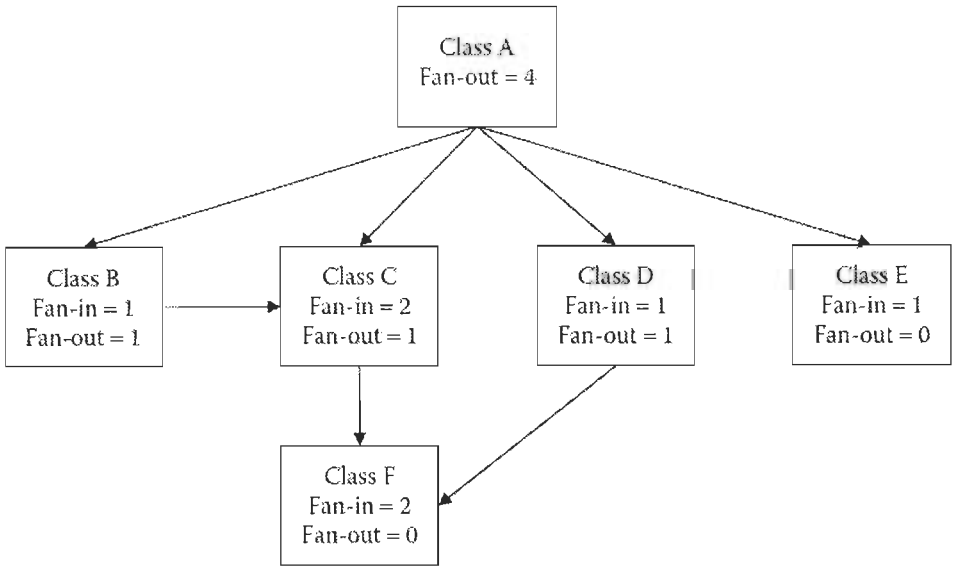


Figure 1 : Exemple de calcul Fan-in/Fan-out.

Plusieurs métriques ont été proposées dans la littérature pour mesurer le couplage. Elles se distinguent par les types de couplage qu'elles mesurent et/ou les formes de couplage retenues dans la définition du couplage de leurs auteurs respectifs. Nous allons présenter, dans ce qui suit, celles que nous utiliserons pour notre étude :

CBO [Chidamber 94] : CBO compte le nombre de classes auxquelles une classe est couplée. Deux classes sont couplées si une méthode définie dans l'une utilise des méthodes ou des variables d'instance définies dans l'autre. Le couplage par héritage étant inclus dans cette définition (exemple à la figure 2).

Ca (Afferent coupling) mesure le couplage entrant, dans un contexte où celui-ci a la même définition que pour CBO. Le Ca d'une classe compte le nombre de classes qui utilisent cette dernière.

Ce (Efferent coupling) mesure le couplage entrant, dans un contexte où celui-ci a la même définition que pour CBO (et pour Ca). Le Ce d'une classe compte le nombre de classes qu'utilisent cette dernière.

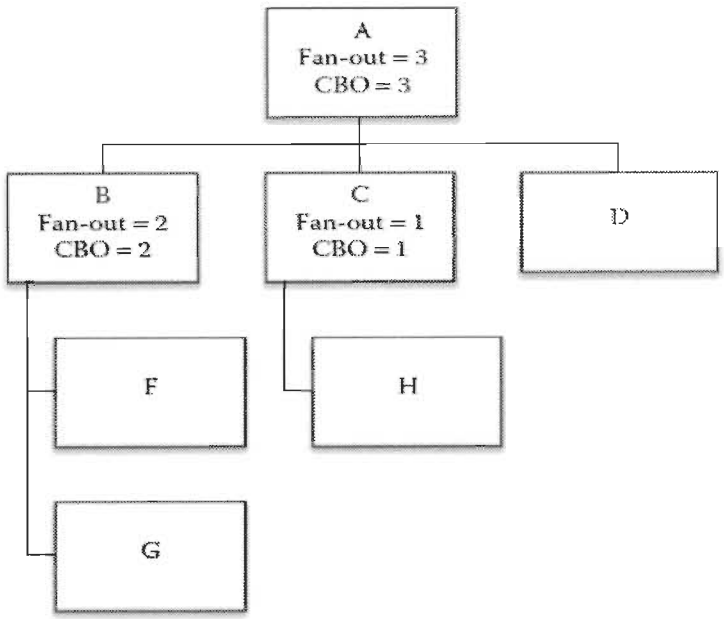


Figure 2: Exemple de calcul du CBO.

MOA (Measure of Aggregation, QMOOD) compte pour une classe donnée, le nombre d'attributs dont les types sont d'autres classes du système. Elle représente le couplage par utilisation de classe comme attribut d'instance d'une autre classe. Il met donc en évidence la relation de type partie-tout (agrégation, partie d'un tout).

IC (Inheritance Coupling) donne le nombre de classes parents auxquelles une classe est couplée. Une classe est couplée à son parent si une de ses méthodes héritées dépend fonctionnellement d'une nouvelle méthode (ou redéfinit) de la classe en question. Cela peut être, par exemple, une méthode héritée invoquant une méthode redéfinie (ou bien l'inverse).

CBM (Coupling Between Methods) donne pour une classe, le nombre total de nouvelles méthodes (ou méthodes redéfinies) auxquelles toutes les méthodes héritées sont couplées. Le couplage étant défini de la même façon que pour IC.

RFC (Response For a Class) est une autre proposition de CK qui compte, pour une classe donnée, l'ensemble des méthodes qui peuvent être exécutées en réponse à un message passé à la classe (invocation d'une méthode de la classe). Il s'agit du cardinal de l'union entre l'ensemble des méthodes définies dans la classe, et l'ensemble des méthodes externes à la classe (mais invoquées à l'intérieur des

méthodes de celle-ci). Appelons cet ensemble E_r, défini par $E_r = I_i \cup_{i,j}\{E_{ij}\}$ avec I_i = ensemble des méthodes M_i de la classe, et E_{ij} l'ensemble des méthodes M_j invoquées par les M_i. Alors RFC = $|E_r|$. Il serait souhaitable lors du calcul de RFC d'inclure les appels de méthodes indirectes en calculant la fermeture transitive du graphe d'appel des méthodes de la classe. Mais, la difficulté d'implémentation de cette approche et son coût d'exécution font qu'on trouve rarement des outils qui calculent RFC de cette façon. RFC mesure en partie le couplage sortant à travers l'appel de méthodes (voir Figure 3).

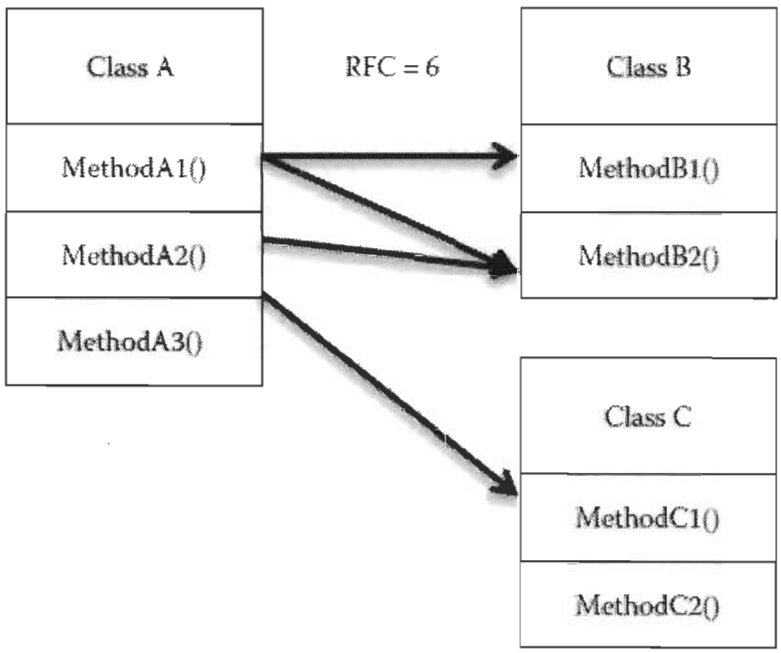


Figure 3: Exemple de calcul de RFC.

3.2 Étude détaillée du couplage

3.2.1 Formes de couplage

Dans la littérature, plusieurs outils ont été proposés pour mesurer le couplage dans les systèmes OO ([Elder 94, Montazeri 95, Briand 99]). On y dégage les différents types de couplage, la force des liens qui régissent chaque type de couplage et la façon de mesurer concrètement ces couplages à travers différentes métriques.

Eder [Eder 94] a divisé les liens entre les classes en trois types de relations, sur la base desquelles il a proposé les formes de couplage suivantes :

- Couplage d'interaction
- Couplage de composant
- Couplage d'héritage

Hitz et Montazeri [Montazeri 95] ont proposé de grouper distinctement deux formes de couplage :

- Couplage niveau classe (CLC)
- Couplage niveau objet (OLC)

En se basant sur tout ce qui a été proposé, Briand et al. [Biand 99] proposent un Framework unifié de mesure du couplage. Il caractérise la mesure du couplage selon six critères :

Le Type de connexion : Il définit le mécanisme par lequel deux classes interagissent. En considérant deux classes C et D ($C \neq D$), voici les types de connexions possibles :

1. Un attribut *a* de C est de type D.
2. Une méthode *m* de C a un paramètre d'entrée de type D ou retourne un objet de type D.
3. Une méthode *m* de C a une variable locale de type D.
4. Une méthode *m* de C invoque une méthode qui a un paramètre de type D.
5. Une méthode *m* de C référence un attribut *a* de D.
6. Une méthode *m* de C invoque une méthode *m'* de D.
7. Autres liens de dépendance de haut niveau de design (HLD : Hight Level Design) entre C et D.

Le choix d'un type de connexion dans la définition d'une mesure de couplage dépend non seulement de la phase de développement, mais aussi de l'objectif visé par la mesure, car il faut noter que les connexions n'ont pas la même force. Par exemple, la métrique MOA ne mesure que le type de connexion 1, alors que RFC ne calcule que le type 6. D'autres métriques (comme CBO) mesurent plusieurs types de connexions à la fois.

Le sens d'impact : Il permet de faire la distinction entre couplage entrant et sortant.

Un fort couplage sortant indique qu'un composant/classe dépend fortement d'autres composants/classes du système, ce qui réduit sa compréhensibilité, vu qu'il importe aussi une partie de la logique des composants qu'il utilise. Or, plus un composant est difficilement compréhensible, plus grande est la probabilité de faire des erreurs lors de son implémentation (et de son changement). Il favorise donc aussi la prédisposition aux fautes. De tels composants ont souvent une faible maintenabilité, testabilité et réutilisabilité.

Un fort couplage entrant indique qu'un composant/classe est fortement utilisé par le reste du système. Ce qui le rend critique du point de vue des fautes, car augmentant ainsi le risque de propagation par *ripple effect* dans les classes qui l'utilisent. Les fautes dans ces classes deviennent difficilement isolables et les modifications difficilement maitrisables vu le potentiel de répercussions sur le reste du système. De telles classes doivent subir des tests plus rigoureux.

La Granularité de mesure : Elle indique le niveau de détail des mesures à effectuer. Deux aspects sont à considérer :

- Le domaine de mesure : attribut, méthode, classe, module ou système
- Le décompte des connexions : selon le domaine de mesure choisi, différentes options sont possibles pour compter les connexions identifiées. Le tableau 1 donne les différentes options dans le cas ou le domaine de mesure est au niveau classe. CBO correspond à l'option D et RFC à l'option C.

Tableau 1:Option pour compter les connexions au niveau classe [Briand 99].

Option	Description
A	Compter les liaisons individuellement pour chaque méthode ou attribut de la classe puis faire la somme.
B	Compter le nombre d'éléments distincts de l'autre cote de la liaison pour chaque méthode ou attribut de la classe puis faire la somme.
C	Compter le nombre d'éléments distincts à la fin de liaisons qui partent de/ou vers un attribut ou méthode de la classe.
D	Compter, pour une classe donnée, le nombre de classes avec qui elle a au moins une liaison identifiée.

La Stabilité de la classe sujet : En faisant la distinction entre les classes dites instables, c'est-à-dire les classes propres au système en développement, et celles dites stables, c'est-à-dire les classes importées par le biais de librairies ou de réutilisation d'autres systèmes, on se retrouve face à différentes options pour mesurer le couplage (voir tableau 2).

Tableau 2 : Option pour la stabilité de la classe « sujet » [Briand 99].

Option #	Description
I	Prendre uniquement en compte les liaisons partant de/ou vers les classes instables.
II	Prendre uniquement en compte les liaisons partant de/ou vers les classes stables.
III	Prendre toutes les liaisons en compte indépendamment de la stabilité de la classe de l'autre côté de la liaison.
IV	Compter séparément les liaisons avec des classes stables et celles instables.

L'option I permet de faire abstraction des classes des librairies qui sont facilement réutilisables. Le couplage sortant mesuré peut ainsi donner une indication sur la réutilisabilité. L'option II, qui intègre les liens avec les classes librairies, peut être utile pour les analyses d'impact des changements, s'il arrive qu'une librairie a besoin d'être remplacée. L'option III ne considère pas la stabilité, donc énumère tous les services qui sont importés par une classe, ce qui donne une indication sur la compréhensibilité d'une classe vu qu'il est admis que l'augmentation du nombre de services externes utilisés par une classe accroit son incompréhensibilité. L'option IV compte séparément le couplage selon la stabilité de la classe sujet ce qui peut donner une indication sur la maintenabilité, si on tient compte de l'hypothèse selon laquelle la maintenabilité est influencée à la fois par les classes non stables et stables, mais à un degré moindre pour les secondes.

Le Couplage direct ou couplage indirect : Le couplage indirect est défini comme la fermeture transitive du couplage direct. Par exemple, si une classe C1 a un lien direct avec une classe C2 et que C2 a un lien direct avec une autre classe C3, alors on dit qu'on a un couplage direct entre C1 et C2 et un couplage indirect entre C1 et C3. Il est à noter que la plupart des métriques de couplage ne mesurent pas le couplage indirect dû à sa difficulté d'implémentation.

Les liens indirects peuvent être utiles pour estimer l'effort pour les activités d'exécution comme les tests et le débogage, ou pour évaluer l'impact des

modifications d'une classe sur le reste du système [Biran 99]. Ils peuvent aussi servir d'indicateurs pour la réutilisabilité : si une classe C est réutilisée par un autre système, il est nécessaire d'importer toutes les autres classes auxquelles C est liée de façon directe ou indirecte.

Les liens directs permettent d'avoir une idée sur la compréhensibilité, qui en effet nécessite la connaissance des fonctionnalités des services directement utilisés par la classe sans trop se soucier des services qu'utilisent ces dernières.

L'Héritage ou non : Trois problèmes (entre autres) sont liés à l'héritage lors de la mesure du couplage. D'abord, il faut décider si le couplage par le simple fait de liens de parenté entre deux classes va être pris en considération, ce qui mène aux options listées dans le tableau 3. Ensuite, il sera question de prise en charge du polymorphisme lors de l'énumération des liens d'invocation de méthodes. Et enfin, il faudra décider de la manière d'assigner les attributs/méthodes aux classes en cas de présence de lien d'héritage entre les classes concernées. Dans ce cas, il est possible de considérer, par exemple, qu'un attribut (ou une méthode) appartient uniquement (ou pas) à la classe ou il a été déclaré, ou qu'il appartient uniquement (ou pas) à la classe ou il a été implémenté.

Le polymorphisme ne parait pas vraiment important pour la compréhensibilité, car les différentes méthodes ont la même signature et sont censées fournir la même fonctionnalité. Mais, il peut être significatif pour la présence de fautes et la testabilité, vu que chaque méthode a sa propre implémentation et qu'une faute peut donc provenir de chacune d'entre elles, ou s'y propager.

Tableau 3: Option de comptage du couplage par héritage [Briand 99]

Option #	Description
I	Compter uniquement le couplage basé sur l'héritage.
II	Compter uniquement le couplage non basé sur l'héritage.
III	Compter séparément le couplage basé sur l'héritage et celui non basé sur l'héritage.
IV	Compter à la fois le couplage basé sur l'héritage et celui non basé sur l'héritage (sans distinction).

3.2.2 Caractéristiques et intérêt du couplage

Le couplage est un aspect important dans le concept OO d'autant plus qu'il est intrinsèquement lié à plusieurs autres mesures du logiciel comme la complexité et l'héritage et à plusieurs attributs externes de qualité comme la réutilisabilité, la robustesse, la modularité, etc. En effet, un fort couplage accroit la complexité du fait du grand nombre d'interactions entre les composants logiciels qui deviennent plus incompréhensibles et donc plus susceptibles de présenter des fautes, ce qui par conséquent affecte négativement leur maintenabilité. Les composants qui présentent un fort couplage sortant sont, d'une part, très sensibles aux changements et difficilement réutilisables du fait de leurs dépendances externes assez élevées. Ils sont, par ailleurs, assez critiques vis-à-vis de la présence de fautes, vu le nombre de composants externes qui intègrent sa définition. Quant aux composants présentant un fort couplage entrant, ils sont assez critiques vu le nombre de composants qui dépendent d'eux. Tout changement ou défaillance intrinsèque risque de se propager dans les autres composants qui dépendent d'eux (*ripple effet*). Par conséquent, un faible couplage est plus qu'une recommandation pour un logiciel de qualité.

Chapitre 4: Collecte des données

La validation empirique à partir des données collectées sur les systèmes logiciels constitue un des principaux supports de la recherche en génie logiciel.

En voulant effectuer de telles études, on se confronte très rapidement à la rareté des données disponibles. Il est possible d'effectuer de la fouille dans les dépôts/répertoires de logiciels pour collecter des données, en faisant intervenir un ensemble d'outils utilisés dans la CM (Configuration Management). Les données obtenues permettront, entre autres, de valider ou d'établir des théories à travers différentes méthodes d'études [Malhotra 16]. Cette option a été retenue dans notre étude. Nous allons procéder comme suit :

1. Calculer les métriques OO statiques à partir du code source.
2. Recenser toutes les fautes signalées après la mise en production (fautes *post-release*) en fouillant dans les modifications qui ont été effectuées au niveau du code source (à l'aide des SCV : Systèmes de Contrôle de Versions).
3. Recouper les données des étapes précédentes (1 et 2) avec les informations supplémentaires sur les fautes (sévérité) obtenues à travers les rapporteurs de bogue (BTS/ITS : *Bug/Issue Tracking Systems*).
4. Effectuer des prétraitements sur les données résultantes afin de les organiser et de les représenter selon les besoins de nos expérimentations.

Dans cette partie, nous allons présenter la méthode utilisée pour extraire les données. Pour cela, nous allons expliquer la structure et l'organisation de la journalisation des changements (*logs*) et des rapports de bogues. Nous allons aussi présenter certains outils de gestion de configuration conçus à cet effet. Nous allons décrire une technique de collecte de données sur les fautes basées sur ces outils. Avant de clore ce chapitre, nous allons détailler les prétraitements effectués sur les données récupérées pour enfin présenter leur description finale.

4.1 Gestion de configuration

En génie logiciel, il existe des outils/applications utilisés pour contrôler et suivre l'évolution du logiciel. Parmi ces derniers, on retrouve une panoplie d'outils pour répertorier, organiser, contrôler, gérer et suivre les changements/modifications à apporter au code source, qui s'inscrivent dans le cadre de la gestion de configuration.

4.1.1 Système de gestion de la configuration (SGC)

Le SGC a pour fonction principale de gérer des changements portant sur tous les livrables, produits durant le cycle de vie du logiciel, comme la spécification des besoins, le document de conception du logiciel, le code source, les manuels d'utilisateurs, etc. [Bersoff 90, Babich 86]. La gestion de la configuration (Configuration Management : CM) s'organise autour de quatre activités principales :

1. **Identification** : consiste à designer de façon distincte tous les artéfacts/livrables appartenant à une configuration en leur attribuant un identifiant unique, et en leur associant toutes les caractéristiques/propriétés liées à leur description.
2. **Contrôle** : s'assure de la maîtrise de l'évolution de la configuration. Les demandes de changement sont étudiées et leur mise en œuvre planifiée, analysée et suivie en cas d'approbation par le chef de projet. Le tableau 4 montre un exemple de formulaire type d'une demande de changement (*Change Request* : CR)
3. **Administration** : l'administration de la configuration assure une traçabilité de toutes les activités, les modifications/changements inclus, qui affectent la configuration du logiciel, ou le logiciel lui-même.
4. **Audit** : vérification de l'intégrité du système d'information et l'exactitude des informations sur les éléments de la configuration.

Pour mener à bien ces activités, un ensemble d'outils tels que les référentiels de contrôle de source (VCS) et les référentiels de bogues (BTS) existent. Il est

possible de collecter beaucoup d'informations sur la configuration d'un produit logiciel en interrogeant directement ces outils.

Tableau 4 : Formulaire de demande de changement [Malhotra 16].

Change Request Form				
Change Request ID				
Type of Change Request	☐ Enhancement	☐ Defect Fixing	☐ Other (Specify)	
Project				
Requested By	*Project team member name*			
Brief Description of the Change Request	*Description of the change being requested*			
Date Submitted				
Date Required				
Priority	☐ Low	☐ Medium	☐ High	☐ Mandatory
Severity	☐ Trivial	☐ Moderate	☐ Serious	☐ Critical
Reason for Change	*Description of why the change is being requested*			
Estimated Cost of Change	*Estimates for the cost of incurring the change*			
Other Artifacts Impacted	*List other artifacts affected by this change*			
Signature				

4.1.1.1 Les Systèmes de contrôle de versions

Le système de contrôle de versions ou VCS (Version Control System) en anglais, est un outil logiciel qui permet de suivre et d'enregistrer les changements/modifications survenus sur chaque artéfact d'un système logiciel. Cela consiste, donc, à maintenir chaque version de chaque fichier du code source.

La terminologie suivante est liée aux VCS [Ball 97] :

- **Numéro de révision** : Les différentes versions d'un artéfact sont distinctement identifiées par un numéro unique appelé numéro de révision.
- **Numéro de sortie** (release) : Chaque version du produit sera identifiée pas un numéro unique de sortie.
- **Tronc** : Il s'agit de la version de base (aussi appelée *master*) sur laquelle des changements pourront être apportés par la suite.
- **Tag** : Toutes les révisions d'une *release* sont regroupées sous un nom symbolique qui est le tag. Il indique le numéro de release.
- **Branche** : Il s'agit d'une sorte de ramification du projet de base qui adopte une ligne de développement différente (par exemple l'ajout de nouvelles

fonctionnalités ou de nouveaux modules). Une branche évolue donc parallèlement au tronc et aux autres branches. Chaque branche est aussi identifiée par un numéro.

- **Head** : Au niveau produit, le *head* (aussi appelé *tip*) fait référence à la version en cours de développement. Plus exactement à la version la plus récente parmi tous les artéfacts d'une branche. Chaque branche a ainsi son propre *head.*
- **Patch/diff** : Il s'agit d'un changement apporté à un fichier ou un ensemble de changements répartis sur plusieurs fichiers (ajout, suppression, modification de code).
- **Commit (soumission)** : C'est la validation effective d'un changement, qui permet de l'enregistrer dans la configuration. Un *commit* s'accompagne d'un message descriptif pouvant contenir des informations pertinentes sur la nature de la modification effectuée.

4.1.1.2 Les systèmes de suivi de bogue (ITS/BTS)

Un système de suivi de bogues (ITS/BTS) fait le suivi et la maintenance de divers problèmes (bogues/défauts) inhérents à un produit logiciel tout au long de son cycle de vie. Les ITS permettent de rapporter les problèmes liés à un projet logiciel, de faire leur évaluation et leur suivi. Ils fournissent, ainsi, des informations pertinentes vis-à-vis de ces problèmes. On peut, entre autres, y retrouver la date du rapport du problème (bogue/défaut), sa sévérité, la description du comportement qu'il engendre au sein du programme ou module, la façon détaillée de le reproduire, les informations sur la personne qui l'a rapporté et sur le ou les développeurs responsables de sa résolution. Toutes ces informations sont accessibles depuis une base de données à travers des outils (interfaces) graphiques, le plus souvent web. On devrait typiquement avoir les propriétés suivantes pour un bogue dans n'importe quel ITS :

- Rapporteur : La personne ayant rapporté le problème
- Assigné : Le(s) développeur(s) en charge de sa résolution

- Composant : le(s) composant(s) du produit logiciel concerné(s) par le problème
- Type : le type de problème
- Description : une description du problème
- Version : la/les version(s) du produit logiciel concernée(s) par le problème
- Liste de détails indiquant comment reproduire le bogue constaté
- Sévérité : le niveau de gravité du problème
- Priorité : le niveau d'urgence de résolution du problème
- État : L'état en cours dans le processus de résolution du problème.

Le fonctionnement d'un ITS repose sur le concept de cycle de vie d'un bogue (Figure 4). Pouvant conduire à un changement (correctif) dans le produit logiciel, le suivi d'un bogue se fait en conformité avec le processus de demande de changement (*Change Request Workflow : CRW*). Le bogue passe donc par différentes étapes qui, une fois regroupées, constitueront son cycle de vie. À chacune de ces étapes, on attribuera un état au bogue qui indiquera le processus de traitement. Le cycle de vie d'un bogue démarre avec l'état « opened » une fois soumis par le rapporteur. Après une première vérification de sa reproductibilité, son état va être modifié selon les décisions suivantes :

- Le problème ne sera pas corrigé. Différentes raisons peuvent conduire à cette conclusion. Par exemple, la différence entre le coût nécessaire pour corriger le bogue et le bénéfice obtenu en retour peut être jugée trop grande, et dans ce cas son état passera à « differed ».
- Le problème va être traité et un calendrier est mis en place à cet effet. Ce choix conduira aux états suivants :
    - « Fixed » indiquant que des changements ont été effectués dans le code (*patch*) pour résoudre le problème.
    - « Duplicated » indiquant qu'il s'agit d'une duplication d'un problème déjà signalé.

- « Resolved » indiquant que le problème a été pris en compte, mais qu'aucune modification dans le code n'aura lieu.

Les ITS sont le plus souvent intégrés à d'autres systèmes de gestion de projet comme les VCS. On peut citer comme exemples JIRA et BUGZILLA, que l'on présentera plus en détail dans la suite. Notons que l'implémentation de ces concepts présentés ci-haut peut différer légèrement d'un ITS à un autre.

BUGZILLA

Bugzilla est l'un des plus célèbres BTS open source. Disponible sous Windows, Mac et Linux, il offre une interface web dans une architecture trois tiers. Il est utilisé dans le cadre de plusieurs projets d'envergure dont Mozilla, Eclipse, Facebook, Yahoo, Apache Ant et certaines distributions Linux. Il a été développé par *Mozilla Foundation* et est distribué sous tri-licence MPL/LGPL/GPL. Il est accessible via http et peut être intégré à un VCS (SVN /CVS).

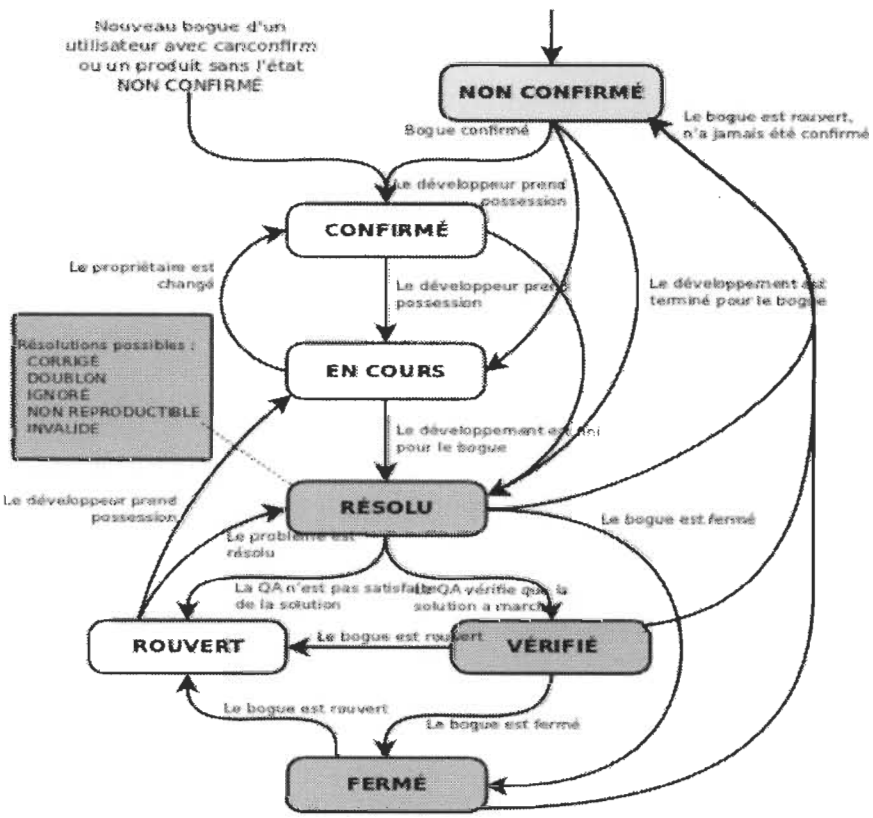


Figure 4: Workflow Bugzilla (http://www.bugzilla.org).

JIRA

Jira est à la fois un I/BTS et un système de gestion de projets écrit en JAVA. Il est sous licence commerciale, mais est néanmoins totalement gratuit pour les projets open source. Il est surtout connu pour sa forte communauté d'utilisateurs et de développeurs, sa documentation bien fournie et sa grande facilité d'intégration avec d'autres outils. Il offre une possibilité d'accès à distance via REST, SOAP, et XML-RPC. Plusieurs grands projets utilisent JIRA.

Le cycle de vie par défaut d'un bogue dans JIRA reste très basique. Le choix est laissé à l'administrateur de JIRA de le personnaliser pour un projet et un bogue donné. La présentation des données est basée sur des filtres hautement paramétrables offrant la possibilité d'effectuer des recherches exhaustives et très ciblées. La migration de Bugzilla vers JIRA est conseillée, et JIRA met à disposition des outils conçus à cet effet.

4.2 *BugInfo* : outil d'extraction d'information sur les bogues

Buginfo[2] est un outil open source permettant de faire de la fouille de données sur les bogues dans les répertoires logiciels (SVN et CVS). Il permet d'identifier les classes ayant été affectées par des bogues.

Il fonctionne de façon relativement simple. L'idée est de se connecter à un VCS et d'analyser la description des *commits* pour identifier les corrections de bogues et d'extraire les informations sur les modifications du code (*patchs)* associées à ces dernières. On peut distinguer deux étapes pour chaque *commit* examiné :

1. **Reconnaissance du bogue** : Cela est fait par analyse textuelle dans les descriptifs des soumissions (*commits*) en utilisant une expression régulière (E) donnée en paramètre. Ce motif couvre des termes particuliers fréquemment utilisés pour décrire la correction d'un bogue. Ces termes dépendent du nom du projet et/ou du ITS qu'il utilise. Par exemple, pour le projet apache ANT qui utilise Bugzilla, les *commits* correspondant à des corrections de bogues vont contenir le mot « bugfix » ou « bugzilla » (sans

[2] https://kenai.com/projects/buginfo

respect de la casse). Dans ce cas, notre expression régulière ressemblera à :

E = .*(([bB][uU][gG][fF][iI][xX])|([bB][uU][gG][zZ][iI][lL][lL][aA])).*

2. **Extraction d'information** : Une fois qu'un commit a été identifié comme étant la correction d'un bogue, les informations additionnelles du patch vont être utilisées pour connaître les classes concernées par le bogue. Pour ce faire, trois paramètres permettant d'identifier le nom complet d'un fichier source vont être utilisés.
    - **L'extension du fichier** qui permet d'identifier les fichiers correspondants aux classes, par exemple .java pour le code écrit en JAVA
    - **La racine** du dossier principal
    - **Le séparateur** de dossier (/ par défaut).

    Ces trois paramètres vont servir à identifier, parmi tous les fichiers ayant subi des modifications pour la correction du bogue, ceux qui correspondent à des fichiers de code source. Les fichiers ainsi retenus vont être considérés comme contenant des classes défectueuses et leur compteur de bogues sera incrémenté de 1. Cela suppose qu'une classe est considérée comme étant fautive si elle a subi des modifications dans le cadre de la correction d'un bogue.

L'outil prend tous ses paramètres via un fichier de configuration (voir Figure 5). Les paramètres vont permettre, entre autres, de : limiter la fouille sur une version spécifique (en utilisant un intervalle de numéro de révision), définir le motif des chemins d'accès aux fichiers de code source (dossier racine, chemin, extension), ou choisir l'option de stockage des résultats dans un fichier ou une base de données.
Buginfo présente des limitations. En regard de nos objectifs, nous avons énuméré celles qui nous concernent :

- L'identification des bogues lors de la *phase 1* est un peu approximative. L'expression régulière ne permet pas à coup sûr d'identifier les bogues.

- Le risque de duplication dans le décompte des bogues. L'outil ne prend pas en considération, pour compter les fautes d'une classe, le fait qu'on peut avoir un bogue dont la résolution s'est faite sur plusieurs *commits*. Chaque commit retenu à la *phase 1* est considéré comme un nouveau bogue (différent), ce qui n'est pas forcément le cas.
- La non-intégration des ITS. Aucune information concernant la nature des bogues. Il est impossible, par exemple, de connaître la sévérité des bogues.

Pour nos propres besoins, nous allons étendre Buginfo (BuginfoV2) pour pallier ou atténuer les limitations listées ci-dessus.

```
#Repository type; svn or cvs
repotype = svn

#URL of the svn or cvs repository
repourl = http://svn.apache.org/repos/asf/tomcat/trunk

#Specifies the revision where the analyze should start. It's a number for the svn
#repositories and an date (2002/07/07 09:45:00) for the cvs repositories.
repostartrev = 389418

#Specifies the revision where the analyze should end. It's a number for the svn
#repositories and an date (2002/07/07 09:45:00) for the cvs repositories.
#When no repoendrev is given the repositroy will be analyzed till HEAD revision.
#repoendrev = 2009/08/08 09:12:00

#Svn only! Login to the svn repository. In cvs the login will be taken from
#working copy settings.
reposvnuser = login

#Svn only! Password to the svn repository. In cvs the password will be taken
#from working copy settings.
reposvnpass = pass

#Regular expression. When a comment in a revision fits to this regular expresion,
#the revision is interpretted as a bugfix.
repobugfixregex = .*(([bB][uU][gG][fF][iI][xX])|([bB][uU][gG][zZ][iI][lL][lL][aA])).*

#Regular expression. Source code prefix.
reposrcpathprefixregex = .*/java/

#Regular expression. Source code postfix
reposrcpathpostfixregex = .java

#Path separator. Default is '/'
repofileseparator =

#Cvs only! Path to working copy.
repoworkingcopy = C:\\Programme\\eclipse\\workspace-Freigabe\\workspace-zvk\\common-zvk

#Decides whether the number of modifications (revisions) will be read from
#repository and soterd in the database or a csv file.
readnumberofmodifications = true

#Decides whether the commiters will be read from repository and soterd in
#the database or a csv file.
```

Figure 5: Exemple de fichier de paramètre Buginfo.

4.2.1 BuginfoV2

En partant de Buginfo, nous allons apporter des améliorations, en effectuant des modifications dans le code source du logiciel, pour produire BuginfoV2. L'outil sera connecté aux ITS JIRA et BUGZILLA.

Nous allons ainsi pouvoir :

- Augmenter la précision de l'identification des *commits* correspondants à des corrections de bogue (phase 1). Nous utiliserons l'identifiant unique des bogues contenu dans l'ITS pour réduire le champ de couverture de l'expression régulière (E), qui était assez vaste sur Buginfo.
- Réduire le risque de duplication dans le décompte des fautes pour une classe. En effet, chaque classe fautive est associée à une liste contenant les identifiants des bogues trouvés. Avant l'ajout d'un nouveau bogue, nous vérifions sa présence préalable dans la liste. Cette approche nous permet d'éviter la duplication (phase 2).
- Obtenir toutes les informations sur la nature des bogues plus particulièrement la sévérité. Nous allons utiliser un compteur de bogues par niveau de sévérité. À chaque nouvelle faute identifiée et associée à une classe, nous incrémentons uniquement le compteur correspondant à la sévérité de la faute.

D'autres modifications ont aussi été apportées à l'outil pour améliorer la flexibilité de Buginfo. Nous pourrons, par exemple, choisir au niveau des paramètres d'utiliser un fichier local comme ITS en lieu et place d'une connexion distante. En effet, il se trouve que les interfaces utilisateurs (web) de JIRA et Bugzilla permettent de faire des recherches exhaustives sur les bogues, et d'exporter les résultats de la recherche sous différents formats de fichier. Nous avons mis à profit cette fonctionnalité dans la nouvelle version BuginfoV2. Ainsi, une partie de la recherche pourra être effectuée manuellement par l'utilisateur et les résultats seront soumis en entrée à BuginfoV2. Cela offre, par ailleurs, la possibilité d'exploiter les informations d'un ITS même si celui-ci n'offre pas ou restreint l'accès via des

appels distants RPC. Pour la génération du rapport final, la seule modification apportée est la possibilité d'intégrer les métriques des classes qui sont données en entrée dans un fichier CSV. Cette amélioration a aussi permis d'augmenter la précision de l'association entre classe et nom de fichier (phase 2), en ne retenant que les noms de fichiers correspondants à des noms de classe contenus dans le fichier des métriques de classe (dans le cas où celui-ci est donné en entrée).
Il faudra noter que ces modifications ont été volontairement limitées au module de recherche pour les répertoires SVN, car étant celui qui concerne notre étude. Elles seront probablement élargies à l'outil CVS au besoin.

4.3 Les systèmes étudiés

Nous présentons, dans cette partie, une brève description des systèmes sur lesquels nous allons collecter les données. Il s'agit de projets open source écrits en JAVA.
Apache Ant (ANT) est un outil de la fondation Apache conçu pour aider à l'automatisation de certaines tâches lors de la phase de développement dans le processus logiciel. C'est un outil très populaire.
Apache ActiveMQ (AMQ) est un outil de gestion de queue de messages (*message broker*) très populaire.
Apache Ivy (IVY) est un outil de gestion de dépendances. Il est connu pour sa simplicité et sa flexibilité.
LUCENE est un outil de recherche et d'indexation de texte.
Apache Camel (CAMEL) est un puissant Framework d'intégration d'applications d'entreprise.

4.4 Sources de données (Procédure d'extraction des données)

4.4.1 Calcul des métriques

Les métriques utilisées pour cette étude ont été calculées avec CKJM[3] (*Chidamber and Kemerer JAVA Metrics*), un outil open source conçu pour calculer 19 métriques OO à partir du bytecode (fichier .class) d'applications JAVA. Il s'utilise

[3] http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/index.html

en ligne de commande, et peut aussi être intégré à d'autres outils pour générer automatiquement des rapports. Il calcule une large variété de métriques (couplage, héritage, abstraction, taille… etc.) en se basant sur les définitions des suites CK et QMOOD.

4.4.2 Collecte des données sur les fautes

Pour faire des études empiriques en génie logiciel, on distingue principalement deux approches quant aux sources des données utilisées (publiques, partiellement publiques) [Radjenović 13].

La première approche est de partir de données (publiques) proposées par des organismes reconnus (et disponibles au téléchargement public). Il existe des dépôts web dédiés au regroupement et à la mise à disposition d'ensemble de données pour la communauté des chercheurs en génie logiciel. Cette approche est adoptée par plusieurs études d'une part par souci de validité, les données étant connues et déjà utilisées par un large public, et d'autre part, par la présentation de ces dernières qui sont quasi prêtes à l'usage.

La deuxième approche consiste à faire de la fouille (*mining*) à travers les systèmes de gestion de la configuration d'organismes de développement logiciel pour se constituer soi-même une base de données ou pour compléter des données obtenues avec la première approche. Il y a plusieurs difficultés liées à cette approche dont la principale est l'accès au code source.

En ce qui nous concerne, les deux approches vont être nécessaires. Nous allons réutiliser des données disponibles sur des répertoires web de données (approche 1), tout en fouillant dans les ITS et CSV pour compléter ces données. La réutilisation va concerner uniquement les métriques logicielles calculées avec CKJM et disponibles sur PROMISE (sauf pour ActiveMQ). Quant à la collecte des informations sur les bogues, nous allons utiliser l'outil buginfoV2 pour identifier les classes fautives, la sévérité des bogues ainsi que leur nombre. Ce qui nous donne au final, pour chaque projet, un tableau dont les lignes représentent les classes et les colonnes. Les 19 métriques plus les 5 niveaux de sévérité. La figure 6 donne une illustration graphique de ce procédé.

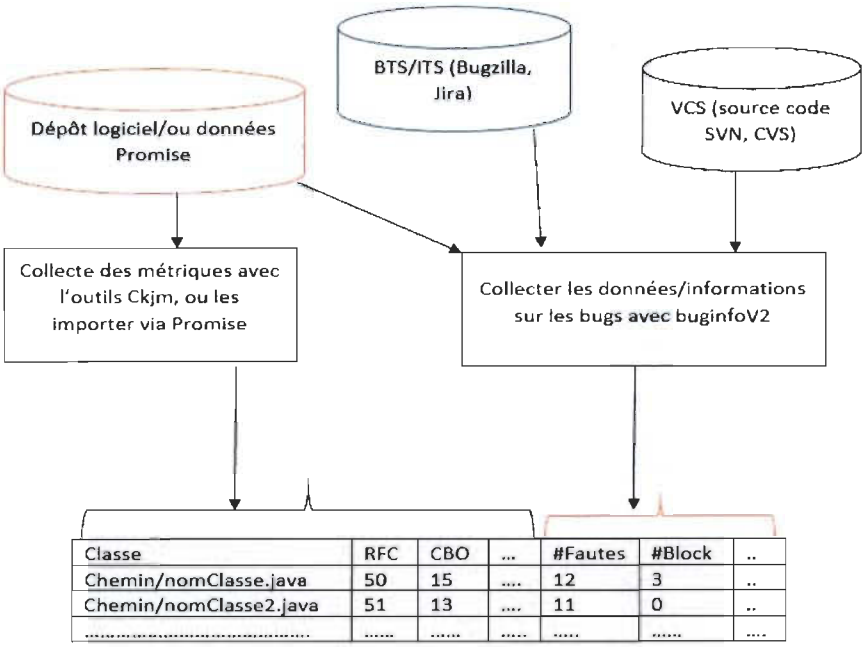
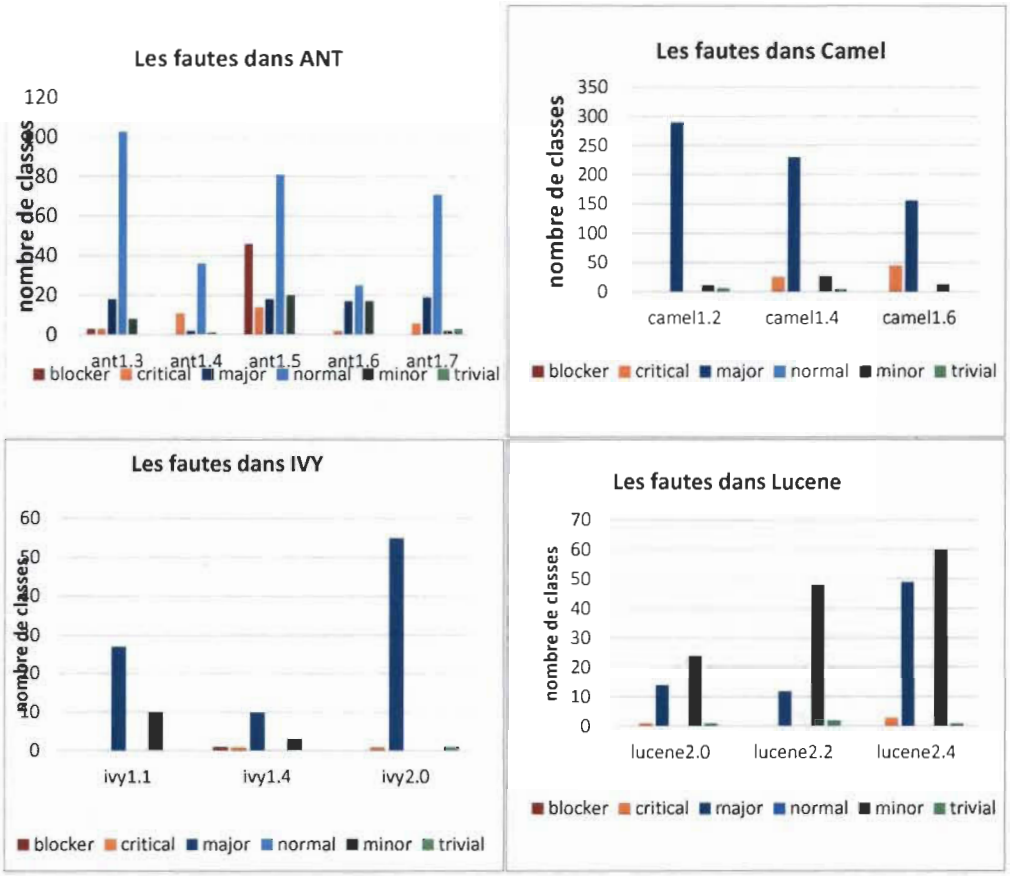


Figure 6: Procédure de collecte des données sur les fautes.

4.5 Présentation des données sur les fautes

Les figures 7 et 8 présentent une description des données récoltées sur les fautes. On constate que les fautes de niveau de sévérité majeur (major) et normal sont les plus fréquentes et que celles de niveau de sévérité bloquant (blocker) sont très peu fréquentes, ce qui semble assez cohérent. Les systèmes IVY et LUCENE ont les taux de fautes de niveau de sévérité bloquant ou critique (critical) les plus bas. Nous avons donc décidé de les écarter dans nos expérimentations. On note que certaines versions sont plus touchées par ces fautes que d'autres. Cela va orienter nos choix de systèmes/versions à étudier et des considérations à faire pour la représentation finale de la sévérité.



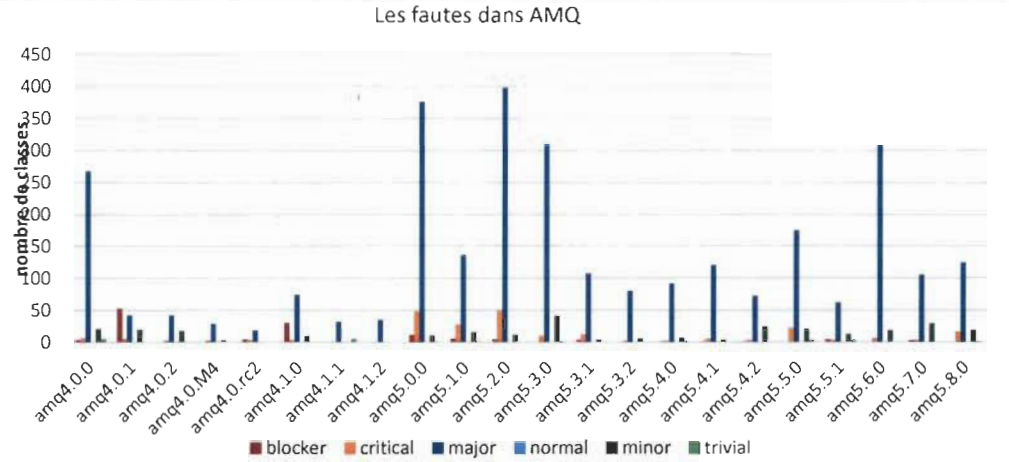


Figure 7: Les fautes collectées : CAMEL, IVY, LUCENE, AMQ, ANT.

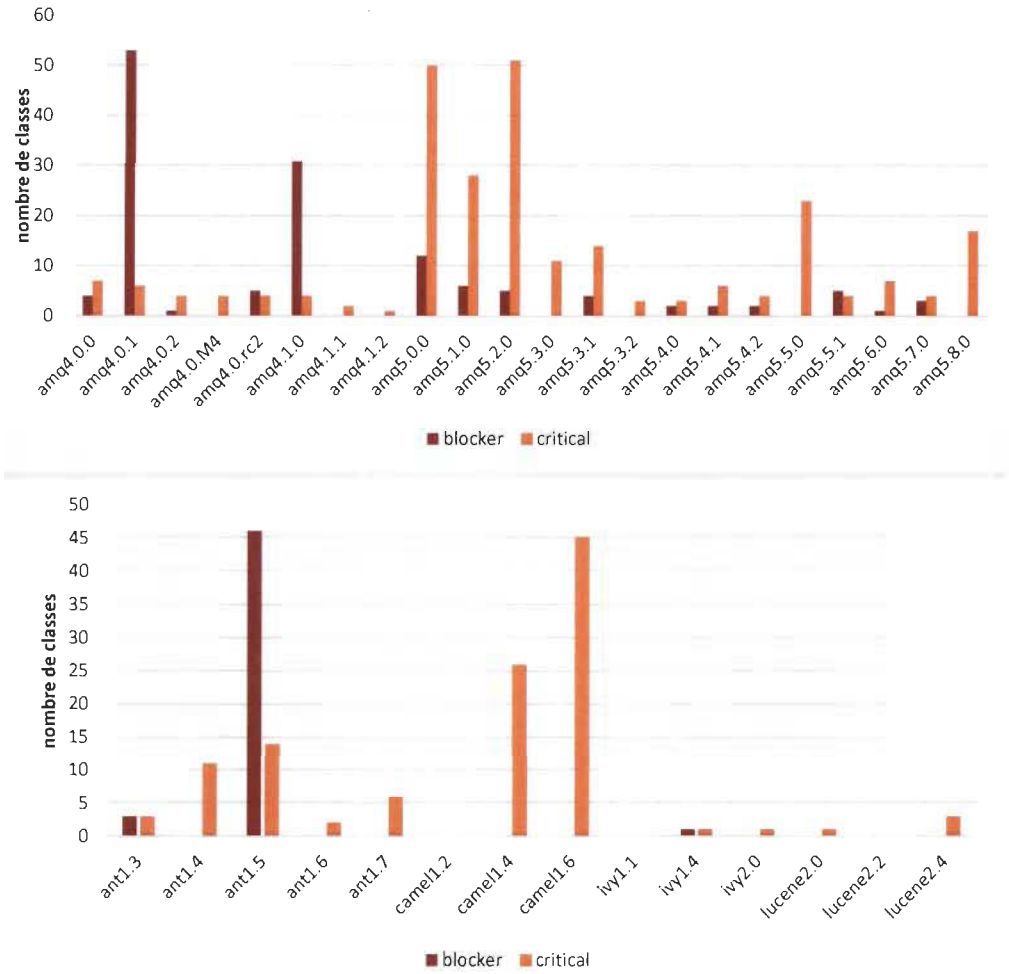


Figure 8 : Classes contenant des fautes critiques.

4.6 Prétraitement et organisation finale des données

Étant donné qu'une classe peut contenir des fautes de différents niveaux de sévérité, nous allons explorer différents types de regroupements en fonction du nombre de fautes permettant d'exprimer cette information sous forme d'une variable.

- Binaire : La sévérité sera évaluée par une variable binaire. Cela peut se faire (par exemple) par le regroupement des niveaux en 2 catégories distinctes. Nous pourrons distinguer deux catégories de classes : celles qui

ont une faute (au moins une) dans le niveau retenu, et les autres. Ce qui nous donne une variable binaire, avec 1 pour la première catégorie de classes, et 0 pour la seconde.

Cette transformation va permettre d'utiliser des méthodes d'analyse qui nécessitent des variables binaires. La fusion de certains niveaux pourra pallier le manque d'observations pour certains niveaux de sévérité. Nous avons choisi pour notre étude de considérer qu'une classe présente une faute critique si et seulement si elle est affectée par au moins une faute de sévérité de type *bloquant* ou *critique* (fusion des niveaux de sévérité blocker et critical).

- Linéaire (Ranking) : Ce regroupement a pour but d'ordonner les classes selon les niveaux de gravité des fautes trouvées. Chaque classe aura un indice indiquant son niveau par rapport aux autres classes. Cette méthode a été présentée par [Oyetoyan 13] et se base sur des transformations matricielles. L'intérêt de cette transformation est d'avoir la sévérité sur une seule dimension numérique et linéaire. Ce qui va permettre d'utiliser des méthodes d'analyse nécessitant des variables linéaires. Il s'agit d'une autre forme de classification de la sévérité.

Étant donné que notre étude porte sur les fautes post-release, nous avons décidé, en calculant les fautes sévères associées à une classe d'une version i, de considérer celles détectées dans les versions supérieures j (j>i). Cette approche permet non seulement d'augmenter la quantité de données sur les fautes sévères, mais aussi (et surtout) de prendre en compte l'évolution du produit logiciel dans la construction des modèles de prédiction. Pour AMQ, nous avons choisi d'étudier la version 5.0, et nous allons donc inclure les fautes sévères des versions 5.1 à 5.8. Pour ANT, l'étude portera sur la version 1.3 et nous avons inclus les fautes sévères des versions 1.4 à 1.7 ; et pour CAMEL nous étudierons la version 1.4 en incluant les fautes sévères de la version 1.6.

Chapitre 5: Étude expérimentale

5.1 Méthodologie d'analyse

5.1.1 Concepts théoriques

Nous allons, dans cette partie, présenter de façon brève certains des concepts utilisés pour mener nos expérimentations. Nous emploierons le terme modèle pour faire référence à l'implémentation de la technique mathématique permettant de modéliser les relations entre la variable dépendante et la/les variable(s) indépendante(s).

5.1.1.1 Analyse de corrélation

L'analyse de corrélation est une technique statistique utilisée pour mesurer le sens et l'intensité de liaisons de nature linéaire entre deux variables. Cette analyse est effectuée en calculant le coefficient de corrélation linéaire *r* dont les valeurs varient entre -1 et 1. La valeur absolue de r indique la force de la liaison, et son signe le sens de la liaison. La valeur nulle indique l'absence de lien. Si la liaison est forte, une valeur positive de *r* indique que deux variables varient dans le même sens, et une valeur négative indique qu'elles varient dans le sens opposé. L'analyse de la corrélation nous permettra ainsi de vérifier si deux métriques sont linéairement interdépendantes, le sens et la force de cette liaison.

5.1.1.2 Construction/extraction de caractéristiques/variable

L'analyse en composantes principales (ACP)

L'ACP est une technique standard d'analyse de données qui est très utile pour examiner l'interdépendance entre des variables ainsi que les dimensions cachées qu'elles capturent. C'est une technique d'analyse factorielle qui permet de représenter les données dans un nouvel espace composé de dimensions, dites axes principaux (ou composantes principales), totalement de-corrélées. Le nombre d'axes principaux est égal au nombre de variables de départ, et l'importance relative de chaque axe est exprimée à travers la valeur propre qui lui est associée. Chaque axe A_i $(i = 1 \ldots n)$ constitue une combinaison linéaire des variables X_j $(j = 1 \ldots n)$

$$A_{i,}=S_{i1}X_1 + S_{i2}X_2 + \ldots\ldots S_{in}X_n$$

La matrice S_{ij} (saturations) représente les corrélations entre les variables et les axes. Elle donne des indications sur la signification des axes, qui peuvent en retour, donner des indications sur les points communs (ou pas) entre les variables.

Pour faciliter l'identification et l'interprétation des axes d'une ACP, il est d'usage d'effectuer une rotation des axes, tout en maintenant leur orthogonalité. Plusieurs techniques de rotation existent dont VARIMAX qui est très utilisée dans la littérature [Malhotra 16]. Cette dernière maximise la somme des variances de la matrice de saturation.

Dans notre étude, nous allons utiliser l'ACP d'une part comme technique d'analyse de l'information véhiculée par les différentes métriques de couplage, et d'autre part comme technique d'extraction/construction de caractéristiques (en utilisant les axes comme variables dépendantes).

Le Clustering (regroupement de variables)

Le but de cette méthode est de réduire un ensemble de variables $X = \{X_1, X_2, \ldots X_n\}$ en un ensemble de groupes (*cluster*) $C = \{C_1, C_2, \ldots C_k\}$ avec k<n. Les clusters sont construits de façon à ce que les variables qu'ils résument soient les plus proches possible.

Dans notre cas, nous allons utiliser les techniques VARHCA (*Hierarchical Cluster Analysis*) et VARKMeans (partitionnement en K-moyennes ou *K-Means*). Il s'agit des implémentations (dans l'outil Tanagra[4]) de techniques de regroupement de variables, basées sur le principe des composantes latentes [Vigneau 03]. Le premier adopte une approche de classification ascendante hiérarchique, tandis que pour le second c'est celle des nuées dynamiques.

Dans notre étude, nous allons utiliser la classification de variables d'une part pour explorer les relations entre les différentes métriques de couplage (avec VARHCA et VARKMeans), et d'autre part pour représenter le couplage dans d'autres dimensions (avec VARHCA). Dans ce dernier cas, les clusters seront les variables dépendantes.

[4] http://eric.univ-lyon2.fr/~ricco/tanagra/

5.1.1.3 Sélection de caractéristiques/ variables

Les techniques de réduction de caractéristiques sont parfois utilisées dans la construction des modèles de prédiction dans le but de réduire la redondance de l'information et augmenter l'efficience des modèles obtenus. En apprentissage ou classification, elle permet de cibler, dans un groupe de variables explicatives, celles qui sont les plus pertinentes pour expliquer les observations de la variable dépendante dans le but de :

- Réduire le nombre de variables
- Améliorer la connaissance des liens de causalité entre les variables explicatives et la variable à expliquer
- Améliorer la qualité de la prédiction (du moins son efficience) : minimiser le nombre de variables explicatives sans trop dégrader la qualité du modèle.

STEPDISC (Stepwise Discriminant Analysis) est une technique de sélection de variables reposant sur le critère du LAMBDA de WILKS, qui est la transposition multidimensionnelle de la décomposition de la variance. Deux stratégies itératives sont possibles :

- L'approche FORWARD qui consiste à ajouter (pour chaque itération) à l'ensemble des variables sélectionnées (au départ vide), la variable (non sélectionnée) qui donne la meilleure amélioration statistiquement significative du LAMDA.
- L'approche BACKWARD qui consiste à inclure au départ toutes les variables dans la sélection, puis d'en retirer à chaque itération celle dont le retrait entrainera la plus faible dégradation (non statistiquement significative selon un seuil fixé) du LAMBDA.

5.1.1.4 Régression logistique

La régression logistique (RL) est une méthode (statistique) standard de construction de modèle de prédiction. Elle est très utilisée pour l'assurance qualité logicielle basée sur les métriques, plus particulièrement la prédiction de la

prédisposition des classes aux fautes. Le modèle de régression logistique est basé sur l'équation suivante :

$$P(X_1, X_2, \ldots\ldots X_n) = \frac{e^{(C_0 + C_1 X_1 + \ldots\ldots C_n X_n)}}{1 + e^{(C_0 + C_1 X_1 + \ldots\ldots C_n X_n)}}$$

Ou les X_i sont les variables indépendantes (les métriques) représentées par le vecteur X, les C_i sont les coefficients (estimés) correspondants aux Xi, et P la probabilité de survenance de la modalité de référence de la variable dépendante pour chaque observation.

Dans le cas particulier d'une seule variable explicative (n=1), on parle de régression logistique univariée (RLU), sinon (n>1), la régression logistique est dite multivariée (RLM). Dans notre cas, les deux types de RL seront utilisés pour étudier les liens entre les métriques de couplage (variables dépendantes) et la prédisposition des classes aux fautes sévères. La RLU permettra d'examiner l'influence individuelle de chaque métrique sur la présence/absence de faute(s) sévère(s) dans les classes, alors que la RLM permettra d'analyser l'efficacité de l'effet combiné de celles-ci dans la prédiction de la prédisposition aux fautes sévères (capacité à expliquer la variable indépendante).

5.1.1.5 Modèles d'apprentissage automatique (supervisé)

Pour évaluer la capacité du couplage à prédire la prédisposition aux fautes sévères, nous avons utilisé, en plus de la régression logistique, six techniques d'apprentissage automatique : Naive Bayes (NB) [Russell 95], Perceptron Neural Network (NNP) [Haykin 04, Sherrod 03], Radial Basis Fonction (RBF) [Sherrod 03, Haykin 04], Support Vector Machine (SVM) [Platt 98, Cortes 95], Random Forest (RF) [Breiman 01] et Decision tree (DT) [Quinlan 93]. Nous avons utilisé l'outil Tanagra pour construire ces modèles.

5.1.2 Protocole expérimental

L'étude peut être subdivisée en trois parties :

La première partie consistera à analyser des métriques de couplage (différence, proximité, représentation, etc.). Pour cela nous allons :

- Analyser les statistiques descriptives qui donnent une vue globale des caractéristiques des systèmes et des métriques étudiées.
- Utiliser l'analyse de corrélation, l'ACP et le regroupement pour étudier les liens entre les métriques de couplage, identifier les différentes catégories de couplage capturées à travers celles-ci et identifier les métriques les plus représentatives de ces catégories.

La deuxième partie traite des liens entre le couplage et la présence de fautes critiques dans les classes. L'investigation sera faite par le biais de la régression logistique (RLU puis RLM) et de la sélection de variables (STEPDISC).

La troisième et dernière partie consistera à évaluer la capacité des métriques de couplage (à travers des modèles de prédiction) à identifier les classes présentant des fautes critiques dans un contexte empirique. Les données des trois systèmes sous étude seront fusionnées lors de cette expérimentation. Les variables dépendantes vont être représentées sous trois formats (voir figure 9) :

1. Données originales avec les métriques de couplage.
2. Données (résultantes de techniques de construction de caractéristique ACP et VARHCA) représentées par les axes principaux après transformation ACP.
3. Données représentées par des clusters après transformation VARHCA.

Pour chacune de ces représentations, nous allons procéder à l'expérimentation dont les étapes sont les suivantes :

1. Diviser les données en deux échantillons de même taille (apprentissage et test).
2. Appliquer la technique de sélection d'attributs à l'échantillon d'apprentissage.
3. Construire les modèles avec les attributs de l'étape 2 sur les données d'apprentissage.
4. Appliquer les modèles obtenus à l'échantillon de test.
5. Évaluer les performances des modèles.

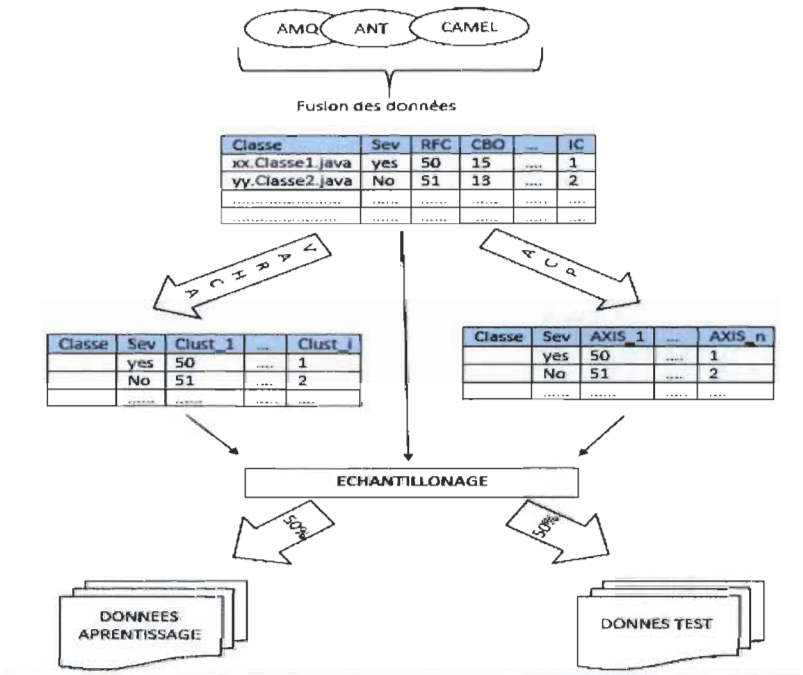


Figure 9 : Processus de représentation et d'échantillonnage des données.

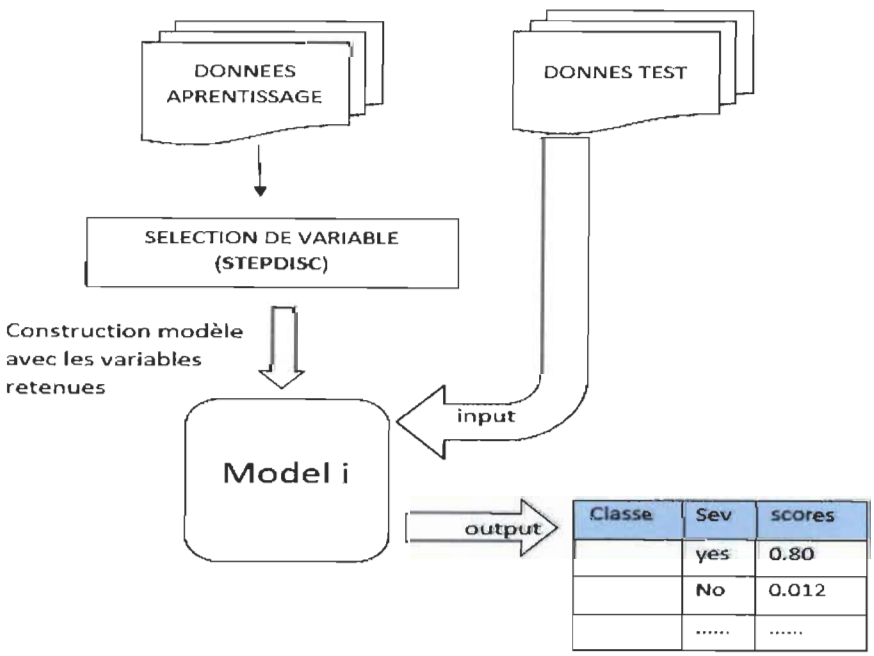


Figure 10 : Procédure de construction des modèles.

5.2 Présentation des résultats et discussion

5.2.1 Description des données

5.2.1.1 Statistiques descriptives

Les statistiques descriptives donnent un point de vue résumé de nos observations en s'appuyant sur des indicateurs tels que la moyenne, médiane, minimum, maximum, etc., et des graphiques, tels que l'histogramme des fréquences. Elles permettent, entre autres, d'identifier les tendances, de visualiser les dispersions, ou simplement de repérer les valeurs extrêmes prises par une variable.

Dans ce qui suit, nous allons effectuer l'analyse descriptive sur les métriques de couplage et de taille pour chacun des systèmes étudiés (AMQ, ANT et Camel). Pour chaque système, nous présenterons un tableau de statistiques générales, et les histogrammes de chaque métrique.

Tableau 5 : Statistiques descriptives AMQ.

Statistique	WMC	CBO	RFC	Ca	Ce	NPM	MOA	IC	CBM	LOC
Nb. d'observations	1073	1073	1073	1073	1073	1073	1073	1073	1073	1073
Minimum	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Maximum	155,00	216,00	344,00	212,00	78,00	121,00	21,00	2,00	10,00	2990,00
Eff. du minimum	8	7	8	117	98	16	710	1030	1030	4
Eff. du maximum	1	1	1	1	1	1	1	2	1	1
1er Quartile	4,00	4,00	9,00	1,00	2,00	3,00	0,00	0,00	0,00	36,00
Médiane	8,00	7,00	21,00	2,00	5,00	7,00	0,00	0,00	0,00	100,00
3e Quartile	13,00	12,00	35,00	5,00	7,00	10,00	1,00	0,00	0,00	215,00
Moyenne	11,43	12,23	29,69	6,38	6,37	9,42	0,88	0,04	0,06	205,27
Variance (n)	192,15	363,92	1240,28	302,53	69,43	126,73	4,22	0,04	0,19	116433,58
Écart-type (n)	13,86	19,08	35,22	17,39	8,33	11,26	2,05	0,21	0,43	341,22

Pour le système AMQ, l'analyse du tableau 5 montre 1073 classes contenant en moyenne 205 lignes de code, ce qui est assez considérable même si la moitié d'entre elles ont moins de 100 lignes de code. On note un grand effectif du minimum (0) pour MOA, IC, et CBM, ce qui montre une faible utilisation du couplage par attribut et par héritage, d'autant plus que 75% des classes ont une valeur nulle pour ces deux dernières métriques. Par contre, la moyenne et la médiane sont relativement élevées pour le CBO avec des valeurs d'au moins 22

(histogramme CBO, figure 11) pour environ 12% des classes et atteignant un max de 216, ce qui indique une utilisation assez importante du couplage. L'écart entre le premier, le deuxième et le troisième quartile pour la métrique RFC démontre une forte variation entre les classes. 98 classes ont un couplage sortant nul, contre 117 pour le couplage entrant. Le premier groupe correspondant souvent à des classes génériques de haut niveau d'abstraction, et le second à des classes implémentant des concepts plus concrets. On note aussi une variabilité plus faible de la métrique Ca comparée à Ce. La grande majorité des classes a un couplage entrant compris entre 0 et 22 (voir histogramme Ca, figure 11), tandis qu'une petite partie des classes a une valeur de Ca très largement supérieure au troisième quartile, reflétant ainsi le fait qu'une infime minorité des classes est largement utilisée par le reste du système.

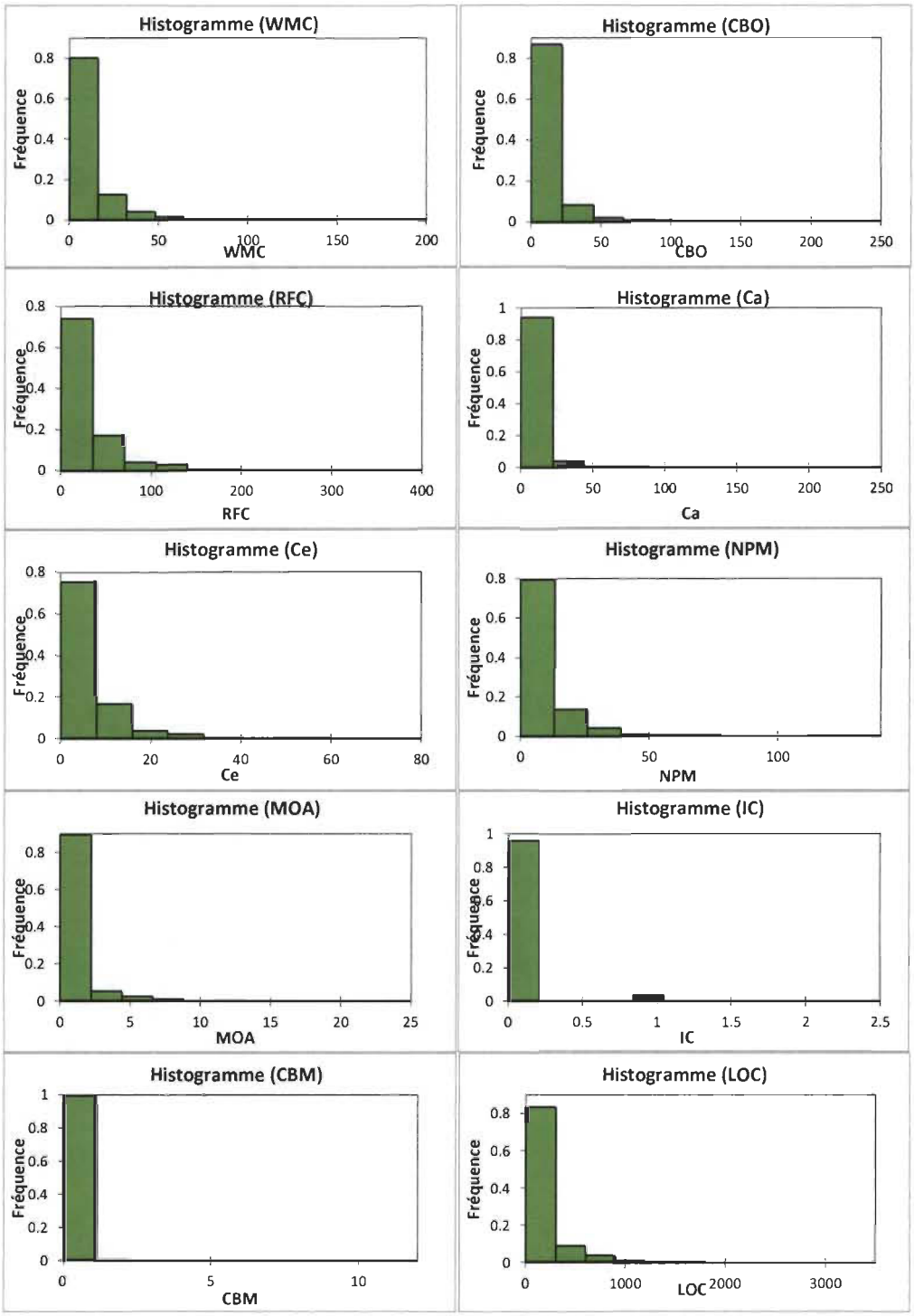


Figure 11 : Histogrammes des métriques - AMQ.

Le tableau 6 donne les statistiques générales des métriques pour le système ANT. Le nombre de classes (126) est largement inférieur à celui d'AMQ mais avec un plus grand pourcentage de larges classes (25% des classes ont au moins 400 lignes de code). On note un grand effectif du minimum (0) pour MOA, IC, et CBM, ce qui montre une faible utilisation du couplage par attribut et par héritage, mais dans de moindres proportions comparées à AMQ. On note en effet que la moyenne, le maximum et le troisième quartile de ces deux dernières métriques (IC et CBM) sont légèrement plus hauts, même si les moyennes de la plupart des métriques restent sensiblement proches entre ces deux systèmes. Par contre, on trouve une moyenne et un troisième quartile relativement élevés pour la métrique RFC avec des valeurs d'au moins 46 pour 25% des classes, ce qui est cohérant vu le fort taux de classes de taille assez élevée, évoqué un peu plus haut. Les histogrammes (voir figure 12) montrent une plus forte variabilité pour le couplage sortant Ce que pour le couplage entrant, ce qui met en évidence le fait qu'il y a un petit nombre de classes avec un Ca très élevé qui sont largement utilisées par les autres classes du système.

Tableau 6 : Statistiques descriptives ANT.

Statistique	WMC	CBO	RFC	Ca	Ce	NPM	MOA	IC	CBM	LOC
Nb. d'observations	126	126	126	126	126	126	126	126	126	126
Minimum	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Maximum	71,00	103,00	186,00	99,00	21,00	59,00	7,00	4,00	7,00	2193,00
Eff. du minimum	2	2	2	43	15	3	78	67	67	1
Eff. du maximum	1	1	1	1	1	1	1	2	2	1
1er Quartile	4,00	4,00	15,00	0,00	2,00	4,00	0,00	0,00	0,00	84,75
Médiane	8,00	7,00	28,50	1,00	4,50	5,50	0,00	0,00	0,00	168,50
3e Quartile	14,00	10,00	46,00	4,00	7,75	11,00	1,00	1,00	1,75	405,50
Moyenne	10,57	10,38	34,42	5,52	5,23	8,34	0,70	0,81	1,13	301,07
Variance (n)	105,77	218,55	829,20	220,17	19,35	74,89	1,43	1,11	2,78	113201,42
Écart-type (n)	10,28	14,78	28,80	14,84	4,40	8,65	1,20	1,05	1,67	336,45

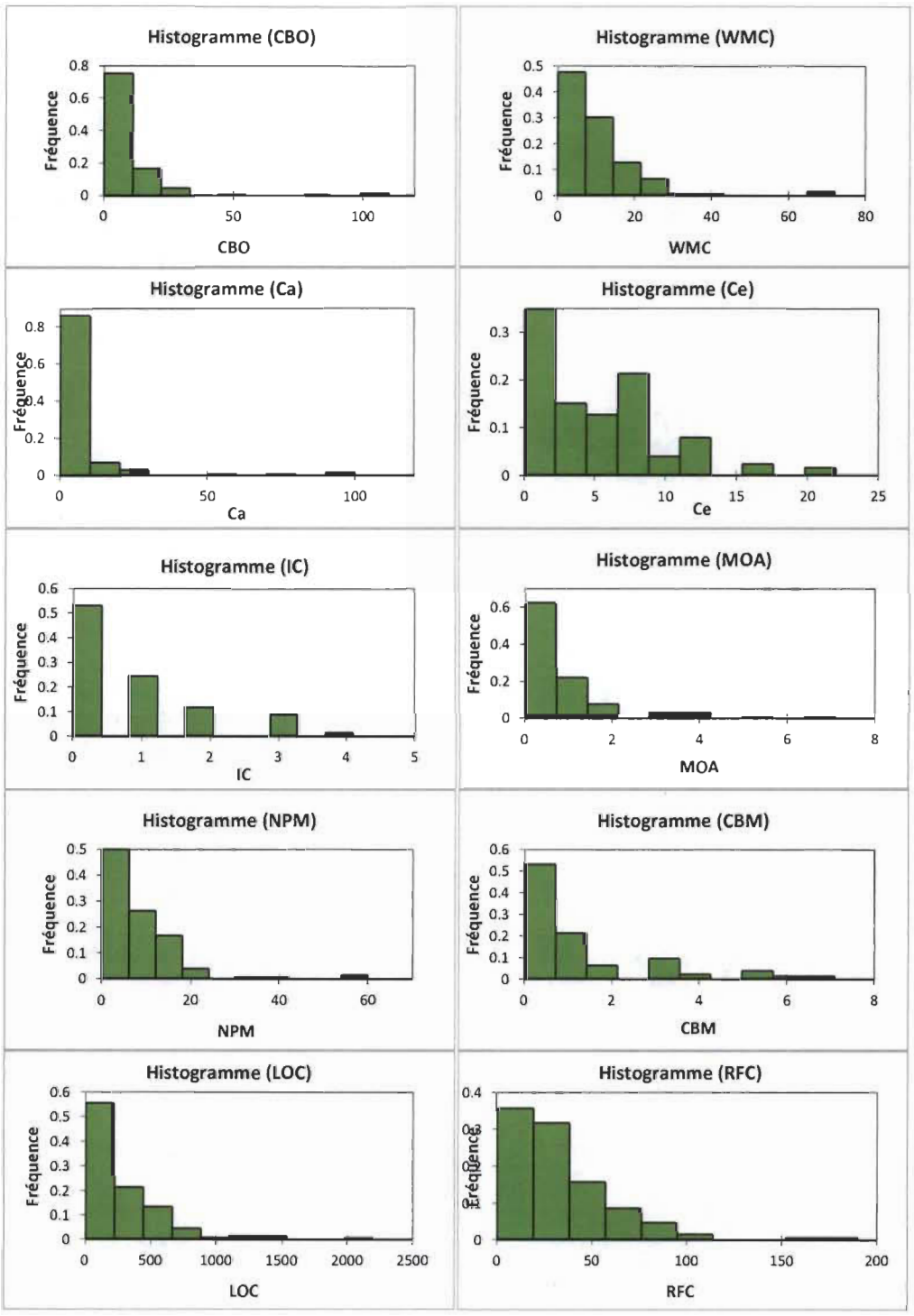


Figure 12 : Histogrammes des métriques - ANT.

Le tableau 7 donne les statistiques descriptives pour CAMEL. Le nombre de classes (608) est largement supérieur à celui d'ANT mais de plus petites tailles (75% ont moins de 141 lignes de code) d'une moyenne d'environ 109 lignes de code par classe. On note que 50% des classes ont la valeur nulle pour les métriques MOA, IC et CBM, ce qui montre une faible utilisation du couplage par attribut et par héritage (comme pour les deux autres systèmes). Les valeurs des quartiles et de la moyenne de RFC sont plus faibles que pour les deux autres systèmes, ce qui est cohérent vu que la taille des classes est en moyenne plus petite pour ce système. La moyenne pour la métrique Ca est légèrement supérieure au troisième quartile et vient confirmer le fait qu'une petite partie des classes a un couplage entrant très élevé (possiblement le noyaux), et est donc très sollicitée par le reste des classes du système.

Tableau 7 : Statistiques descriptives CAMEL.

Statistique	WMC	CBO	RFC	Ca	Ce	NPM	LOC	MOA	IC	CBM
Nb. d'observations	608	608	608	608	608	608	608	608	608	608
Minimum	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Maximum	94,00	272,00	186,00	270,00	53,00	86,00	1056,00	9,00	3,00	14,00
Eff. du minimum	12	26	12	96	70	14	11	368	415	415
Eff. du maximum	1	1	1	1	1	1	1	2	1	1
1er Quartile	3,00	3,00	6,00	1,00	1,00	2,00	21,00	0,00	0,00	0,00
Médiane	5,00	6,00	14,00	1,00	4,00	4,00	56,50	0,00	0,00	0,00
3e Quartile	11,00	11,00	27,25	4,00	8,00	8,00	141,25	1,00	1,00	1,00
Moyenne	8,31	10,10	20,23	5,02	5,62	6,74	109,05	0,74	0,36	0,64
Variance (n)	92,93	331,80	434,42	313,48	33,62	75,31	21048,92	1,56	0,32	1,68
Écart-type (n)	9,64	18,22	20,84	17,71	5,80	8,68	145,08	1,25	0,56	1,30

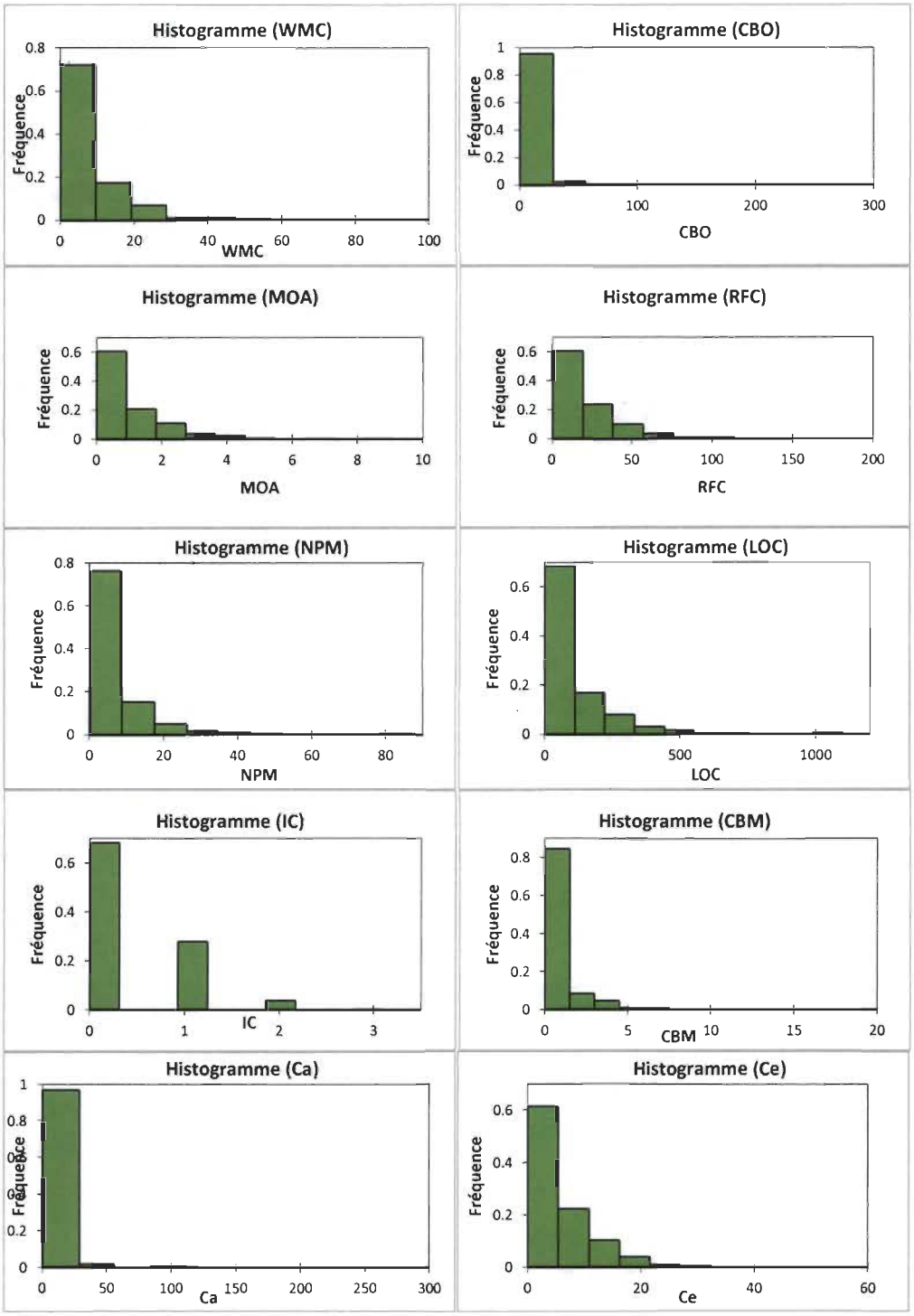


Figure 13 : Histogrammes des métriques - CAMEL.

Sur la base de distinctions des tailles de projets/systèmes énoncées par [Briand 99], on dispose d'un système de grande taille (AMQ avec plus de 1000 classes), d'un système de taille moyenne (CAMEL avec un nombre de classes compris entre 200 et 1000 classes), et d'un système de petite taille (ANT avec moins de 200 classes)

5.2.1.2 Évolution des métriques à travers les versions

La figure 14 donne une vue de l'évolution des moyennes des métriques de couplage entre les versions et entre les systèmes. Nous avons ajouté un deuxième axe vertical représentant la taille (en termes de nombre de classes) des versions. Le choix des versions a été orienté par le nombre de fautes sévères (critical ou blocker) découvertes. Nous avons retenu 5 versions d'ANT (1.3-1.7), 3 versions d'AMQ (5.0-5.2) et 2 versions de Camel (1.2 et 1.4).

On note une augmentation de la taille des systèmes au cours de leur évolution. Les moyennes des métriques Ce, Ca et CBO semblent très peu varier entre les différentes versions et systèmes, ce qui prédit des distributions un peu similaires pour ces métriques d'une version à une autre et voire d'un système à un autre. On peut en déduire, par exemple, qu'une classe a en moyenne un couplage sortant (Ce) compris entre 5 et 7. Il y a une assez grande variation de RFC entre le système ANT et les autres. Cependant, cette variation reste très faible entre les versions (d'un même système). La taille (nombre de classes) ne semble pas trop influencer la moyenne de cette métrique, même si on note une légère augmentation entre CAMEL et AMQ. C'est le constat opposé pour les métriques IC et CBM dont les tendances semblent liées à la taille des systèmes. En effet, on note une très légère diminution des moyennes des couplages par héritage quand le nombre de classes des systèmes augmente. Les variations restent assez minimes quand les systèmes évoluent. La métrique MOA varie très peu entre les systèmes et les versions, et prend des valeurs légèrement supérieures pour le système AMQ.

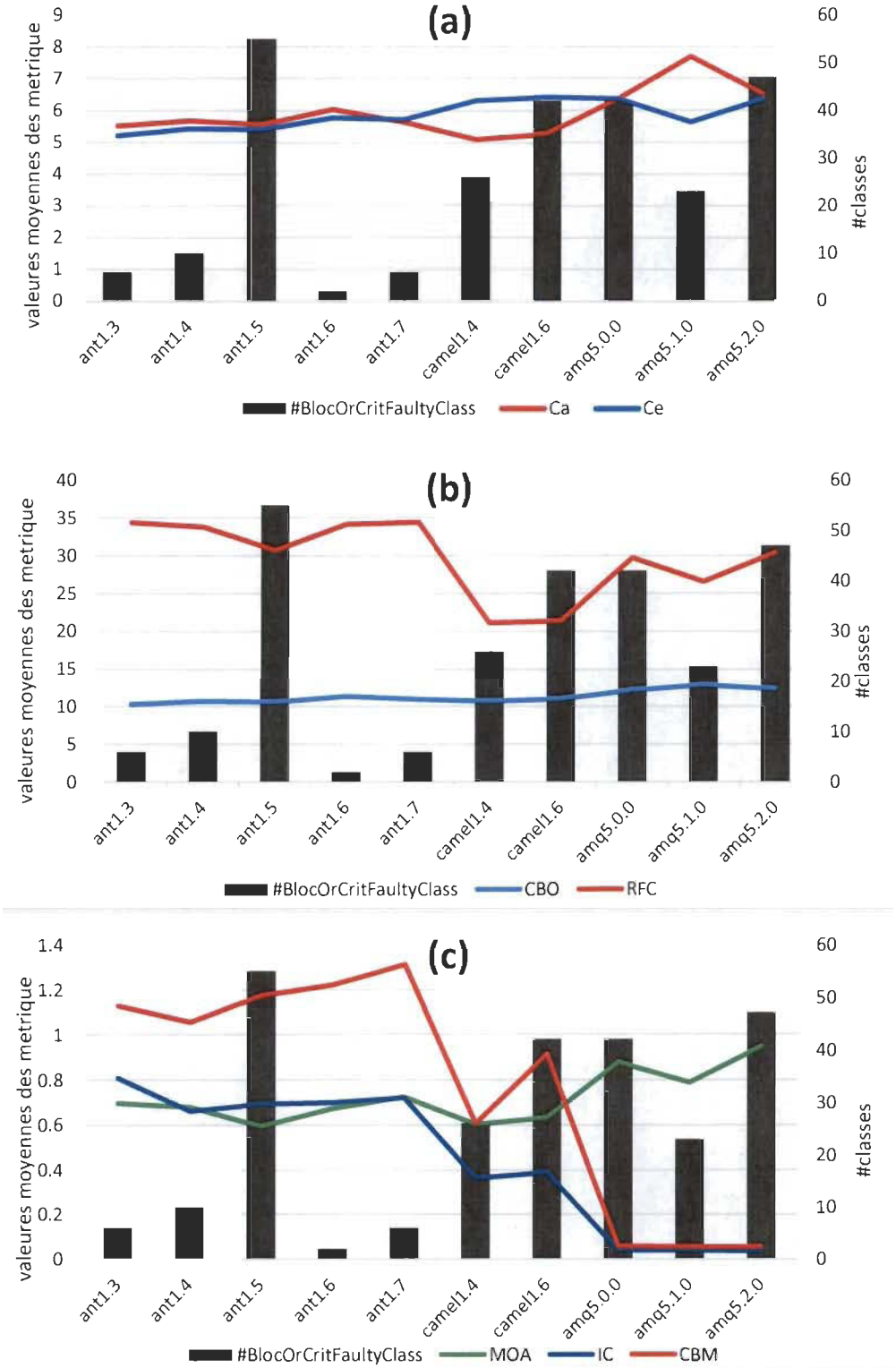


Figure 14 : Évolutions des métriques de couplage : a) Ca, Ce ; b) CBO, FRC ; c) MOA, IC, CBM

5.2.2 Relation entre les différentes métriques de couplage

5.2.2.1 Analyse de corrélation entre les métriques

Le tableau 8 montre les corrélations entre les différentes métriques calculées pour les trois systèmes sous étude. En nous basant sur les suggestions de Hopkins [Zhou 06], nous allons considérer les coefficients de corrélations inférieurs à 0.1 comme des corrélations mineures, entre 0.1-0.3 triviales, entre 0.3-0.5 modérées, 0.5-0.7 fortes, 0.7-09 très fortes et 0.9-1 presque parfaites. Ainsi, nous pouvons déjà remarquer (pour tous les systèmes) des corrélations presque parfaites (positives) entre la métrique WMC, qui somme les complexités cyclomatiques des méthodes, et NPM, qui compte le nombre de méthodes publiques. Ces deux métriques, en plus de LOC, capturent la taille des classes. Les corrélations (positives) des métriques WMC et NPM avec la métrique LOC sont moins fortes, bien qu'assez importantes (entre 0.55 et 0.66 pour WMC et LOC). En effet, l'idée qu'un grand nombre de méthodes induirait un grand nombre de lignes de code est moins évidente pour les classes génériques. Par exemple, les interfaces ont souvent un grand nombre de méthodes, mais avec un nombre de lignes de code (LOC) se limitant juste à la déclaration de ces dernières, ou encore les classes se trouvant à la racine d'un arbre d'héritage peuvent contenir un grand nombre de déclarations de méthodes abstraites dont les comportements seront effectivement définis par les classes filles. Il serait donc intéressant d'analyser la taille en combinant ces deux perspectives. On capture de façon quelque peu implicite ces deux dimensions de la taille dans la mesure de RFC. En effet, pour le calcul de RFC, CKJM fait la somme du nombre de méthodes de la classe et du nombre de méthodes d'autres classes invoquées dans la classe. Ce constat peut expliquer les fortes corrélations notées entre RFC et ces métriques de taille. D'un autre côté, la métrique RFC se trouve être fortement corrélée avec le couplage sortant (Ce) tandis que sa corrélation avec le couplage entrant (Ca) reste modérée (voir triviale pour CAMEL). En effet, RFC capture le couplage sortant via l'appel de méthodes d'autres classes (une partie de

celles comptées lors du calcul de Ce). On note aussi que la métrique Ce a des corrélations modérées (voir forte pour AMQ) avec MOA et LOC et CBO. Quant au couplage sortant Ca, il présente des corrélations presque parfaites avec CBO. On constate des liens d'intensités variées entre la plupart de ces métriques. Seul le couplage par héritage (IC, CBM) semble ne pas avoir de liens significatifs avec les autres métriques.

Tableau 8 : Corrélations entre les métriques.

Métrique	Métrique	Corr. AMQ	Corr. ANT	Corr. Camel
Ca	CBM	0,05	-0,10	-0,06
Ca	Ce	0,06	-0,05	-0,01
Ca	IC	0,13	-0,15	-0,07
Ca	LOC	0,11	0,29	0,01
Ca	MOA	0,21	0,13	-0,01
Ca	NPM	0,30	0,40	0,14
CBM	LOC	0,02	0,16	0,12
CBO	Ca	**0,92**	**0,96**	**0,96**
CBO	CBM	0,02	-0,01	-0,02
CBO	Ce	0,44	0,20	0,27
CBO	IC	0,08	-0,02	-0,01
CBO	LOC	0,33	0,40	0,17
CBO	MOA	0,43	0,25	0,13
CBO	NPM	0,48	0,49	0,25
CBO	RFC	0,42	0,43	0,23
Ce	CBM	-0,06	0,28	0,15
Ce	IC	-0,08	0,42	0,23
Ce	LOC	0,60	0,50	0,61
Ce	MOA	0,65	0,46	0,48
Ce	NPM	0,57	0,42	0,49
IC	CBM	**0,80**	**0,87**	0,77
IC	LOC	0,04	0,23	0,12
MOA	CBM	-0,01	0,07	0,05
MOA	IC	-0,01	0,07	0,15
MOA	LOC	0,56	0,28	0,35
NPM	CBM	0,02	0,15	0,13
NPM	IC	0,04	0,21	0,07
NPM	LOC	0,59	0,69	0,72
NPM	MOA	0,62	0,42	0,26
RFC	Ca	0,13	0,27	0,03
RFC	CBM	0,00	0,22	0,15
RFC	Ce	**0,80**	0,66	0,73
RFC	IC	0,02	0,32	0,15
RFC	LOC	**0,86**	**0,91**	**0,93**
RFC	MOA	0,69	0,42	0,38
RFC	NPM	0,76	0,75	0,79
WMC	Ca	0,25	0,42	0,12
WMC	CBM	0,02	0,14	0,15
WMC	CBO	0,46	0,52	0,26
WMC	Ce	0,63	0,46	0,58
WMC	IC	0,06	0,20	0,13
WMC	LOC	0,74	**0,81**	**0,81**
WMC	MOA	0,67	0,43	0,33
WMC	NPM	**0,94**	**0,95**	**0,97**
WMC	RFC	**0,86**	**0,86**	**0,87**

5.2.2.2 ACP

Le tableau 9 donne des indications sur le niveau de significativité des axes principaux. Pour chaque axe, nous avons la valeur propre correspondante, la

proportion de variance totale expliquée, et la proportion de variance cumulée. Pour les trois systèmes, les trois premiers axes expliquent au moins 83% de la variance totale.

Les tableaux 10, 11 et 12 donnent les informations sur les liens entre les variables et les axes pour AMQ, ANT et CAMEL (dans l'ordre).

Avant la rotation, nous remarquons que l'axe le plus important (AXIS_1) a une très forte corrélation avec Ce et RFC sur l'ensemble des trois systèmes. AXIS_1 est la dimension la plus représentative du couplage sortant. Le deuxième axe (AXIS_2) est fortement corrélé avec le couplage par héritage sur l'ensemble des trois systèmes. Pour AMQ, 77% de la variance de CBM est contenue dans cet axe (AXIS_2), contre 82% pour IC. Sur les deux autres systèmes, CBO et Ca ont de fortes corrélations avec le deuxième axe, mais dans un sens opposé au couplage par héritage, ce qui rend moins évidente son interprétation. Nous observons les mêmes conclusions pour le troisième axe qui a des corrélations plus ou moins modérées avec la plupart des métriques. Ces trois axes résument une grande partie de la variance contenue dans les métriques. Par exemple, pour AMQ, ces trois premiers axes suffisent pour capturer 100% de la variance de Ca, 90% d'IC et CBM, 85 et 86% de Ce et RFC.

La rotation VARIMAX, donne une représentation des données sur des nouveaux axes orthogonaux plus faciles à interpréter. Pour AMQ, cinq axes principaux se dégagent.

Le premier axe est fortement corrélé à RFC et presque parfaitement corrélé avec Ce. Il résume 87% de la variance pour l'un et 37% pour l'autre pour 21% de la variance totale. Le deuxième axe constitue 26% de la variance totale et explique 91 et 88% de la variance de CBM et IC. Le troisième contient 99 et 88% de la variance de Ca et CBO et explique 28% de la variance totale des données. Le quatrième axe est très fortement corrélé avec MOA (80% de sa variance) et capture 15% de la variance totale. Le cinquième axe résume 8 % de la variance totale et explique 46% de celle de la métrique RFC avec laquelle il est fortement corrélé.

Nous observons les mêmes tendances pour les deux autres systèmes. On peut tirer quelques remarques intéressantes. D'abord, on note une distinction assez nette entre le couplage entrant, le couplage sortant et le couplage par héritage. Le couplage sortant est réparti sur trois axes représentant RFC, Ce et MOA. La proximité entre Ce et RFC se manifeste par le fait qu'une partie de la variance de cette dernière s'explique dans l'axe représentant Ce. Le couplage entrant est représenté par Ca et CBO dans un seul axe. La métrique CBO capture à la fois le couplage entrant et sortant, mais d'après nos observations, elle reste plus proche de Ca que de Ce. Le couplage par héritage est aussi représenté sur un seul axe.

Tableau 9 : Informations (valeurs propres) sur les axes principaux.

	Axes	1	2	3	4	5	6	7	Tot.
AMQ	Valeurs propres	2,94	1,86	1,42	0,37	0,22	0,18	0	7
	Différence	1,07	0,44	1,04	0,15	0,04	0,18	-	-
	Proportion	41,95%	26,63%	20,28%	5,35%	3,16%	2,62%	0,02%	-
	Cumulatif	41,95%	68,58%	88,86%	94,20%	97,36%	99,98%	100,00%	-
ANT	Valeurs propres	2,72	2,12	1,18	0,59	0,29	0,11	0	7
	Différence	0,6	0,94	0,59	0,3	0,18	0,1	-	-
	Proportion	38,82%	30,27%	16,80%	8,41%	4,08%	1,55%	0,06%	-
	Cumulatif	38,82%	69,10%	85,90%	94,31%	98,39%	99,94%	100,00%	-
Camel	Valeurs propres	2,43	1,97	1,44	0,67	0,28	0,21	0	7
	Différence	0,45	0,53	0,78	0,39	0,07	0,2	-	-
	Proportion	34,67%	28,21%	20,64%	9,53%	3,99%	2,93%	0,02%	-
	Cumulatif	34,67%	62,88%	83,52%	93,05%	97,05%	99,98%	100,00%	-

Tableau 10 : Corrélations et communalités (AMQ).

Facteurs de charge après rotation										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
Ce	**0,93**	87 % (87 %)	-0,06	0 % (88 %)	0,10	1 % (89 %)	0,29	9 % (97 %)	0,16	3 % (100 %)
RFC	0,61	37 % (37 %)	0,02	0 % (37 %)	0,13	2 % (39 %)	0,36	13 % (52 %)	0,70	48 % (100 %)
CBM	-0,01	0 % (0 %)	**0,95**	91 % (91 %)	0,00	0 % (91 %)	0,00	0 % (91 %)	-0,01	0 % (91 %)
IC	-0,04	0 % (0 %)	**0,94**	88 % (88 %)	0,07	1 % (89 %)	-0,01	0 % (89 %)	0,02	0 % (89 %)
Ca	-0,07	1 % (1 %)	0,05	0 % (1 %)	**0,99**	99 % (99 %)	0,06	0 % (100 %)	0,03	0 % (100 %)
CBO	0,30	9 % (9 %)	0,02	0 % (9 %)	**0,94**	88 % (97 %)	0,16	3 % (99 %)	0,08	1 % (100 %)
MOA	0,37	13 % (13 %)	0,00	0 % (13 %)	0,18	3 % (17 %)	**0,90**	80 % (97 %)	0,17	3 % (100 %)
Var. Expl.	1,48	21 % (21 %)	1,80	26 % (47 %)	1,93	28 % (74 %)	1,05	15 % (89 %)	0,55	8 % (97 %)
Facteurs de charge										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
Ce	-0,81	65 % (65 %)	0,25	6 % (71 %)	0,37	14 % (85 %)	0,27	7 % (93 %)	-0,21	4 % (97 %)
RFC	-0,83	68 % (68 %)	0,15	2 % (71 %)	0,39	15 % (86 %)	0,16	3 % (88 %)	0,26	7 % (95 %)
CBM	-0,01	0 % (0 %)	-0,88	77 % (77 %)	0,36	13 % (90 %)	0,00	0 % (90 %)	-0,24	6 % (96 %)
IC	-0,04	0 % (0 %)	-0,90	82 % (82 %)	0,28	8 % (90 %)	0,01	0 % (90 %)	0,22	5 % (95 %)
Ca	-0,55	30 % (30 %)	-0,35	12 % (43 %)	-0,75	57 % (100 %)	-0,02	0 % (100 %)	0,03	0 % (100 %)
CBO	-0,80	65 % (65 %)	-0,22	5 % (70 %)	-0,54	29 % (99 %)	0,10	1 % (99 %)	-0,06	0 % (100 %)
MOA	-0,81	65 % (65 %)	0,12	2 % (66 %)	0,26	7 % (73 %)	-0,52	27 % (100 %)	-0,04	0 % (100 %)
Var. Expl.	2,94	42 % (42 %)	1,86	27 % (69 %)	1,42	20 % (89 %)	0,37	5 % (94 %)	0,22	3 % (97 %)

Tableau 11 : Corrélations et communalités (ANT).

Facteurs de charge après rotation										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
CBM	**0,97**	95 % (95 %)	-0,02	0 % (95 %)	0,06	0 % (95 %)	0,04	0 % (95 %)	0,05	0 % (96 %)
IC	**0,93**	87 % (87 %)	-0,07	0 % (87 %)	0,19	4 % (91 %)	0,00	0 % (91 %)	0,14	2 % (93 %)
Ca	-	1 % (1 %)	**0,99**	98 % (98 %)	-0,09	1 % (99 %)	0,03	0 % (99 %)	0,08	1 % (100 %)
CBO	-	0 % (0 %)	**0,97**	94 % (94 %)	0,13	2 % (96 %)	0,11	1 % (97 %)	0,16	2 % (100 %)
Ce	0,21	4 % (4 %)	0,01	0 % (5 %)	**0,89**	79 % (84 %)	0,24	6 % (90 %)	0,32	10 % (100 %)
MOA	0,02	0 % (0 %)	0,10	1 % (1 %)	0,19	4 % (5 %)	**0,96**	93 % (98 %)	0,15	2 % (100 %)
RFC	0,16	2 % (2 %)	0,24	6 % (8 %)	0,33	11 % (19 %)	0,20	4 % (23 %)	**0,88**	77 % (100 %)
Var. Expl.	1,89	27 % (27 %)	2,00	29 % (56 %)	1,01	14 % (70 %)	1,04	15 % (85 %)	0,95	14 % (98 %)
Facteurs de charge										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
CBM	-	27 % (27 %)	-0,64	40 % (68 %)	-0,48	23 % (91 %)	-0,20	4 % (95 %)	0,04	0 % (95 %)
IC	-	34 % (34 %)	-0,67	45 % (79 %)	-0,38	15 % (94 %)	-0,04	0 % (94 %)	-0,01	0 % (94 %)
Ca	-	17 % (17 %)	0,81	66 % (83 %)	-0,41	17 % (99 %)	-0,06	0 % (100 %)	-0,02	0 % (100 %)
CBO	-	36 % (36 %)	0,73	53 % (90 %)	-0,30	9 % (99 %)	-0,01	0 % (99 %)	-0,11	1 % (100 %)
Ce	-	58 % (58 %)	-0,22	5 % (63 %)	0,42	17 % (80 %)	0,26	7 % (87 %)	-0,36	13 % (100 %)
MOA	-	32 % (32 %)	0,11	1 % (33 %)	0,56	31 % (65 %)	-0,59	35 % (100 %)	0,06	0 % (100 %)
RFC	-	67 % (67 %)	0,08	1 % (68 %)	0,23	5 % (73 %)	0,36	13 % (86 %)	0,37	14 % (100 %)
Var. Expl.	2,72	39 % (39 %)	2,12	30 % (69 %)	1,18	17 % (86 %)	0,59	8 % (94 %)	0,29	4 % (98 %)

Tableau 12 : Corrélations et communalités (CAMEL).

Facteurs de charge après rotation										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
RFC	**0,93**	86 % (86 %)	0,07	1 % (87 %)	0,08	1 % (87 %)	0,17	3 % (90 %)	0,31	10 % (100 %)
Ca	-0,01	0 % (0 %)	**1,00**	99 % (99 %)	-0,04	0 % (99 %)	-0,02	0 % (99 %)	-0,08	1 % (100 %)
CBO	0,11	1 % (1 %)	**0,98**	96 % (97 %)	-0,01	0 % (97 %)	0,06	0 % (97 %)	0,16	3 % (100 %)
CBM	0,07	1 % (1 %)	-0,02	0 % (1 %)	**0,96**	93 % (94 %)	-0,01	0 % (94 %)	0,03	0 % (94 %)
IC	0,04	0 % (0 %)	-0,03	0 % (0 %)	**0,90**	81 % (82 %)	0,09	1 % (82 %)	0,10	1 % (83 %)
MOA	0,16	3 % (3 %)	0,03	0 % (3 %)	0,04	0 % (3 %)	**0,97**	94 % (97 %)	0,18	3 % (100 %)
Ce	0,44	19 % (19 %)	0,07	0 % (20 %)	0,11	1 % (21 %)	0,26	7 % (27 %)	**0,85**	73 % (100 %)
Var. Expl.	1,10	16 % (16 %)	1,96	28 % (44 %)	1,77	25 % (69 %)	1,04	15 % (84 %)	0,90	13 % (97 %)
Facteurs de charge										
Métriques	**Axis_1**		**Axis_2**		**Axis_3**		**Axis_4**		**Axis_5**	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
RFC	-0,77	59 % (59 %)	-0,08	1 % (59 %)	0,38	15 % (74 %)	-0,38	14 % (88 %)	-0,31	10 % (98 %)
Ca	-0,33	11 % (11 %)	0,85	73 % (84 %)	-0,40	16 % (99 %)	0,06	0 % (100 %)	-0,05	0 % (100 %)
CBO	-0,54	30 % (30 %)	0,79	62 % (92 %)	-0,27	8 % (100 %)	0,00	0 % (100 %)	0,05	0 % (100 %)
CBM	-0,43	19 % (19 %)	-0,54	29 % (48 %)	-0,64	41 % (89 %)	-0,04	0 % (89 %)	-0,16	3 % (92 %)
IC	-0,49	24 % (24 %)	-0,55	30 % (54 %)	-0,58	34 % (88 %)	0,09	1 % (89 %)	0,15	2 % (91 %)
MOA	-0,59	35 % (35 %)	-0,10	1 % (36 %)	0,40	16 % (52 %)	0,69	47 % (99 %)	-0,10	1 % (100 %)
Ce	-0,81	66 % (66 %)	-0,11	1 % (67 %)	0,39	15 % (82 %)	-0,20	4 % (86 %)	0,35	12 % (98 %)
Var. Expl.	2,43	35 % (35 %)	1,97	28 % (63 %)	1,44	21 % (84 %)	0,67	10 % (93 %)	0,28	4 % (97 %)

5.2.2.3 VAR-CLUST

Nous allons nous servir de deux techniques de clustering (VARKMeans et VARHCA) pour essayer de regrouper les métriques de couplage selon leurs ressemblances.

Les tableaux 13, 15 et 17 donnent une information résumée des clusters et les tableaux 14, 16 et 18 donnent les corrélations entre les métriques et les différents clusters, pour les systèmes AMQ, ANT et CAMEL (resp.).

La technique VARKMean donne 3 clusters de deux métriques et 1 cluster d'une métrique pour un total de 4 clusters, ce qui laisse supposer quatre catégories de métriques de couplage.

La technique VARHCA donne 3 clusters (sauf pour ANT, pour qui elle donne la même chose que VARKMean) dont deux de 2 métriques et 1 de 3 (trois) métriques, ce qui laisse supposer 3 catégories de métriques de couplage. En effet, l'analyse des corrélations (clusters-métriques) permet d'identifier clairement trois catégories, à savoir :

- Le couplage entrant : les classificateurs ont regroupé les métriques CBO et Ca pour former un cluster. Étant donné que CBO mesure aussi une partie du

couplage entrant, on considère donc ce cluster comme représentant le couplage entrant.

- Le couplage sortant : La proximité entre les métriques RFC et Ce est mise en évidence par les deux techniques de regroupement, et est confirmée sur les trois systèmes. La proximité de MOA avec ces deux dernières métriques est moins grande. En effet, avec VARHCA, on retrouve dans le même cluster les métriques RFC, Ce et MOA (pour AMQ et CAMEL), mais avec un coefficient de corrélation plus faible pour ce dernier. Alors qu'avec VARKMean, MOA représente à elle seule un cluster. On en déduit que Ce et RFC sont les plus représentatives du couplage entrant, même si MOA et CBO capturent elles aussi une part de celui-ci.
- Le couplage par héritage : IC et CBM restent de façon constante les seules constituantes du même cluster sur les trois systèmes et ceci pour les deux techniques confondues.

Tableau 13 : Résumé des clusters (AMQ).

VARKMeans				VARHCA			
Cluster	**# Members**	**Var. Expl.**	**Prop. Expl.**	**Cluster**	**# Members**	**Var. Expl.**	**Prop. Expl.**
1	1	1,00	1,00	1	2	1,80	0,90
2	2	1,80	0,90	2	3	2,43	0,81
3	2	1,92	0,96	3	2	1,92	0,96
4	2	1,80	0,90	-	-	-	-
Total		6,52	0,93	Total		6,14	0,88

Tableau 14 : Corrélations clusters - métriques (AMQ).

VARKMeans						VARHCA				
Métriq	**# Clust**	**Clust 1**	**Clust 2**	**Clust 3**	**Clust 4**	**Metriq**	**#clust**	**Clust 1**	**Clust 2**	**Clust 3**
CBO	1	0,43	0,05	0,98	0,45	CBO	1	0,05	0,48	0,98
RFC	1	0,69	0,01	0,28	0,95	RFC	1	0,01	0,92	0,28
Ca	1	0,21	0,09	0,98	0,10	Ca	1	0,09	0,15	0,98
Ce	1	0,65	-0,08	0,25	0,95	Ce	1	-0,08	0,91	0,25
MOA	2	1,00	-0,01	0,33	0,71	MOA	1	-0,01	0,86	0,33
IC	1	-0,01	0,95	0,11	-0,03	IC	1	0,95	-0,03	0,11
CBM	1	-0,01	0,95	0,04	-0,03	CBM	1	0,95	-0,03	0,04

Tableau 15 : Résumé clusters (ANT).

VARKMeans				VARHCA			
Cluster	**# Membres**	**Var. Expl.**	**Prop. Expl.**	**Cluster**	**# Membres**	**Var. Expl.**	**Prop. Expl.**
1	1	1,00	1,00	1	2	1,96	0,98
2	2	1,87	0,94	2	2	1,87	0,94
3	2	1,96	0,98	3	2	1,66	0,83
4	2	1,66	0,83	4	1	1,00	1,00
Total		6,50	0,93	Total		6,50	0,93

Tableau 16 : Corrélations clusters - métriques (ANT).

VARKMeans						VARHCA				
Métriques	**# Clust**	**Clust 1**	**Clust 2**	**Clust 3**	**Clust 4**	**Métriques**	**# Clust**	**Clust 1**	**Clust 2**	**Clust 3**
CBO	1	0,43	0,05	0,98	0,45	CBO	1	0,05	0,48	0,98
RFC	1	0,69	0,01	0,28	0,95	RFC	1	0,01	0,92	0,28
Ca	1	0,21	0,09	0,98	0,10	Ca	1	0,09	0,15	0,98
Ce	1	0,65	-0,08	0,25	0,95	Ce	1	-0,08	0,91	0,25
MOA	2	1,00	-0,01	0,33	0,71	MOA	1	-0,01	0,86	0,33
IC	1	-0,01	0,95	0,11	-0,03	IC	1	0,95	-0,03	0,11
CBM	1	-0,01	0,95	0,04	-0,03	CBM	1	0,95	-0,03	0,04

Tableau 17 : Résumé clusters (CAMEL).

VARKMeans				VARHCA			
Cluster	**# Membres**	**Var.**	**Prop.**	**Cluster**	**# Membres**	**Var. Expl.**	**Prop. Expl.**
1	1	1,00	1,00	1	2	1,96	0,98
2	2	1,77	0,89	2	3	2,07	0,69
3	2	1,96	0,98	3	2	1,77	0,89
4	2	1,73	0,86	-	-	-	-
Total		6,46	0,92	Total		5,81	0,83

Tableau 18 : Corrélations cluster métriques (CAMEL).

VARKMeans						VARHCA				
Métriques	**# Clust**	**Clust 1**	**Clust 2**	**Clust 3**	**Clust 4**	**Métriques**	**# Clust**	**Clust 1**	**Clust 2**	**Clust 3**
CBO	1	0,13	-0,01	0,99	0,27	CBO	1	0,99	0,26	-0,01
RFC	1	0,38	0,16	0,13	0,93	RFC	1	0,13	0,87	0,16
Ca	1	-0,01	-0,07	0,99	0,01	Ca	1	0,99	0,01	-0,07
Ce	1	0,48	0,20	0,13	0,93	Ce	1	0,13	0,90	0,20
MOA	1	1,00	0,11	0,06	0,46	MOA	1	0,06	0,71	0,11
IC	1	0,15	0,94	-0,04	0,21	IC	1	-0,04	0,22	0,94
CBM	1	0,05	0,94	-0,04	0,16	CBM	1	-0,04	0,15	0,94

5.2.3 Évaluation des liens entre le couplage et la sévérité des fautes

Le chapitre précédent a permis de dégager les différentes formes de couplage et de voir le recoupement de l'information qu'elles véhiculent. Nous allons maintenant explorer leurs liens avec la sévérité des fautes dans les classes. Nous allons effectuer une analyse de régression logistique univariée, puis multivariée. Une méthode de sélection de variables (STEPDISC Ascendant) sera utilisée pour

trouver la combinaison de métriques (ou catégories de métriques) de couplage la plus pertinente pour expliquer la probabilité de présence de fautes de différents niveaux de sévérité dans les classes.

5.2.3.1 Analyse de régression logistique univariée (RLU)

La RLU est conduite sur les trois systèmes étudiés. Pour rappel, la variable dépendante, à savoir la prédisposition des classes aux fautes sévères, est codée comme suit : 1 pour les classes ayant au moins une faute sévère, et 0 pour les autres. Nous utiliserons les métriques comme variables explicatives.

Le tableau 19 résume les résultats de la RLU pour ANT, AMQ et CAMEL avec les paramètres d'analyse suivants :

- Le **coefficient** donne une idée sur l'impact des métriques sur le modèle : plus il est grand (en valeur absolue), plus fort est l'impact de la métrique dans la détermination de la probabilité de préséance ou pas de faute sévère dans la classe évaluée.
- Le **p-value** est reliée au test statistique et permet de vérifier la significativité du coefficient par rapport au seuil type alpha = 0.05.
- Le **R^2** mesure la proportion de variance totale de la variable dépendante, expliquée par le modèle. Plus sa valeur est élevée (proche de 1), plus fort est l'effet des variables indépendantes (les métriques), et plus précis (ou fidèle) sera le modèle.
- L'**AUC** (area under curve) donne l'aire sous la courbe ROC. Cet indicateur évalue la performance du modèle pour la classification des classes. Elle varie de 0 à 1 (1 correspondant à une classification sans erreur, et un AUC < 0.5 étant pire qu'une classification aléatoire).

D'après les résultats de ANT, seules trois des métriques de couplage (Ce, RFC, et IC) sont significatives pour déterminer la criticité des classes. Comparativement, toutes les métriques de taille ont été trouvées significatives. RFC et Ce ont été les plus significatives parmi toutes les métriques (p-value < 0.0001) et ont les R^2 les plus élevés (0.29 pour RFC et 0.26 pour Ce). Les coefficients de ces dernières (0.72 et 0.6 respectivement) sont aussi les plus élevés.

Pour le système AMQ, toutes les métriques ont été significatives pour la prédisposition des classes aux fautes sévères. IC, CBM et Ca apparaissent comme les moins significatives avec des p-values respectives de 0.04, 0.02 et 0.04. Le coefficient le plus élevé est obtenu avec RFC (0.6), correspondant aussi au R^2 le plus élevé (0.26). RFC est ensuite suivie dans l'ordre (parmi les métriques de couplage) de Ce, MOA et CBO.

Les résultats obtenus avec CAMEL indiquent la métrique RFC et Ce comme largement plus significatives que le reste des métriques de coulage. Toutes les métriques de taille ont un p-value < 0.0001, donc très significatives, mais RFC reste la métrique la plus significative pour la prédisposition aux fautes sévères, avec un R^2 de 0.13 (0.12 pour LOC) et un coefficient de 0.38 (0.33 pour LOC). CBM, IC et Ca ne sont pas significatives.

La Figure 15 donne une illustration permettant de comparer les performances de classification des différents modèles sur les trois systèmes. Les modèles construits avec AMQ atteignent des performances plus élevées. Les modèles obtenus avec ANT surpassent ceux obtenus avec CAMEL, excepté pour les métriques Ca et MOA. La métrique MOA apparaît comme aussi performante que CBO pour AMQ (AUC=0.77), tandis que pour les deux autres systèmes, leur AUC est très faible (0.5 et 0.3). Les tendances restent constantes pour RFC, Ce et CBO. Elles donnent les meilleures performances sur les trois systèmes avec des valeurs de AUC entre 0.76 et 0.81 pour RFC, 0.70 et 0.77 pour Ce et entre 0.68 et 0.77 pour CBO. Les métriques IC, CBM et Ca donnent les pires performances sur les trois systèmes

Tableau 19 : Résultats de RLU.

	Métriques	Ca	CBM	CBO	Ce	IC	LOC	MOA	NPM	RFC	WMC
ANT	Constante	-0,74	-1,11	-0,92	-2,13	-1,17	-1,73	-0,82	-1,5	-2,37	-1,81
	Coefficient	0,02	0,28	0,15	0,6	0,29	0,59	0,09	0,44	0,72	0,57
	p_value	0,82	0,28	0,16	< 0,0001	0,01	0	0,4	0	< 0,0001	0
	R^2	0	0,08	0,02	0,26	0,08	0,22	0,01	0,13	0,29	0,18
	AUC	0,47	0,43	0,71	0,74	0,42	0,75	0,33	0,64	0,78	0,7
ANMQ	Constante	-2,23	-2,21	-2,51	-3,15	-2,21	-2,79	-2,74	-3,23	-3,43	-3,29
	Coefficient	0,08	0,11	0,24	0,52	0,09	0,4	0,47	0,52	0,6	0,55
	p_value	0,04	0,02	< 0,0001	< 0,0001	0,04	< 0,0001	< 0,0001	< 0,0001	< 0,0001	< 0,0001
	R^2	0,01	0,01	0,06	0,23	0,01	0,16	0,19	0,22	0,26	0,24
	AUC	0,66	0,48	0,77	0,77	0,48	0,79	0,77	0,78	0,81	0,83
CAMEL	Constante	-2,84	-2,73	-2,96	-3,46	-2,7	-3,42	-3,07	-3,25	-3,69	-3,36
	Coefficient	0,11	-0,04	0,15	0,3	-0,06	0,33	0,21	0,26	0,38	0,29
	p_value	0,05	0,7	0,01	< 0,0001	0,56	< 0,0001	0	< 0,0001	< 0,0001	< 0,0001
	R^2	0,01	0	0,03	0,08	0	0,12	0,04	0,07	0,13	0,09
	AUC	0,5	0,26	0,68	0,7	0,23	0,72	0,53	0,7	0,76	0,74

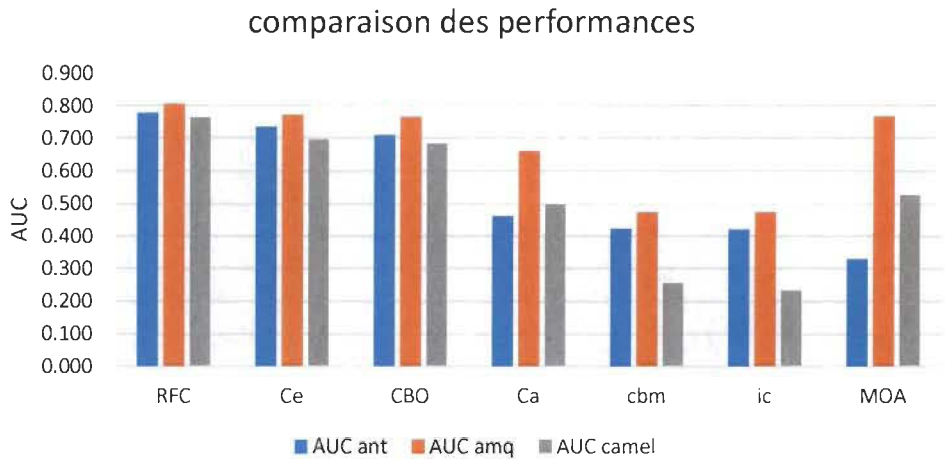


Figure 15 : Performances des modèles RLU.

5.2.3.2 Sélection des métriques pour la prédiction des fautes sévères

Dans ce qui suit, nous allons chercher à déterminer la combinaison de métriques la plus pertinente pour expliquer la présence ou non de fautes critiques dans les classes. Pour cela, nous allons soumettre l'ensemble des métriques de couplage à un algorithme de sélection/réduction de caractéristiques/variables (STEPDISC). Il s'agit d'une sélection pas-à-pas (*Stepwise*) ascendante basée sur la fonction de discriminant. Comme son nom l'indique, la sélection se fait par étape (pas à pas) et à chaque itération la variable la plus pertinente (par rapport au test statistique) est sélectionnée. Un seuil de 0.01 est fixé pour la p-value. La sélection est appliquée sur les métriques, puis sur les axes principaux de l'ACP obtenus par rotation

VARIMAX, et à la fin, sur les Cluster du regroupement selon VARHCA. Le but étant d'évaluer l'importance :

- De chaque métrique dans un contexte incluant les autres métriques,
- De chaque type de couplage dans un contexte incluant les autres métriques,
- Des techniques de construction de caractéristiques pour la prédiction.

Le tableau 20 donne les résultats de la sélection STEPDISC ascendante pour les métriques de couplage.

Avec AMQ, l'algorithme retient quatre métriques (RFC, Ce, CBM et MOA), alors qu'avec Camel et ANT, une seule métrique est sélectionnée (RFC). Les résultats suggèrent donc de façon unanime que RFC est la plus pertinente parmi les métriques de couplage pour déterminer la criticité des classes.

Le tableau 21 donne les résultats de la sélection STEPDISC ascendante pour les clusters obtenus avec VARHCA. Deux clusters sont retenus pour AMQ, un seul est retenu pour ANT et CAMEL. Le cluster le plus significatif pour les trois systèmes est celui lié au couplage sortant. Il représente les métriques RFC et Ce (et MOA pour CAMEL). Le deuxième le plus significatif est celui représentant le couplage par héritage (à travers IC et CBM) pour ANT et AMQ, alors que pour Camel, il s'agit du cluster représentant le couplage entrant (à travers Ca et CBO). Les résultats indiquent donc que le couplage sortant est le plus significatif parmi les types de couplage dans la détermination de la présence de fautes critiques dans les classes.

Le tableau 22 donne les résultats de la sélection STEPDISC ascendante pour les axes principaux de l'ACP (axes obtenus après rotation VARIMAX). Cinq axes sont sélectionnés pour AMQ, deux pour ANT et deux pour Camel. Pour ces deux derniers systèmes, le premier axe sélectionné est celui représentant RFC. Tandis que pour AMQ, l'axe le plus significatif résume 87% de la variance de Ce et 37% de RFC. À la seconde itération, nous avons les métriques Ce (pour ANT) et MOA (pour AMQ) à travers les axes qui les représentent. Pour CAMEL, la deuxième sélection correspond à l'axe représentant le couplage entrant (Ca et CBO). La sélection s'arrête là pour ANT et CAMEL, alors que pour AMQ, nous avons

ensuite (dans l'ordre) : FR1_Axis_5 (résumant 48% de la variance RFC) et FR1_Axis_2 (représentant IC et CBM) et FR1_Axis_6 (axe résiduel).

Tableau 20 : Sélection STEPDISC métriques.

Systèmes	Itération	d.f	Best	Sol.1	Sol.2	Sol.3	Sol.4	Sol.5	Sol.6	Sol.7
AMQ	1	(1, 1071)	**RFC**	**RFC**	**Ce**	**MOA**	**CBO**	**CBM**	**Ca**	**IC**
			L :	L :	L :	L :	L :	L :	L :	L :
			F :	F :	F :	F :	F : 55,11	F : 13,08	F : 4,59	F : 4,45
			p :	p :	p :	p :	p :	p :	p :	p :
	2	(1, 1070)	**Ce**	**Ce**	**MOA**	**CBM**	**IC**	**CBO**	**Ca**	-
			L :	L :	L :	L :	L :	L :	L :	
			F : 17,18	F : 17,18	F : 16,91	F : 16,39	F : 4,28	F : 1,98	F : 0,09	
			p :	p :	p :	p :	p :	p :	p :	
	3	(1, 1069)	**CBM**	**CBM**	**MOA**	**IC**	**CBO**	**Ca**	-	-
			L :	L :	L :	L :	L :	L :		
			F : 20,72	F : 20,72	F : 10,43	F : 7,84	F : 0,42	F : 0,40		
			p :	p :	p :	p :	p :	p :		
	4	(1, 1068)	**MOA**	**MOA**	**IC**	**CBO**	**Ca**	-	-	-
			L :	L :	L :	L :	L :			
			F : 10,21	F : 10,21	F : 1,73	F : 0,21	F : 0,20			
			p :	p :	p :	p :	p :			
	5	(1, 1067)	-	**IC**	**Ca**	**CBO**	-	-	-	-
				L :	L :	L :				
				F : 1,72	F : 0,04	F : 0,02				
				p :	p :	p :				
ANT	1	(1, 124)	**RFC**	**RFC**	**Ce**	**IC**	**CBM**	**CBO**	**MOA**	**Ca**
			L :	L :	L :	L :	L :	L :	L :	L :
			F : 33,26	F : 33,26	F : 30,62	F : 8,59	F : 8,41	F : 2,28	F : 0,72	F : 0,05
			p :	p :	p :	p :	p :	p :	p :	p :
	2	(1, 123)	-	**Ce**	**CBM**	**MOA**	**IC**	**Ca**	**CBO**	-
				L :	L :	L :	L :	L :	L :	
				F : 5,71	F : 3,93	F : 2,55	F : 2,09	F : 1,92	F : 0,83	
				p :	p :	p :	p :	p :	p :	
CAMEL	1	(1, 606)	**RFC**	**RFC**	**Ce**	**CBO**	**MOA**	**Ca**	**IC**	**CBM**
			L :	L :	L :	L :	L :	L :	L :	L :
			F : 50,41	F : 50,41	F : 25,02	F : 13,19	F : 12,29	F : 5,61	F : 0,34	F : 0,15
			p :	p :	p :	p :	p :	p :	p :	p :
	2	(1, 605)	-	**Ca**	**CBO**	**IC**	**CBM**	**MOA**	**Ce**	-
				L :	L :	L :	L :	L :	L :	
				F : 4,93	F : 4,69	F : 2,96	F : 2,21	F : 0,97	F : 0,01	
				p :	p :	p :	p :	p :	p :	

Tableau 21 : Sélection STEPDISC VARHCA.

Systèmes	Itération	d.f	Best	Sol.1	Sol.2	Sol.3	Sol.4
AMQ	1	(1, 1071)	**VCHca_1_2**	**VCHca_1_2**	**VCHca_1_3**	**VCHca_1_1**	-
			L : 0,7813	L : 0,7813	L : 0,9786	L : 0,9916	
			F : 299,75	F : 299,75	F : 23,39	F : 9,12	
			p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0026	
	2	(1, 1070)	**VCHca_1_1**	**VCHca_1_1**	**VCHca_1_3**	-	-
			L : 0,7702	L : 0,7702	L : 0,7813		
			F : 15,47	F : 15,47	F : 0,01		
			p : 0,0001	p : 0,0001	p : 0,9328		
	3	(1, 1069)	-	**VCHca_1_3**	-	-	-
				L : 0,7701			
				F : 0,19			
				p : 0,6634			
ANT	1	(1, 124)	**VCHca_1_3**	**VCHca_1_3**	**VCHca_1_2**	**VCHca_1_1**	**VCHca_1_4**
			L : 0,7538	L : 0,7538	L : 0,9315	L : 0,9939	L : 0,9942
			F : 40,49	F : 40,49	F : 9,11	F : 0,76	F : 0,72
			p : 0,0000	p : 0,0000	p : 0,0031	p : 0,3855	p : 0,3981
	2	(1, 123)	-	**VCHca_1_4**	**VCHca_1_2**	**VCHca_1_1**	-
				L : 0,7193	L : 0,7450	L : 0,7521	
				F : 5,91	F : 1,46	F : 0,29	
				p : 0,0165	p : 0,2289	p : 0,5935	
Camel	1	(1, 606)	**VCHca_1_2**	**VCHca_1_2**	**VCHca_1_1**	**VCHca_1_3**	-
			L : 0,9370	L : 0,9370	L : 0,9851	L : 0,9996	
			F : 40,74	F : 40,74	F : 9,17	F : 0,26	
			p : 0,0000	p : 0,0000	p : 0,0026	p : 0,6072	
	2	(1, 605)	-	**VCHca_1_1**	**VCHca_1_3**	-	-
				L : 0,9290	L : 0,9321		
				F : 5,21	F : 3,22		
				p : 0,0228	p : 0,0734		

Tableau 22 : Sélection STEPDISC ACP.

Systèmes	Itération	d.f	Best	Sol.1	Sol.2	Sol.3	Sol.4	Sol.5
AMQ	1	(1, 1071)	**FR_1_Axis_1**	**FR_1_Axis_1**	**FR_1_Axis_4**	**FR_1_Axis_5**	**FR_1_Axis_2**	**FR_1_Axis_6**
			L : 0,8835	L : 0,8835	L : 0,9417	L : 0,9577	L : 0,9901	L : 0,9952
			F : 141,17	F : 141,17	F : 66,26	F : 47,31	F : 10,76	F : 5,19
			p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0011	p : 0,0229
	2	(1, 1070)	**FR_1_Axis_4**	**FR_1_Axis_4**	**FR_1_Axis_5**	**FR_1_Axis_2**	**FR_1_Axis_6**	**FR_1_Axis_3**
			L : 0,8253	L : 0,8253	L : 0,8412	L : 0,8736	L : 0,8787	L : 0,8792
			F : 75,54	F : 75,54	F : 53,81	F : 12,19	F : 5,88	F : 5,34
			p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0005	p : 0,0155	p : 0,0211
	3	(1, 1069)	**FR_1_Axis_5**	**FR_1_Axis_5**	**FR_1_Axis_2**	**FR_1_Axis_6**	**FR_1_Axis_3**	**FR_1_Axis_7**
			L : 0,7830	L : 0,7830	L : 0,8153	L : 0,8204	L : 0,8209	L : 0,8246
			F : 57,76	F : 57,76	F : 13,04	F : 6,29	F : 5,71	F : 0,84
			p : 0,0000	p : 0,0000	p : 0,0003	p : 0,0123	p : 0,0171	p : 0,3604
	4	(1, 1068)	**FR_1_Axis_2**	**FR_1_Axis_2**	**FR_1_Axis_6**	**FR_1_Axis_3**	**FR_1_Axis_7**	
			L : 0,7730	L : 0,7730	L : 0,7781	L : 0,7786	L : 0,7823	-
			F : 13,75	F : 13,75	F : 6,62	F : 6,01	F : 0,88	
			p : 0,0002	p : 0,0002	p : 0,0102	p : 0,0144	p : 0,3480	
	5	(1, 1067)	**FR_1_Axis_6**	**FR_1_Axis_6**	**FR_1_Axis_3**	**FR_1_Axis_7**		
			L : 0,7682	L : 0,7682	L : 0,7686	L : 0,7724	-	-
			F : 6,70	F : 6,70	F : 6,09	F : 0,89		
			p : 0,0097	p : 0,0097	p : 0,0138	p : 0,3451		
	6	(1, 1066)	-	**FR_1_Axis_3**	**FR_1_Axis_7**			
				L : 0,7638	L : 0,7675	-	-	-
				F : 6,12	F : 0,90			
				p : 0,0135	p : 0,3438			
ANT	1	(1, 124)	**FR_1_Axis_5**	**FR_1_Axis_5**	**FR_1_Axis_3**	**FR_1_Axis_1**	**FR_1_Axis_4**	**FR_1_Axis_2**
			L : 0,8691	L : 0,8691	L : 0,8927	L : 0,9623	L : 0,9976	L : 0,9983
			F : 18,67	F : 18,67	F : 14,91	F : 4,86	F : 0,30	F : 0,21
			p : 0,0000	p : 0,0000	p : 0,0002	p : 0,0294	p : 0,5859	p : 0,6471
	2	(1, 123)	**FR_1_Axis_3**	**FR_1_Axis_3**	**FR_1_Axis_1**	**FR_1_Axis_4**	**FR_1_Axis_2**	**FR_1_Axis_6**
			L : 0,7618	L : 0,7618	L : 0,8315	L : 0,8667	L : 0,8674	L : 1,0000
			F : 17,32	F : 17,32	F : 5,58	F : 0,34	F : 0,24	F : -16,10
			p : 0,0001	p : 0,0001	p : 0,0198	p : 0,5606	p : 0,6248	p : 0,0000
	3	(1, 122)	-	**FR_1_Axis_1**	**FR_1_Axis_4**	**FR_1_Axis_2**	**FR_1_Axis_6**	
				L : 0,7241	L : 0,7594	L : 0,7601	L : 1,0000	-
				F : 6,35	F : 0,39	F : 0,27	F : -29,06	
				p : 0,0130	p : 0,5358	p : 0,6029	p : 0,0000	
CAMEL	1	(1, 606)	**FR_1_Axis_1**	**FR_1_Axis_1**	**FR_1_Axis_2**	**FR_1_Axis_4**	**FR_1_Axis_5**	**FR_1_Axis_3**
			L : 0,9325	L : 0,9325	L : 0,9892	L : 0,9922	L : 0,9982	L : 0,9983
			F : 43,87	F : 43,87	F : 6,63	F : 4,77	F : 1,11	F : 1,03
			p : 0,0000	p : 0,0000	p : 0,0103	p : 0,0293	p : 0,2921	p : 0,3116
	2	(1, 605)	**FR_1_Axis_2**	**FR_1_Axis_2**	**FR_1_Axis_4**	**FR_1_Axis_5**	**FR_1_Axis_3**	**FR_1_Axis_6**
			L : 0,9217	L : 0,9217	L : 0,9247	L : 0,9307	L : 0,9308	L : 1,0000
			F : 7,11	F : 7,11	F : 5,11	F : 1,19	F : 1,10	F : -40,84
			p : 0,0079	p : 0,0079	p : 0,0241	p : 0,2756	p : 0,2951	p : 0,0000
	3	(1, 604)	-	**FR_1_Axis_4**	**FR_1_Axis_5**	**FR_1_Axis_3**	**FR_1_Axis_6**	
				L : 0,9139	L : 0,9198	L : 0,9200	L : 1,0000	-
				F : 5,16	F : 1,20	F : 1,11	F : -47,31	
				p : 0,0234	p : 0,2732	p : 0,2927	p : 0,0000	

5.2.3.3 Analyse de régression multivariée

Nous avons effectué une analyse de régression logistique multivariée sur les trois systèmes avec les différentes métriques de couplage dans le but de voir l'effet combiné de ces dernières sur la prédisposition des classes aux fautes sévères. Les résultats (obtenus avec tanagra) sont résumés dans les tableaux 23 et 24.

La colonne CHI² TEST (LR) donne le test du rapport de vraisemblance pour la significativité globale de la régression (p-value). Nous obtenons une p-value sensiblement nulle avec la loi du KHI^2 à 7 degrés de liberté (le nombre de variables/métriques). Les trois modèles sont donc globalement significatifs au sens alpha = 5%. Les R^2 obtenus avec les systèmes AMQ et ANT sont assez élevés (0.3 et 0.4), mais le taux global d'erreurs de la classification est plus bas pour Camel (6%), suivi par ANT (9%). On note que RFC est la seule métrique significative (p-value < 0.05) sur l'ensemble des trois modèles, ensuite viennent MOA (pour ANT et AMQ) puis CBM (pour ANT) qui sont aussi significatives.

Tableau 23 : Paramètres d'évaluation des MRL.

Systèmes	R^2	Chi-2 test (PL)	error_rate
AMQ	0,31	0(173,8284)	0,09
ANT	0,41	0 (44,3929)	0,25
Camel	0,16	0 (35,9674)	0,06

Tableau 24 : Paramètres RLM.

	Métriques	constante	CBO	RFC	Ca	Ce	MOA	IC	CBM
AMQ	Coef.	-3,58	0,03	0,02	-0,03	0,01	0,15	-0,37	0,81
	Std-dev	0,2	0,09	0	0,09	0,09	0,06	0,9	0,46
	Wald	333,03	0,13	21,08	0,15	0,03	5,74	0,17	3,07
	Signif	0	0,7162	0	0,6964	0,8712	0,0166	0,6806	0,0797
ANT	Coef.	-3,05	0,15	0,04	-0,15	0,07	-0,65	-0,82	0,55
	Std-dev	0,56	0,19	0,01	0,18	0,17	0,26	0,48	0,27
	Wald	29,58	0,64	9,09	0,72	0,17	6,13	2,93	4,22
	Signif	0	0,4253	0,0026	0,3948	0,677	0,0133	0,0867	0,0401
CAMEL	Coef.	-3,77	0,05	0,03	-0,04	-0,04	0,17	-0,54	0,06
	Std-dev	0,32	0,14	0,01	0,14	0,13	0,13	0,53	0,2
	Wald	136,37	0,12	12,74	0,07	0,08	1,81	1,02	0,1
	Signif	0	0,7255	0,0004	0,7847	0,7741	0,1779	0,312	0,7493

5.2.4 Évaluation effective de la capacité des métriques à prédire la sévérité des fautes

Dans ce qui précède, nous avons pu mettre en évidence l'existence de liens entre le couplage et la prédisposition des classes aux fautes sévères. Nous avons pu évaluer la force de ces liens en fonction des types de couplage avec différentes approches. Nous allons maintenant étudier la capacité effective du couplage à distinguer les classes contenant des fautes critiques. Nous allons utiliser un jeu de données pour construire les modèles (échantillon d'apprentissage), puis les évaluer sur un autre jeu de données (échantillon de test). Nous allons utiliser plusieurs méthodes/techniques de prédiction basées sur l'apprentissage automatique en plus de la régression logistique. Les modèles seront construits à la fois avec les métriques de couplage, les clusters de la classification VARHCA et les Axes principaux de l'ACP (rotation VARIMAX). Un algorithme de sélection de variables (STEPDISC) sera appliqué pour choisir la meilleure combinaison de variables (métriques/axes/clusters).

L'approche la plus commune pour évaluer les performances des modèles de prédiction est d'étudier la matrice de confusion. Celle-ci donne la proportion de bons classements et de mauvais classements par rapport à un point de séparation (cut-off) prédéterminé. Cette approche reste donc subjective, car dépendante du choix du cut-off. Ainsi, à la place de chercher à associer les classes à des clusters prédéterminés par un cut-off, nous allons plutôt les classer dans un ordre décroissant selon les probabilités qu'elles ont de présenter au moins une faute sévère. L'approche semble être plus réaliste quand il s'agit d'utiliser les modèles dans un cadre industriel (de production) pour diverses raisons :

- Facilite/ Flexibilise l'utilisation/réutilisation des modèles.
- Permet la comparaison des modèles sur une base commune.

Pour cette expérimentation, nous allons fusionner les données des trois systèmes pour les raisons suivantes :

- Augmenter la taille de l'échantillon de données : échantillon de test et échantillon d'apprentissage.

- Lissage des données / souci de généralisation. Réduire l'influence que peuvent avoir les spécificités de chaque système sur les données.

Plusieurs objectifs sont à la fois visés avec cette démarche :

- Évaluer la capacité effective du couplage à prédire la prédisposition des classes aux fautes sévères. Les modèles sont testés sur des données autres que celles utilisées pour leurs constructions.
- Comparer les techniques de prédiction à la prédisposition des classes aux fautes sévères.
- Comparer les types de représentation des données (Métriques, Clusters et Axes) pour les modèles de prédiction à la prédisposition des classes aux fautes sévères.

5.2.4.1 Modèles basés sur les composantes principales

Le tableau 25 donne une indication sur la signification des axes. Les métriques CE, MOA, et RFC sont représentées chacun sur un axe distinct (respectivement Axis_1, Axis_4 et Axis_5). Les métriques CBM et IC sont résumées sur le même axe (Axis_2), de même que les métriques Ca et CBO (Axis_3).

Quatre axes sont retenus par la sélection STEPDISC (tableau 26) avec un seuil de significativité de 0.01. Nous retrouvons le couplage sortant au premier plan (RFC et Ce et MOA dans l'ordre) à travers leurs axes représentatifs (respectivement Axis_1, Axis_4 et Axis_5). Ensuite, nous avons le couplage par héritage (IC et CBM) dont l'axe représentatif est sélectionné avec un score de 13,33 pour le test de Fisher (p-value=0.0003).

Tableau 25 : Représentation des variables sur les axes principaux.

Métriques	Axis_1		Axis_2		Axis_3		Axis_4		Axis_5	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
Ce	0,89	79 % (79 %)	0,02	0 % (79 %)	0,09	1 % (80 %)	0,28	8 % (87 %)	0,36	13 % (100 %)
CBM	0,00	0 % (0 %)	0,95	90 % (90 %)	-0,02	0 % (90 %)	0,00	0 % (90 %)	0,01	0 % (90 %)
IC	0,03	0 % (0 %)	0,94	89 % (89 %)	-0,02	0 % (89 %)	0,02	0 % (89 %)	0,04	0 % (89 %)
Ca	-0,09	1 % (1 %)	-0,02	0 % (1 %)	0,99	99 % (100 %)	0,05	0 % (100 %)	-0,01	0 % (100 %)
CBO	0,23	5 % (5 %)	-0,01	0 % (5 %)	0,96	92 % (97 %)	0,13	2 % (99 %)	0,11	1 % (100 %)
MOA	0,25	6 % (6 %)	0,01	0 % (6 %)	0,14	2 % (8 %)	0,92	85 % (93 %)	0,26	7 % (100 %)
RFC	0,36	13 % (13 %)	0,04	0 % (13 %)	0,07	1 % (13 %)	0,30	9 % (22 %)	0,88	78 % (100 %)
Var. Expl.	1,04	15 % (15 %)	1,79	26 % (40 %)	1,94	28 % (68 %)	1,03	15 % (83 %)	0,99	14 % (97 %)

Tableau 26 : Sélection STEPDISC des AXES.

Itération	d.f	Best	Sol.1	Sol.2	Sol.3	Sol.4	Sol.5
1	(1, 901)	**FR_2_Axis_5**	**FR_2_Axis_5**	**FR_2_Axis_1**	**FR_2_Axis_4**	**FR_2_Axis_2**	**FR_2_Axis_3**
		L : 0,9364	L : 0,9364	L : 0,9709	L : 0,9807	L : 0,9870	L : 0,9975
		F : 61,23	F : 61,23	F : 27,02	F : 17,75	F : 11,86	F : 2,26
		p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0006	p : 0,1332
2	(1, 900)	**FR_2_Axis_1**	**FR_2_Axis_1**	**FR_2_Axis_4**	**FR_2_Axis_2**	**FR_2_Axis_3**	**FR_2_Axis_7**
		L : 0,9073	L : 0,9073	L : 0,9171	L : 0,9234	L : 0,9339	L : 0,9356
		F : 28,88	F : 28,88	F : 18,96	F : 12,66	F : 2,41	F : 0,73
		p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0004	p : 0,1209	p : 0,3941
3	(1, 899)	**FR_2_Axis_4**	**FR_2_Axis_4**	**FR_2_Axis_2**	**FR_2_Axis_3**	**FR_2_Axis_7**	**FR_2_Axis_6**
		L : 0,8879	L : 0,8879	L : 0,8943	L : 0,9048	L : 0,9065	L : 0,9067
		F : 19,56	F : 19,56	F : 13,06	F : 2,48	F : 0,75	F : 0,57
		p : 0,0000	p : 0,0000	p : 0,0003	p : 0,1153	p : 0,3869	p : 0,4485
4	(1, 898)	**FR_2_Axis_2**	**FR_2_Axis_2**	**FR_2_Axis_3**	**FR_2_Axis_7**	**FR_2_Axis_6**	-
		L : 0,8749	L : 0,8749	L : 0,8854	L : 0,8872	L : 0,8874	
		F : 13,33	F : 13,33	F : 2,54	F : 0,76	F : 0,59	
		p : 0,0003	p : 0,0003	p : 0,1116	p : 0,3820	p : 0,4439	
5	(1, 897)	-	**FR_2_Axis_3**	**FR_2_Axis_7**	**FR_2_Axis_6**	-	-
			L : 0,8724	L : 0,8742	L : 0,8744		
			F : 2,57	F : 0,78	F : 0,59		
			p : 0,1092	p : 0,3788	p : 0,4408		

Les modèles sont construits avec les 4 axes principaux retenus lors de la sélection STEPDISC et les résultats de leurs performances sont présentés sur le tableau 27. Les 7 modèles sont représentés en colonne. Pour chaque modèle, la colonne Target size donne les pourcentages de classes dans l'ordre décroissant de leur score, puis dans la colonne TP-rate (True Positives), nous avons le pourcentage correspondant de classes réellement touchées par au moins une faute sévère (les vrais positifs). Les lignes du tableau correspondant au 10% et 30 % des classes ayant les scores les

plus élevés sont encadrées en rouge, et la ligne pour chaque modèle, correspondant au TP-rate >= 90% est coloriée en orange dans le but de faciliter la lecture.

Pour la cible des 10%, NNP donne les meilleures performances (avec 50% des classes contenant des fautes critiques qui peuvent être découvertes), suivie par RL et RF. NB et RBF sont les moins performants (28% et 22%).

Pour la cible des 30%, les méthodes NNP et RL donnent les meilleures performances (avec 77% de vrais-positifs). RBF et NB restent toujours les moins performantes.

Pour la cible des 90% de vrais positifs, la méthode NB donne le meilleur modèle (avec 100% des TP qui peuvent être détectés en se focalisant sur 40% des classes), suivie par RBF (50% des classes pour 100% de TP), puis par RF (50% de classes pour 92% de TP). Les modèles obtenus avec les méthodes SVM et DT sont les moins performants (60%).

Tableau 27 : Scores modèles ACP.

NB		SVM		RBF		RF		DT		RL		NNP	
Target_size	TP_Rate	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-
0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0,28	5	0,31	5	0	5	0,2604	5	0,219	5	0,34	5	0,30
10	0,28	10	0,41	10	0,22	10	0,4375	10	0,365	10	0,44	10	0,50
15	0,28	15	0,51	15	0,49	15	0,5104	15	0,479	15	0,57	15	0,57
20	0,28	20	0,63	20	0,53	20	0,60	20	0,594	20	0,63	20	0,68
25	0,28	25	0,70	25	0,53	25	0,68	25	0,698	25	0,74	25	0,76
30	0,45	30	0,73	30	0,53	30	0,74	30	0,708	30	0,77	30	0,77
35	0,75	35	0,77	35	0,53	35	0,74	35	0,708	35	0,78	35	0,81
40	1	40	0,82	40	0,65	40	0,84	40	0,802	40	0,81	40	0,83
45	1	45	0,88	45	0,89	45	0,90	45	0,823	45	0,85	45	0,85
50	1	50	0,89	50	1	50	0,93	50	0,823	50	0,88	50	0,88
55	1	55	0,90	55	1	55	0,94	55	0,854	55	0,92	55	0,91
60	1	60	0,93	60	1	60	0,98	60	0,917	60	0,94	60	0,92
65	1	65	0,96	65	1	65	1	65	0,917	65	0,94	65	0,96
70	1	70	0,97	70	1	70	1	70	0,917	70	0,97	70	0,97
75	1	75	0,98	75	1	75	1	75	0,917	75	0,98	75	0,99
80	1	80	0,98	80	1	80	1	80	0,917	80	0,99	80	0,99
85	1	85	0,98	85	1	85	1	85	0,917	85	0,99	85	0,99
90	1	90	0,99	90	1	90	1	90	0,917	90	0,99	90	0,99
95	1	95	0,99	95	1	95	1	95	0,927	95	0,99	95	0,99
100	1	100	1	100	1	100	1	100	1	100	1	100	1

5.2.4.2 Modèles basés sur les clusters

Le tableau 28 donne une indication sur la signification des clusters. Seule la métrique MOA est représentée de façon isolée par un cluster. RFC et Ce sont résumées par le cluster 3, CBM et IC par le cluster 1, et sur le cluster 2 nous avons Ca et CBO.

Deux clusters sont retenus par la sélection STEPDISC (tableau 29) au seuil de significativité de 0.01. Le premier à être sélectionné est le cluster 3, qui représente le couplage sortant via Ce et RFC, ensuite c'est le cluster 1, qui correspond au couplage par héritage via IC et CBM (avec un p-value de 0.001).

Tableau 28 : Corrélations des clusters avec les métriques.

Métriques	# Appartenance	Cluster 1	Cluster 2	Cluster 3	Cluster 4
CBO	1	-0,02	0,98	0,35	0,34
RFC	1	0,07	0,17	0,93	0,61
Ca	1	-0,04	0,98	0,03	0,16
Ce	1	0,05	0,19	0,93	0,58
MOA	1	0,03	0,26	0,64	1,00
IC	1	0,95	-0,03	0,08	0,04
CBM	1	0,95	-0,04	0,04	0,02

Tableau 29 : Sélection STEPDISC Clusters.

itérations	Best	Sol.1	Sol.2
1	**VCHca_2_3**	**VCHca_2_3**	**VCHca_2_4**
	L : 0,8884	L : 0,8884	L : 0,9397
	F : 113,17	F : 113,17	F : 57,86
	p : 0,0000	p : 0,0000	p : 0,0000
2	**VCHca_2_1**	**VCHca_2_1**	**VCHca_2_4**
	L : 0,8778	L : 0,8778	L : 0,8867
	F : 10,86	F : 10,86	F : 1,77
	p : 0,0010	p : 0,0010	p : 0,1831
3		**VCHca_2_4**	**VCHca_2_2**
		L : 0,8760	L : 0,8772
		F : 1,88	F : 0,65
		p : 0,1707	p : 0,4199

Les modèles sont construits avec les 2 clusters retenus lors de la sélection STEPDISC et les résultats de leurs performances sont présentés dans le tableau 30. Pour la cible des 10%, les méthodes NNP, RL et SVM donnent les meilleures performances (avec 43% des classes critiques qui peuvent être découvertes), mais avec un taux de vrais positifs (TP-rate) plus bas que pour les modèles basés sur les axes principaux. DT donne les plus mauvaises des performances (6%), suivie de RF (29%).

Tableau 30 : Scores modèles Cluster.

NB		SVM		RBF		RF		DT		RL		NNP	
Target_size	TP_Rate	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-
0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0,33	5	0,32	5	0	5	0,21	5	0,063	5	0,33	5	0,31
10	0,33	10	0,43	10	0,32	10	0,29	10	0,063	10	0,43	10	0,43
15	0,33	15	0,53	15	0,49	15	0,42	15	0,063	15	0,55	15	0,55
20	0,33	20	0,58	20	0,49	20	0,49	20	0,063	20	0,57	20	0,65
25	0,33	25	0,72	25	0,49	25	0,57	25	0,063	25	0,70	25	0,70
30	0,43	30	0,76	30	0,49	30	0,63	30	0,344	30	0,76	30	0,73
35	0,73	35	0,77	35	0,49	35	0,68	35	0,719	35	0,79	35	0,79
40	0,98	40	0,81	40	0,77	40	0,71	40	0,99	40	0,82	40	0,82
45	1	45	0,86	45	0,97	45	0,74	45	0,99	45	0,88	45	0,89
50	1	50	0,89	50	1	50	0,83	50	0,99	50	0,89	50	0,90
55	1	55	0,91	55	1	55	0,95	55	1	55	0,92	55	0,93
60	1	60	0,93	60	1	60	1	60	1	60	0,93	60	0,94
65	1	65	0,95	65	1	65	1	65	1	65	0,95	65	0,95
70	1	70	0,96	70	1	70	1	70	1	70	0,96	70	0,96
75	1	75	0,97	75	1	75	1	75	1	75	0,97	75	0,97
80	1	80	0,97	80	1	80	1	80	1	80	0,97	80	0,97
85	1	85	1	85	1	85	1	85	1	85	1	85	1
90	1	90	1	90	1	90	1	90	1	90	1	90	1
95	1	95	1	95	1	95	1	95	1	95	1	95	1
100	1	100	1	100	1	100	1	100	1	100	1	100	1

Pour la cible des 30%, les méthodes SVM et RL donnent les meilleures performances (avec 76% de vrais-positifs, ce qui est légèrement plus bas que pour les modèles avec les axes de l'ACP), suivies par NNP (73%). Les modèles obtenus avec les méthodes DT et NB sont les moins performants.

Pour la cible des 90% de vrais positifs, NB et DT donnent les meilleures performances (avec 98% et 99% des TP qui peuvent être détectées en se focalisant sur 40% des classes), suivies par RBF (45% des classes pour 97% de TP). Le reste des modèles atteignent cet objectif avec un taux de 55 % (un taux légèrement inférieur à celui obtenu avec les axes principaux).

5.2.4.3 Modèles basés sur les métriques

Le tableau 31 donne le résultat de la sélection STEPDISC avec un seuil de significativité de 0.01. Seules deux métriques sont retenues (RFC et IC). RFC est sélectionnée en premier, ensuite IC.

Tableau 31 : Sélection STEPDISC métriques.

Itération	Best	Sol.1	Sol.2	Sol.3	Sol.4	Sol.5
1	RFC	RFC	Ce	MOA	CBO	IC
	L : 0,8894	L : 0,8894	L : 0,9178	L : 0,9397	L : 0,9825	L : 0,9828
	F : 112,09	F : 112,09	F : 80,74	F : 57,86	F : 16,05	F : 15,77
	p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0000	p : 0,0001	p : 0,0001
2	IC	IC	CBM	Ce	MOA	CBO
	L : 0,8786	L : 0,8786	L : 0,8814	L : 0,8847	L : 0,8863	L : 0,8879
	F : 11,04	F : 11,04	F : 8,08	F : 4,77	F : 3,15	F : 1,47
	p : 0,0009	p : 0,0009	p : 0,0046	p : 0,0292	p : 0,0764	p : 0,2250
3	-	Ce	MOA	CBO	Ca	CBM
		L : 0,8740	L : 0,8754	L : 0,8768	L : 0,8780	L : 0,8785
		F : 4,72	F : 3,28	F : 1,84	F : 0,59	F : 0,12
		p : 0,0300	p : 0,0705	p : 0,1748	p : 0,4434	p : 0,7280

Tableau 32 : Scores modèles métriques.

NB		SVM		RBF		RF		DT		RL		NNP	
Target_size	TP_Rate	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-	Target	TP-
0	0,00	0	0,00	0	0	0	0	0	0	0	0	0	0
5	0,13	5	0,33	5	0	5	0,22	5	0,20	5	0,33	5	0,33
10	0,42	10	0,45	10	0,31	10	0,33	10	0,20	10	0,46	10	0,45
15	0,42	15	0,57	15	0,52	15	0,43	15	0,26	15	0,56	15	0,55
20	0,42	20	0,61	20	0,52	20	0,53	20	0,51	20	0,64	20	0,64
25	0,42	25	0,68	25	0,52	25	0,58	25	0,70	25	0,70	25	0,70
30	0,42	30	0,71	30	0,52	30	0,63	30	0,70	30	0,73	30	0,73
35	0,66	35	0,78	35	0,52	35	0,66	35	0,70	35	0,76	35	0,78
40	0,88	40	0,80	40	0,76	40	0,70	40	0,70	40	0,79	40	0,79
45	1	45	0,83	45	0,96	45	0,82	45	0,95	45	0,85	45	0,86
50	1	50	0,85	50	1	50	0,91	50	1	50	0,89	50	0,89
55	1	55	0,89	55	1	55	0,94	55	1	55	0,92	55	0,92
60	1	60	0,91	60	1	60	1	60	1	60	0,93	60	0,93
65	1	65	0,92	65	1	65	1	65	1	65	0,96	65	0,97
70	1	70	0,92	70	1	70	1	70	1	70	0,98	70	0,98
75	1	75	0,92	75	1	75	1	75	1	75	0,99	75	0,99
80	1	80	0,93	80	1	80	1	80	1	80	0,99	80	0,99
85	1	85	0,93	85	1	85	1	85	1	85	0,99	85	0,99
90	1	90	0,95	90	1	90	1	90	1	90	0,99	90	0,99
95	1	95	0,95	95	1	95	1	95	1	95	1	95	1
100	1	100	1	100	1	100	1	100	1	100	1	100	1

Le tableau 32 présente les performances des modèles construits avec les 2 métriques retenues lors de la sélection STEPDISC (RFC et IC).

Pour la cible des 10%, RL, NNP et SVM arrivent en première position avec des performances respectives de 46%, 45% et 45% de TP. DT apparait comme la moins performante avec seulement 20% des classes contenant des fautes sévères qui peuvent être découvertes.

Pour la cible des 30%, NNP et RL donnent les meilleures performances avec 73% de TP. NB apparaît comme le moins performant avec seulement 42% de TP.
Pour la cible des 90% de vrais positifs, les méthodes NB, RBF et DT donnent les meilleures performances. En nous focalisant sur 45% des classes, nous avons 100% des TP qui peuvent être détectés par NB, 96% par RBF et 95% pour DT. Il vient juste après RBF (50% des classes pour 91% de TP). Le modèle SVM apparait comme le moins performant (60% de classes pour 90% de TP).

Chapitre 6: Menaces à la validité

Les résultats obtenus à travers cette étude sont assez concluants. Cependant, il y a un certain nombre de considérations qu'il faudra évoquer en ce qui concerne ses limitations et sa validité.

6.1 Validité interne

La collecte de données sur les fautes avec BuginfoV2 est faite de façon empirique. Malgré les améliorations apportées, l'utilisation (dans un processus automatique) des commentaires des *commits* ne garantit pas à coup sûr la détection de patch pour une faute donnée. En plus, l'assignation d'une faute à une classe est faite sans prendre en considération le fait qu'il puisse s'agir d'une répercussion de la correction d'une faute. En effet, aucune distinction n'est faite entre les classes effectivement fautives et celles impactées par la correction d'une faute. De ce fait, il peut arriver que certaines classes, probablement celles ayant un fort couplage sortant, puissent être considérées comme atteintes par des fautes sévères, alors qu'elles ne sont pas en réalité à l'origine de ces fautes. Sans une analyse manuelle des patchs, il est extrêmement difficile dans un processus automatique de distinguer le(s) classe(s) à l'origine de la faute de celles impactées par effet de propagation.

La définition de la sévérité (ainsi que sa catégorisation) considérée dans notre étude est à relativiser. En effet, il ne s'agit pas d'une définition objective mesurée ou calculée, mais plutôt d'une information que nous avons reprise telle qu'elle est donnée dans les ITS.

Par ailleurs, le choix de fusionner les données de systèmes différents pourrait être source de questionnements. Mais cette approche, utilisée dans d'autres études, permet en même temps de réduire le biais de nos résultats pour les systèmes étudiés. En plus, les métriques concernées ont toutes été mesurées par le même outil (CKJM).

6.2 Validité externe

Dans ce travail, nous avons mené des expérimentations empiriques sur trois systèmes orientés objet écrits en JAVA. Par conséquent, toutes les conclusions qui découlent de nos résultats restent limitées. Nous ne pourrons pas les généraliser pour le reste des langages orientés objet, bien que le langage JAVA soit un bon représentant de ces langages. Pour cela, il faudrait conduire d'autres études basées sur d'autres langages OO (comme C#) et/ou d'autres systèmes de divers domaines.

6.3 Validité conceptuelle

Pour représenter le couplage, nous avons utilisé une implémentation des métriques de CK et QMOOD avec l'outil CKJM. Ce dernier se base sur le code JAVA compilé pour effectuer les mesures. Il existe donc un grand nombre de mesures du couplage qui n'ont pas été considérées dans notre étude, notamment le couplage capturé par les métriques dynamiques et le couplage induit par le caractère réflexif de certains langages OO (l'introspection).

Chapitre 7: Conclusion et perspectives

L'apparition de défaillances dans les systèmes logiciels après leur mise en production est un fait très courant. Certaines de ces défaillances ont pour cause la présence de fautes dans le code source dont la correction peut s'avérer très couteuse. Ainsi, la découverte et la correction de ces fautes, du moins les plus sévères, dès les premières phases du processus logiciel constituent une des principales préoccupations de l'assurance qualité. Dans ce sens, l'approche basée sur les métriques offre un cadre pratique pour appliquer des mesures préventives et correctives (priorisation des tests, *refactoring*) à différents niveaux du processus.

Cette étude a porté sur la nature des relations (potentielles) entre le couplage et la sévérité des fautes dans les systèmes orientés objet. Nous avons identifié différentes formes de couplage ainsi que la manière dont elles pourraient être liées à la présence de fautes sévères dans les classes logicielles. Nous avons, par la suite, évalué la force de ces liaisons. Pour mener notre étude, nous avons commencé par recueillir des données sur plusieurs systèmes logiciels développés en JAVA. Nous avons établi une procédure automatisée de collecte de données sur les classes contenant des fautes sévères, procédure que nous avons implémentée dans l'outil BuginfoV2. Pour évaluer le couplage, nous avons utilisé les métriques de couplage calculées avec l'outil CKJM. À partir des données recueillies sur 3 logiciels open source (AMQ, ANT, CAMEL), nous avons effectué plusieurs expérimentations, en utilisant différentes techniques de classification (RLU, RMU, DT, RBF, SVM, NNP, NB, RF), d'analyse statistique standard (analyse de corrélations, statistiques descriptives), de regroupement de variables (VARKMeans, VARHCA), de sélection de variables (STEPDISC), et d'analyse factorielle (ACP). Les résultats obtenus montrent l'existence de 3 formes de couplage : (1) le couplage sortant capturé à travers les métriques RFC, Ce et MOA, (2) le couplage entrant capturé par les métriques Ca et CBO, et le couplage via l'héritage capturé à travers les métriques IC et CBM.

Le couplage sortant s'est révélé être le plus significativement lié à la présence de fautes critiques dans les classes, notamment le couplage sortant via l'appel de méthodes représenté par RFC. Nous avons trouvé que le couplage entrant n'avait pas d'influence significative sur la présence de fautes sévères dans les classes. Le couplage par héritage a montré une faible influence positive. L'utilisation de la technique de sélection de variables a montré que le couplage sortant et le couplage par héritage constituaient les meilleures combinaisons de métriques de couplage pour construire des modèles de prédiction de qualité, capables de prédire un taux important de classes critiques.

Nous avons montré que les méthodes d'apprentissage automatique pouvaient être aussi performantes (voir plus), que la régression logistique plus largement utilisée dans le domaine. Nous avons aussi démontré l'efficacité de l'utilisation des techniques d'extraction de caractéristiques (ACP, Clustering) pour la construction des modèles de prédiction. Les modèles construits à partir des composants principaux ou des clusters ont atteint des performances comparables à celles obtenues directement à partir des métriques.

Comme suite à notre étude, il serait intéressant de synthétiser un indice de criticité des classes basé sur le couplage et d'autres métriques comme la complexité, car le fait que le couplage sortant (par invocation de méthode) apparaisse comme le plus significativement lié à la sévérité des fautes montre de façon implicite l'importance que peut avoir la complexité dans le même sens. L'utilisation de ces deux attributs dans une seule dimension pourrait s'avérer très pertinente pour identifier les parties les plus critiques d'un système OO. Comme première piste, on pourrait adopter une approche basée sur les graphes d'appels et l'algorithme de classement de page de Google (Page Rank).

Chapitre 8: Références

[Abreu 94] F.B.E. Abreu and R. Carapuca, Object-oriented software engineering: measuring and controlling the development process, in: Proceedings of the 4th International Conference on Software Quality, 1994.

[Abreu 96] F.B.E. Abreu and W. Melo, Evaluating the impact of object oriented design on software quality, in: Proceedings of the 3rd International Software Metrics Symposium, pp. 90–99, 1996.

[Aggarwa 07] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, Investigating effect of design metrics on fault proneness in object-oriented systems, Journal of Object Technology, 6, 127–141, 2007.

[Aggarwal 09] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study, Software Process Improvement and Practice, 14, 39–62, 2009.

[Ambros 11] M. D'Ambros, M. Lanza and R. Robbes, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empirical Software Engineering, 1–47, 2011.

[Ball 97] T. Ball, J.M. Kim, A.A. Porter and H.P. Siy, If Your Version Control System Could Talk, Proc. ICSE Workshop Process Modelling and Empirical Studies of Software Eng., 1997.

[Bansiya 02] J. Bansiya, C.G. Davis, A hierarchical model for object-oriented design quality assessment, IEEE Transactions on Software Engineering, 28, 4–17, 2008

[Basili 96] V.R. Basili, L.C. Briand and W.L. Melo, A validation of object-oriented design metrics as quality indicators, IEEE Transactions on Software Engineering, 22, 751–761,1996.

[Bieman 95] J.M. Bieman and B.K. Kang, Cohesion and reuse in an object-oriented system, ACM SIGSOFT Software Engineering Notes 20, 259–262, 1995.

[Breiman 01] L. Breiman, Random Forests, Machine Learning, 2001, vol. 45, no. 1, pp. 5-32, 2001.

[Briand 00] L.C. Briand, J. Wüst, J.W. Daly and D. Victor Porter, Exploring the relationships between design measures and software quality in object-oriented systems, Journal of Systems and Software, vol. 51, no. 1, pp. 245–273, 2000.

[Briand 01] L. Briand, J. Wüst and H. Lounis, Replicated Case Studies for

Investigating Quality Factors in Object-Oriented Designs, Empirical Software Engineering: An International Journal, vol 6, no 1, 11-58, 2001.

[Briand 02] L.C. Briand and J. Wust, Empirical studies of quality models in object-oriented systems, in: Advances in Computers, vol. 56, pp. 97–166, 2002.

[Briand 97] L. Briand, P. Devanbu and W. Melo, An investigation into coupling measures for C++, in: International Conference on Software Engineering, Association for Computing Machinery, pp. 412–421, 1997

[Briand 98] L.C. Briand, J. Daly, V. Porter ans J. Wust, Predicting fault-prone classes with design measures in object-oriented systems, in: Proceedings of the International Symposiumon Software Reliability Engineering (ISSRE 1998), pp. 334–343,1998.

[Briand 99] L.C. Briand and J.W. Daly, A unified framework for coupling measurement in object-oriented systems, IEEE Transactions on Software Engineering, vol. 25, no. 1, pp. 91–121, 1999.

[Cartwrigth 00] M. Cartwright, M. Shepperd, An empirical investigation of an object-oriented software system, IEEE Transactions on Software Engineering, 26, 786–796, 2000.

[Catal 11] C. Catal, Software fault prediction: a literature review and current trends, Expert Systems with Applications, 38, 4626–4636 2011.

[Chauhan 12] A.S. Chauhan and S.K. Dubey, Analytical Review of Fault-proneness for Object Oriented Systems, International Journal of Scientific and Engineering Research, Volume 3, Issue 12, December 2012.

[Chidamber 91] S.R. Chidamber and C.F. Kemerer, Towards a metrics suite for object oriented design, in: Oopsla 91 Conference Proceedings: Object-Oriented Programming Systems, Languages, and Applications, ACM, 1991.

[Chidamber 94] S. R. Chidamber and C. F. Kemerer, Metrics suite for object oriented design, IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476–493, 1994.

[Chidamber 94] S.R. Chidamber and C.F. Kemerer, A metrics suite for object-oriented design, IEEE Transactions on Software Engineering, 20, 476–493, 1994.

[Chowdhury 11] I. Chowdhury and M. Zulkernine, Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities, Journal of Systems Architecture, 57, 294–313,2011.

[Cortes 95] C. Cortes, V. Vapnik., Support-vector networks, Machine

Learning, 20 (3): 273–297,1995.

[Eder 94] J. Eder, G. Kappel and M. Schrefl, Coupling and Cohesion in Object-Oriented Systems, Technical Report, Univ. of Klagenfurt, 1994.

[Elish 11] M.O. Elish, A.H. Al-Yafei, M. Al-Mulhem, Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems: a case study of Eclipse, Advances in Engineering Software, 42, 852–859, 2011.

[Emam 01] K. El Emam, S. Benlarbi, N. Goel and S.N. Rai, The confounding effect of class size on the validity of object-oriented metrics, IEEE Transactions on Software Engineering, vol. 27, no. 7, pp. 630–650, 2001.

[Etzkorn 99] L. Etzkorn, J. Bansiya and C. Davis, Design and code complexity metrics for OO classes, Journal of Object-Oriented Programming, 12, 35–40, 1999.

[Gyimothy 05] T.Gyimothy, R. Ferenc and I. Siket, Empirical validation of object-oriented metrics on open source software for fault prediction, IEEE Trans. Software Engineering, vol. 31, Issue 10, 897 – 910, Oct. 2005.

[Halstead 77] M.H. Halstead, Elements of Software Science, Elsevier Science Inc., New York, NY, USA, 1977.

[Haykin 04] S. Haykin, Neural Networks: A comprehensive Foundation, Second Edition, Pearson Education 2004.

[Henderson 96] B. Henderson-Sellers, Software Metrics, Prentice-Hall, Hemel Hempstaed, UK, 1996.

[Kpodjedo 11] S. Kpodjedo, F. Ricca, P. Galinier, Y.G. Gueheneuc and G. Antoniol, Design evolution metrics for defect prediction in object oriented systems, Empirical Software Engineering, 16, 141–175, 2011.

[Lee 95] Y.S. Lee, B.S. Liang, S.F. Wu and F.J. Wang, Measuring the coupling and cohesion of an object-oriented program based on information flow, in: Proc. International Conference on Software Quality, Maribor, Slovenia, pp. 81–90, 1995.

[Li 93a] W. Li and S. Henry, Maintenance metrics for the object oriented paradigm, in: Proceedings of First International Software Metrics Symposium, pp. 52–60, 1993.

[Li 93b] W. Li and S. Henry, Object-oriented metrics that predict maintainability, Journal of Systems and Software, 23, 111–122, 1993.

[Li 98] W. Li, Another metric suite for object-oriented programming, Journal of Systems and Software, 44, 155–162, 1998.

[Lorenz 94] M. Lorenz and J. Kidd, Object-Oriented Software Metrics, Prentice Hall, Englewood Cliffs, NJ, 1994.

[Malhotra 10] R. Malhotra, A. Kaur and Y. Singh, Empirical validation of object oriented metrics for predicting fault proneness at different severity levels using support vector machines, International Journal of System Assurance Engineering and Management, Volume 1, Issue 3, pp 269-281, September 2010.

[Malhotra 15] R. Malhotra, Empirical Research in Software Engineering: Concepts, Analysis, and Applications, Boca Raton, FL: CRC Press, 2016.

[McCabe 76] T.J. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering, SE-2, 308–320, 1976.

[Montazeri 95] M. Hitz, B. Montazeri, Measuring coupling and cohesion in object-oriented systems, in: Proceedings of the International Symposium on Applied Corporate Computing, vol. 50, pp. 75–76, 1995.

[Olague 07] H. Olague, L. Etzkorn, S. Gholston and S. Quattlebaum, Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes, IEEE Transactions on Software Engineering, 33(8), pp. 402-419, 2007.

[Oyetoyan 13] T.D. Oyetoyan, D.S. Cruzes and R. Conradi, Criticality of Defects in Cyclic Dependent Components. IEEE 13th International Working Conference on source Code Analysis and Manipulation (SCAM), pp. 21-30, 2013.

[Pai 07] G. Pai and B. Dugan, Empirical analysis of software fault content and fault proneness using Bayesian methods, IEEE Transactions on Software Engineering, 33 (2007), 675–686.

[Platt 98] J. Platt. Sequetial minimal optimization: A fast algorithm for training support vector machines, In Technical Report MST-TR-98-14, Microsoft Research, 1998.

[Quinlan 93] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[Radjenović 13] D. Radjenović, M. Heričko, R. Torkar and A. Živkovič, Software fault prediction metrics, Information and Software Technology, v.55 n.8, p.1397-1418, August 2013.

[Rathore 12] S.S. Rathore and A. Gupta, Investigating object-oriented design metrics to predict fault-proneness of software modules, Sixth

International Conference on Software Engineering (CONSEG), CSI, 1-10, 2012.

[Russell 95] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.

[Shatnawi 08] R. Shatnawi and W. Li, The effectiveness of software metrics in identifying error prone classes in post-release software evolution process, Journal of Systems and Software, Vol.81, pp.1868-1882, 2008.

[Sherrod 03] P. Sherrod, DTreg Predictive Modeling Software, 2003.

[Singh 10] Y. Singh, A. Kaur and R. Malhotra, Empirical validation of object oriented metrics for predicting fault proneness models, Software Quality Journal,18(1), 3-35, 2010.

[Tang 99] M.H. Tang, M.H. Kao and M.H. Chen, An empirical study on object-oriented metrics, in Proceedings of the 6th International Software Metrics Symposium, pp. 242–249, November 1999.

[Tegarden 95] D.P. Tegarden, S.D. Sheetz and D.E. Monarchi, A software complexity model of object-oriented systems, Decision Support Systems, 13, 241–26, 1995.

[Vigneau 03] E. Vigneau and E. Qannari, Clustering of variables around latent components, in Communication in Statistics - Simulation and Computation 32(4):1131-1150, January 2003.

[Yu 02] P. Yu, T. Systa and H. Muller, Predicting fault-proneness using OO metrics: An industrial case study, In Proceedings of Sixth European Conference on Software Maintenance and Reengineering, Budapest, Hungary, pp. 99-107, 2002.

[Zhou 06] Y. Zhou and H. Leung, Empirical analysis of object-oriented design metrics for predicting high severity faults, IEEE Transactions Software Engineering, 32 (10), pp. 771-784, 2006.