**1. Introduction**

Les métriques présentées dans cet article sont des métriques de test orientées objet. Le but de cet article n'est pas de mentionner toutes les métriques existantes ou de présenter entièrement les métriques mentionnées, mais de sensibiliser le lecteur à leur existence et d'offrir des références pour des lectures complémentaires. L'objectif de ce travail est de présenter une large revue de la littérature existante sur les mesures de test OO, L'objectif des métriques examinées proposées pour les systèmes OO est de se concentrer sur les métriques de produit pouvant être appliquées à une conception avancée ou au code. Plusieurs auteurs ont formulé leurs suggestions qui doivent être prises en compte lors de la définition des métriques pour les logiciels. Les métriques doivent être définies en poursuivant un objectif clair et leur calcul doit être facile, il est préférable que leur extraction soit automatisée par un outil. L'objectif de ce travail est de fournir aux chercheurs un aperçu de l'état actuel des métriques pour la programmation orientée objet, en mettant l'accent sur les forces et les faiblesses de chaque proposition existante. Ainsi, les chercheurs peuvent avoir une vue d'ensemble approfondie du travail déjà accompli et de celui qui reste à réaliser dans le domaine des métriques de test pour la programmation orientée objet.

**2. Metrics de Test Object-Oriented**

Nous allons maintenant présenter les propositions de métriques sélectionnées pour considération et qui peuvent mieux illustrer le contexte actuel des métriques de test pour la programmation orientée objet.

**2.1 Nombre de méthodes surchargées (NORM).**

Elle permet de calculer le nombre de méthodes

surchargées, pour une classe donnée.

**Méthode de calcul:**

.c: une classe,

.SURc: l’ensemble des méthodes surchargées de c.

La valeur de la métrique NORM pour la classe c sera donnée par la cardinalité de l’ensemble SURc.

Ce calcul est défini par l’équation suivante :

**Impact :**

- Un NORM élevé peut signaler une complexité accrue et une maintenance difficile, tandis qu'un NORM bas peut indiquer une conception plus simple mais potentiellement moins flexible.

**2.2 Nombre de méthodes héritées (NMI).**

Cette métrique mesure le nombre de méthodes héritées par une sous-classe. Aucune mention n'est faite quant à savoir si cette héritage est public ou privé. Dans un langage tel que C++, nous devons considérer la possibilité que l'héritage soit privé. Ainsi, toute classe utilisant des méthodes d'une sous-classe n'aurait pas nécessairement accès à toutes les méthodes héritées.

**Méthode de calcul:**

.c: une classe, C: superclasse de c.

.METc: l’ensemble des méthodes de c.

.METC: l’ensemble des méthodes de C.

La valeur de la métrique NMI pour la classe c sera donnée par la cardinalité de l’intersection de l’ensemble METc et METC

Ce calcul est défini par l’équation suivante :

**Impact :**

-Un NMI élevé peut indiquer une forte dépendance sur les méthodes héritées, ce qui peut rendre le code plus complexe et difficile à maintenir (encapssulation fragile dans le cas d’heritage privee).

-Un NMI élevé peut également indiquer une bonne réutilisabilité du code, car les méthodes héritées peuvent être utilisées dans plusieurs parties du programme.

**2.3 Nombre de méthodes ajoutées à l'héritage (NMA).**

Une méthode est définie comme une méthode ajoutée dans une sous-classe s'il n'existe aucune méthode du même nom dans aucune de ses superclasses.

**Méthode de calcul:**

.c: une classe, C: superclasse de c.

.METc: l’ensemble des méthodes de c.

.METC: l’ensemble des méthodes de C.

La valeur de la métrique NMA pour la classe c sera donnée par la cardinalité de la difference entre METc et l’intersection de l’ensemble METc et METC

Ce calcul est défini par l’équation suivante :

**Impact :**

- Un NMI élevé peut contribuer à une meilleure spécialisation et à une extension des fonctionnalités, mais il peut également augmenter la complexité et les risques associés à la modification et à la maintenance du code.

**2.4 Indice de spécialisation (SIX). [Lorenz and Kidd, 1994]**

L'indice de spécialisation mesure dans quelle mesure les sous-classes remplacent le comportement de leurs superclasses.

.c: une classe, H(c): la position de c dans l’hiérarchie

d’heritage.

.METc: l’ensemble des méthodes de c.

.NORM(c): nombre de methode surhcage de c.

Il est défini par l’équation suivante :

**Impact :**

-Le SIX peut indiquer qu'il y a un excès de méthodes surchargées, ce qui pourrait signifier que l'abstraction initiale n'était peut-être pas appropriée. (qualité de la sous-classification)

-L&K suggèrent une valeur de 15% pour aider à identifier les classes pour lesquelles une action corrective pourrait être appropriée.

**2.5 Number of attributes**

**2.6 Number of static attributes**

**2.7 Number of public attributes**

**2.8 Number of methods**

**2.9 Number of static method.10 Number of classes**

**2.11 Method Inheritance Factor (MIF)**

**2.1Number of Public Methods Defined (PMd)**

**2.13 Number of Public Methods Inherited (PMi)**

**2.14Public Methods Ratio (PMR)**

**2.15 Number of Hidden Methods Defined (HMd)**

**2.16 Number Method Hiding Factor (MHF)of Hidden Methods Inherited (HMi)**

**2.17 Number of Methods Inherited Ratio (NMIR)**

**2.18 Coupling Factor (CF)**

**2.19 Polymorphism Factor (PF)**

**2.20 Number of Attributes Inherited Total (Ait)**

**2.21 Number of Attributes Inherited (Ai)**

**2.22 Number of Attributes Defined (Ad)**

**2.23 Number of Attributes Overidden (Ao)**

**2.24 Attribute Inheritance Factor (AIF)**

**2.25 Method Hiding Factor (AHF)**

**2.26 McCabe Cyclomatic Complexity (VG)**

**2.27 Weighed Methods per Class (WMC)**

**2.28 Lack of Cohesion Methods (LCOM)**

**2.29 Number of Interfaces (NOI)**

**2.30 Afferent Coupling (Ca)**

**2.31 Efferent Coupling**

**2.32 Instability (I)**

**2.33 Abstractness (A)**

**2.34 Normalized Distance from Main Sequence (DMS)**

**2.35 Nested Block Depth (NBD)**

**2.36 Number of Children (NOCh)**

**2.37 Number of Parents (NOPa)**

**2.38 Number of Descendants (NOD**

**2.39 Number of Ancestors (NOA)**

**2.40 Number of Links (NOL)**

**3. Conclusions et travaux futurs.**

Le document présente une suite métrique de base pour le Test orientée objet. Les données métriques offrent une rétroaction rapide aux concepteurs de logiciels et aux gestionnaires. L'analyse et la collecte de ces données peuvent prédire la qualité de la conception. Si elles sont utilisées de manière appropriée, elles peuvent conduire à une réduction significative des coûts de mise en œuvre globale et à des améliorations de la qualité du produit final. La qualité améliorée, à son tour, réduit les efforts de maintenance future. Utiliser des indicateurs de qualité précoce basés sur des preuves empiriques objectives est donc un objectif réaliste.

À l'avenir, nous concevrons un nouvel ensemble de métriques permettant de Tester nos programmes.

***References***