Metriques

Qu'est ce que la métrique ?

- La métrique est un élément de mesure. Elle est basée sur une théorie ensembliste et permet d'associer une pondération à un ensemble de données.

- Les approches sont multiples et variées. Elles peuvent être purement théorique ou davantage basées sur un savoir-faire empirique mais efficace.

.

Métriques de qualité du logiciel (1/2)

- Mesure
  - indication quantitative de l'étendue, quantité, dimension, capacité ou taille d'un attribut de produit ou processus
- Mesurement
  - acte de détermination d'une mesure
- Métrique (IEEE Std. 610.12)
  - mesure quantitative du degré avec lequel un système, composant, ou processus possède un attribut donné
  - peut lier des mesures individuelles
- Indicateur
  - métrique(s) donnant une indication sur le processus, projet, ou produit

Métriques de qualité du logiciel (2/2)

- Les métriques servent à quantifier :
  - l'architecture d'un système
  - à évaluer la productivité
  - à évaluer le temps de réalisation d'un projet
  - à évaluer la maintenabilité d'un système
  - à évaluer le coût d'un projet
  - à analyser les méthodologies

Objectifs des métriques de qualité du logiciel

- 1. Faciliter le contrôle de la gestion, la planification et l'intervention du gestionnaire.
  - Basé sur:
    - Déviations de l'actuel avec l'exécution planifiée
    - Déviations du planning et de la performance réelle de ce qui est planifié

2. Identifier les situations pour le développement ou l'amélioration du processus de maintenance (actions préventives ou correctives).

   Basé sur:

   - Accumulation de métriques concernant la performance des équipes, unités, etc…

Métrique logiciel

- C'est une compilation de mesures issues des propriétés techniques ou fonctionnelles d'un logiciel.
- Il est possible de classer les métriques logicielles en trois catégories :
  - Maintenance applicative
  - Qualité applicative
  - Respect des processus de développement
- Elles se composent toujours de mesures dites « de base »

Mesure de la qualité

- Les processus
  - Ce sont des séries d'activités reliées au développement du logiciel
- Les produits
  - Ce sont tous les objets produits, livrables ou documents qui résultent d'une activité d'un processus
- Les ressources
  - Ce sont des entités exigées par une activité d'un processus

Aspects mesurables

- Chaque entité des trois classes produits, processus et ressources possède
- Des attributs internes : attributs mesurables sur l'entité indépendamment de son environnement
- Des attributs externes : attributs mesurables par rapport aux liens avec son environnement

Exemple

- Attributs internes de processus : durée du processus ou d'une activité, effort mis en œuvre dans le processus ou dans une de ses activités,
- Attributs externes de produit : l'efficacité, la portabilité, la facilité de compréhension, …
- Attributs internes de produit : taille, complexité, couplage, cohésion, …
- Attributs internes de ressource : personnel, matériels, méthodes, …

Quelques remarques

- Les attributs internes de produits sont souvent utilisés pour prédire les attributs externes
- Ces prédictions permettent de contrôler le développement
- Il est très difficile de définir objectivement des mesures qui dépendent de beaucoup d'autres mesures

Les indicateurs les plus adaptables pour la mesure de la qualité sont définis par :

- Le nombre de défauts détectés dans un projet
- Le temps moyen pour éliminer un défaut dans un projet
- Le nombre de défauts par ligne de code
- Le nombre de lignes de documentation du code.
- Le pourcentage du code inspecté par le testeur logiciel.

Métriques de base

Les métriques se divisent en deux catégories :

- les métrique traditionnelles

- Métriques orienté objets

Metriques traditionnelles

Les métriques traditionnelles se divisent en deux groupes :

- métriques mesurant la taille et la complexité : KLOC, **Halstead**

- métriques mesurant la structure du logiciel : **McCabe**

Les métriques traditionnelles

Les métriques de taille sont utilisées pour obtenir la taille des produits logiciels générés dans les phases de spécification, de conception et de construction.

Les mesures établies dans les phases préliminaires pemettant de prédire l'effort exigé pour la production du logiciel.

Exemple

- Nombre de lignes de code
- Nombre de ligne de commentaire
- Nombre de lignes vides

Les métriques traditionnelles

La longueur des fonctions devrait être de 4 à 40 lignes de programme. Une définition de fonction contient au moins un prototype, une ligne de code, et une paire d'accolades, qui font 4 lignes.

En règle générale, une fonction plus grande que 40 lignes de programme doit pouvoir s'écrire en plusieurs fonctions plus simples. A toutes règles son exception, les fonctions contenant un état de sélection avec beaucoup de branches ne peuvent pas être décomposées en fonctions plus petites sans réduire la lisibilité.

La longueur d'un fichier devrait contenir entre 4 et 400 lignes de programme, ce qui équivaut déjà à un fichier possédant entre 10 et 40 fonctions. Un fichier de 4 lignes de programme correspond à une seule fonction de 4 lignes, c'est la plus petite entité qui peut raisonnablement occuper un fichier source complet.

Un fichier de plus de 400 lignes de programme est généralement trop long pour être compris en totalité.

Les métriques traditionnelles

Pour aider à sa compréhension, on estime qu'au minimum 30 % et maximum 75 % d'un fichier devrait être commenté.

Si moins d'un tiers du fichier est commenté, le fichier est soit très trivial, soit pauvrement expliqué. Si plus de 75% du fichier est commenté, le fichier n'est plus un programme, mais un document.

Seul un fichier « header » déroge à cette règle car lorsqu'il est correctement commenté, le pourcentage de commentaires peut parfois dépasser 75%.

Le nombre cyclomatique de Mc Cabe v(G)

La complexité Cyclomatique (complexité de McCabe), introduite par Thomas McCabe en 1976, est le calcul le plus largement répandu des métriques statiques. Conçue dans le but d'être indépendante du langage, la métrique de McCabe indique le nombre de chemins linéaires indépendants dans un module de programme et représente finalement la complexité des flux de donnés.

Il correspond au nombre de branches conditionnelles dans l'organigramme d'un programme.

Le nombre cyclomatique évalue le nombre de chemins d'exécution dans la fonction et ainsi donne une indication sur l'effort nécessaire pour les tests du logiciel.

Pour un programme qui consiste en seulement des états séquentiels, la valeur pour v(G) est 1.

Métriques de Mc Cabe

- Mac Cabe étudier le logiciel en analysant le **graphe de contrôle** du programme et calcule la **complexité** structurelle ou **nombre cyclomatique** de ce graphe
- Soit
  - n = Nombre de noeuds (blocs d'instructions séquentielles)
  - e = Nombre d'arcs (branches suivies par le programme)
  - v = nombre cyclomatique

Métriques de Mc Cabe

- Le nombre cyclomatique donne une évaluation du **nombre des chemins indépendants** dans le graphe et donc une indication sur le **nombre de tests nécessaires**
- Cette métrique indique la borne supérieure du nombre de tests à effectuer pour que tous les arcs soient couverts au moins une fois.

Métriques de Mc Cabe

- **Calcul du nombre cyclomatique:**
  - Cas n° 1: 1 point d'entrée; 1 point de sortie
    - v= e-n +2
  - Cas n° 2 i points d'entrée; s points de sortie
    - v = e - n + i + s
- Rappel
  - v = nombre cyclomatique
  - n = Nombre de noeuds
  - e = Nombre d'arcs

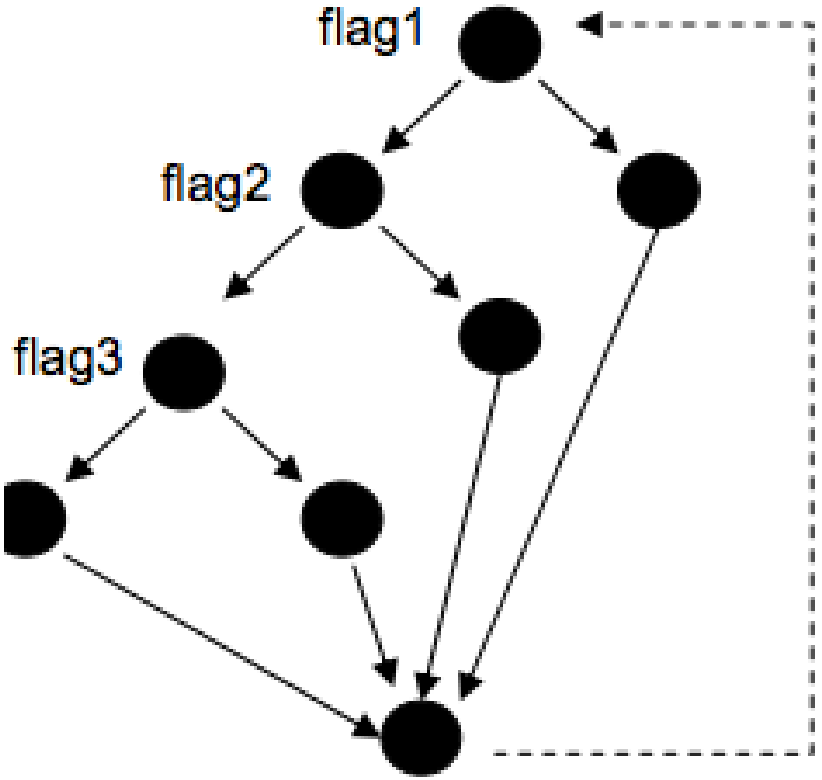
Exemple

Soit le programme suivant :

```
If(flag1=1 & flag2=0){
    if (flag3=1)
                traitement 3;
    else
                traitement 2;
}
Else
Traitement 1;
```

Exemple

Le graphe correspondant est



calcul

E=10, n=8,

Donc v= e-n +2 = 4

Ou

V = π + 1 avec π le nombre de predicats de branchements conditionnel (dans le cas d'un cas(switch) traitant N cas est de N-1)

Generalisation

La formule dans le cas general est :

$V = e - n + 2p$

- n : noeux
- e: aretes
- p: composantes connexes

Métriques de Mc Cabe

- Dans la pratique il semble que la limite supérieure du nombre cyclomatique soit de 30 environ.
- La **valeur maximum du nombre cyclomatique** peut être définie comme un **critère de qualité** dans le plan qualité.
- Rq : Le 'selon'(switch) peut donner un nombre cyclomatiquecatastrophique avec une compréhension fort simple du code !!!

Métriques de Mc Cabe

Exercice : soit le programme « recherche dichotomique » en langage C:

```
void recherche_dico (elem cle, elem t[], int taille, boolean &trouv, int &A)
{ int d, g, m;
      g=0; d=taille -1;
      A (d+g) /2;
      if (t[A]= =cle) trouv=true;
      else trouv=false;
      while (g <=d && !trouv)
            { m= (d+g) /2;
                  if (t[m]= =cle)
                  {           trouv=true;
                              A=m;
                  }
                  else if (t[m]> cle) g=m+1;
                              else d=m-1;
            }
}
```

Calculer le nombre cyclomatique de cette procédure.

Métriques d'Halstead

- Complexité liée à la distribution des variables et instructions.
- Métrique textuelle pour évaluer la taille d'un programme.
- Alternative au calcul du nombre de lignes de code source.
- Calcul a posteriori qui ne peut donc en aucun cas supplanter COCOMO ou la méthode des points de fonction.

- Objectif : calculer l'effort de programmation

Métriques d'Halstead

- La base des mesures est fournie par le vocabulaire utilisé. On évalue le nombre d'opérateurs et d'opérandes.
  - n1 = nombre **d'opérateurs** uniques
  - n2 = nombre **d'opérandes** uniques (termes, constantes , variables)
  - N1 = nombre total d'apparition de ces opérateurs
  - N2 = nombre total d'apparition de ces opérandes.

- Exemple :
- a := a + 1;
  - n1=3 opérateurs ➔ + := ;
  - n2=2 opérandes ➔ a 1

Métriques d'Halstead

- **Mise en œuvre** : « Une fois que le code a été écrit, cette mesure peut être appliquée pour prédire la difficulté d'un programme et d'autres quantités, en employant les équations de Halstead :

- n= n1 + n2 taille du Vocabulaire du programme
- N = N1 + N2 la longueur du programme

- Nous avons alors les mesures suivantes dites de Healstead

- V = N*log2(n) : le volume d'une fonction doit etre entre 20 et 1000. le volume d'un fichier doit etre entre 100 et 8000

- D = (n1 /n2) * (N2 / n2) : la difficulté du programme

- L = 1 / D : le niveau du programme. Un programme au bas niveau est plus enclin aux erreurs qu'un programme de haut niveau

Métriques d'Halstead

- Effort a l'implementation est proportionnelle au volume V et au niveau de difficulté D
  - E= V * D
  -
- Le temps pour implémenter un programme en secondes est
  - T = E / 18
- Le nombre de bugs fournis est
  - $B = (E^{(2/3)}) / 3000$

Exemple

```
Z = 20;
Y = -2;
X = 5;

While  X>0

  Z = Z + Y;

 if  Z > 0 then
    X = X – 1;
  end-if;

End-while;

Print(Z);
```

OPERATOR	COUNT	OPERAND	COUNT
IF–Then– end if	1	Z	5
While End–While	1	Y	2
=	5	X	4
;	8	20	1
>	2	–2	1
+	1	5	1
–	1	0	2
print	1	1	1
( )	1		

$n_1 = 9$ $N_1 = 21$ Length: $N = 21+17 = 38$

$n_2 = 8$ $N_2 = 17$ Volume: $V = 38 \log_2(17)=155$

Indice de maintenabilite

MIwoc = (171 – 5,2*log2(V) – 0,23*(CC) – 16,2*log2(loc))*100/171

Le poids des commentaires :

MIcw = 50*sin(sqrt(2,4*perCM)

Indice de maintenabilite MI = MIwoc + MIcw

Ou :

- V est le volume du code source(moyenne par module) (voir les mesures Halstead)
- CC est la complexite cyclomatique du code(moyenne par module)
- perCM = cLoc/loc ou cLoc est le nombre de lignes de commentaires et loc est le nombre de ligne de code

On admet la regle suivante :

Si MI >= 85 alors bonne maintenabilite

Si 65 =<MI <85 alors maintenabilite moyenne

Si MI < 65 alors maintenabilite difficile

Métriques d'Halstead

- Calculez les mesures de Halstead pour le pseudo-code suivant :

```
read x , y , z;
type=''scalène'';
i f ( x==y or x==z or y==z ) type="isocèle";
i f ( x==y and x==z ) type="équilatéral";
i f ( x>=y+z or y>x+z or z>=x+z ) type="pas un triangle";
i f ( x<=0 or y<=0 or z<=0) type="données erronées";
print type;
```

METRIQUES O.O : ORIENTE OBJETS

Études des métriques OO

Les métriques de Moose proposées par Chidamber et Kemerer en 1994 (6 métriques) :

1) Nombre pondère de méthodes par classe

Soit C une classe avec n méthodes et supposons que la complexité d'une méthode Mi est ci alors :

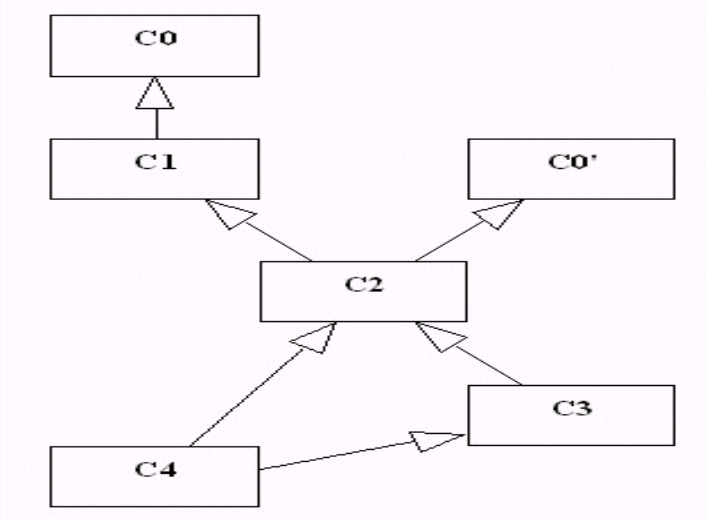
Weighted Methods per Class : cette indice doit etre bas

$$WMC = \sum_{i=1}^{n} ci$$

Métriques de Chidamber et Kamerer

2) Profondeur de l'arbre de l'heritage ou DIT : distance entre la classe et la classe racine

Depth of Inheritance Tree of a Class

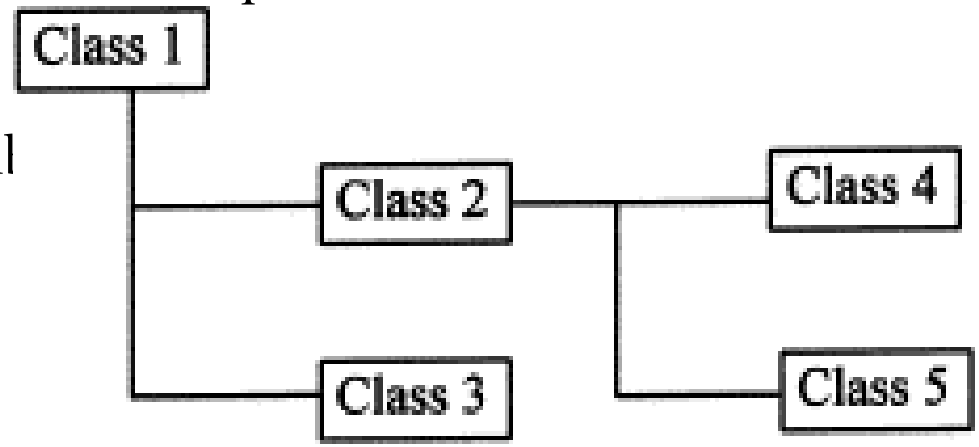


Métriques de Chidamber et Kamerer

3) Nombre de fils dans l'arbre d'heritage (nombre de sous-classes immediates) NOC

Plus le nombre d'enfants est elevee plus la reutilisation effective est importante et necessitera un effort de test.

Numl



Métriques de Chidamber et Kamerer

4) Couplage entre les objets. Un objet est coulpe a un autre objet si le permier invoque des methodes du 2 ou exploite les variables d'instance du 2 . Soit K ensemble des classes de l'application. Pour une classe Ci dans K, CBOci = |Co| ou Co = { C | C K\{Ci}, couple(Ci,C)} avec couple(Ci,C) conforme a la definition donnee.

Plus le CBO d'une classe est grand plus l'effort de test est eleve.

Un couplage excessif entre classes se fera au detrimet de la modularite et empechera la reutilisation

Métriques de Chidamber et Kamerer

5) RFC : cardinalite de l'ensemble reponse d'une classe qui est l'ensemble des methodes qui peuvent etre directement appelees lors de l'execution de n'importe quelle methode de cette classe(=reponse a un message)

Si ce nombre est elevee le debogage et le test de cette classe devient alors difficile.

Métriques de Chidamber et Kamerer

6) LCOM: Lack of cohesion in methods

Il y a LCOM1 : le nomre de pairs de methodes qui n'accedent pas aux memex attributs

LCOM2 : le nombre de pairs de methodes qui n'accedent pas aux memes attributs dimunier de ceux qui accedent (0 si negatif)

Une forte cohesion est signe d'une bonne encapsulation.

Une faible cohesion indique qu'une classe doit etre divise en 2 ou plusieurs et peut etre des erreurs dans le processus de developpement.

INTERPRETATION DES MESURES

Indice de specialisation

L'indice de spécialisation d'une classe est défini par la formule suivante :

$$\frac{NORM \times DIT}{NOM}$$

Avec :

NORM : Number of Overriden Methods, le nombre de méthodes redéfinies.

DIT : Depth in Inheritance Tree, la profondeur dans l'arbre d'héritage.

NOM : Number Of Methods, le nombre de méthodes

Indice de specialisation: Interpretation

- Cet indice augmente quand :

* le nombre de méthodes redéfinies augmente,

* la profondeur d'héritage augmente.

- Il diminue quand :

* le nombre de méthodes spécifiques à la classe augmente,

* le nombre de méthodes redéfinies diminue.

Cet indice peut-être considéré comme trop élevé lorsqu'il est supérieur à 1.5. Il faut alors penser à refactoriser en utilisant des interfaces par exemples.

La moyenne est de 0.05.

.

Indice d'instabilite

- L'indice d'instabilité d'un paquetage est défini par la formule suivante :

Ce/(Ca+Ce)

Avec :

Ca : Couplage afferent, le nombre de classes en dehors du paquetage qui dépendent de classes de ce paquetage(responsabilite).

Ce : Couplage efferent, le nombre de classes de ce paquetage qui dépendent de classes en dehors de ce paquetage(independance).

Indice d'instabilite: interpretation

-Cet indice est toujours compris entre 0 et 1. Un score proche de 0 signifie que le paquetage peut être considéré comme "stable", alors qu'un score proche de 1 indique un paquetage potentiellement à risque.

- Cet indice va faire ressortir les paquetages qui dépendent plus des autres que les autres ne dépendent d'eux.
- Ces paquetages peuvent être dangereux, puisqu'une modification dans un des paquetages dont ils dépendent impacte potentiellement leur fonctionnement.
- lorsqu'un paquetage instable est détecté, il faut alors considérer un autre indicateur : c'est la "**distance from the main sequence**".

Niveau abstraction

-Le niveau d'abstraction d'un paquetage est simplement le rapport entre le nombre d'interfaces et le nombre total de types de ce paquetage :

$$I/T$$

Avec :

I : Le nombre d'interfaces et de classes abstraites du paquetage.

T : Le nombre total de types de ce paquetage..

L'indice d'abstraction est toujours compris entre 0 et 1.

- Comme pour l'indice d'instabilité, le niveau d'abstraction dépend du paquetage considéré.
- Certains paquetages ne nécessitent pas un niveau d'abstraction élevé. Cet indicateur n'a pas grande valeur (hormis une valeur indicative) lorsqu'il est utilisé seul. Il faut l'utiliser conjointement avec l'indice d'instabilité grâce à la "distance from the main sequence »

Distance From the main sequence

-La distance from the main sequence est définie par la formule suivante :

$$|\text{Abstractness} + \text{Instability} - 1|$$

Avec :

Abstractness : Le niveau d'abstraction.

Instability : L'indice d'instabilité.

- Cet indicateur est toujours compris entre 0 et 1. Il représente en fait l'équilibre qui doit résider entre le niveau d'abstraction et l'indice d'instabilité.
- la valeur est proche de 0, plus le design du paquetage est considéré comme bon. En effet, cette valeur va tendre vers 0 lorsque :
  - le paquetage est instable mais possède peu d'interfaces,
  - beaucoup d'autres paquetages dépendent de ce paquetage, mais celui-ci possède beaucoup d'interfaces.
- Une valeur au delà de 0,5 indique qu'il faut penser à refactoriser le paquetage.

Distance From the main sequence

-La distance from the main sequence est définie par la formule suivante :

$$|\text{Abstractness} + \text{Instability} - 1|$$

Avec :

Abstractness : Le niveau d'abstraction.

Instability : L'indice d'instabilité.

- Cet indicateur est toujours compris entre 0 et 1. Il représente en fait l'équilibre qui doit résider entre le niveau d'abstraction et l'indice d'instabilité.
- la valeur est proche de 0, plus le design du paquetage est considéré comme bon. En effet, cette valeur va tendre vers 0 lorsque :
  - le paquetage est instable mais possède peu d'interfaces,
  - beaucoup d'autres paquetages dépendent de ce paquetage, mais celui-ci possède beaucoup d'interfaces.
- Une valeur au delà de 0,5 indique qu'il faut penser à refactoriser le paquetage.

Code Coverage

ou couverture de code

- C'est le Pourcentage de chemins couvert par les tests.
- A 100%, le nombre de tests unitaires d'une méthode est égal à son indice de complexite cyclomatique.

Metriques de MOOD

Ensemble de metriques pour mesuer les attributs des proprietes suivantes :

- Encapsulation
- Heritage
- Couplage
- Polymorphisme

Facteurs Encapsulation

– **MHF** : Method Hiding Factor (10-30%)

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)}$$

avec

– $M_d(C_i)$ le nombre de methodes decla
– $M_h(C_i)$ le nombre de methodes cachees
– TC le nombre total de classes.

– **AHF** : Attribute Hiding Factor (70-100%)

$$\frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$

avec

– $A_d(C_i)$ le nombre d'attributs declares dans une classe C_i
– $A_h(C_i)$ le nombre d'attributs caches

Facteurs Heritage

- **MIF** : Method Inheritance Factor (65-80%)

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

avec

– $M_i(C_i)$ le nombre de methodes heritees (et non surchargees) de C_i

– $M_a(C_i)$ le nombre de methodes qui peuvent etre appelees depuis la classe i

– **AIF** : Attribute Inheritance Factor (50-60%)

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

avec

– $A_i(C_i)$ le nombre d'attributs herites de C_i

– $A_a(C_i)$ le nombre d'attributs auxquels C_i peut acceder

Facteurs de couplage

CF : Coupling Factor (5-30%)

– Mesure le couplage entre les classes sans prendre en compte celui du a l'heritage

$$CF = \frac{\sum_{i=1}^{TC}\sum_{j=1}^{TC} client(C_i, C_j)}{TC^2 - TC}$$

avec

– client(C_i,C_j) = 1 si la classe i a une relation avec la classe j, et 0 sinon

– Les relations d'héritage ne sont pas prises en compte dans les relations

Facteurs polymorphisme

PF : Polymorphism Factor (3-10%)

– Mesure le potentiel de polymorphisme d'un système

$$PF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} M_n(C_i) \times DC(C_i)}$$

avec

– $M_o(C_i)$ le nombre de méthodes surchargées dans la classe i

– $M_n(C_i)$ le nombre de nouvelles méthodes dans la classe i

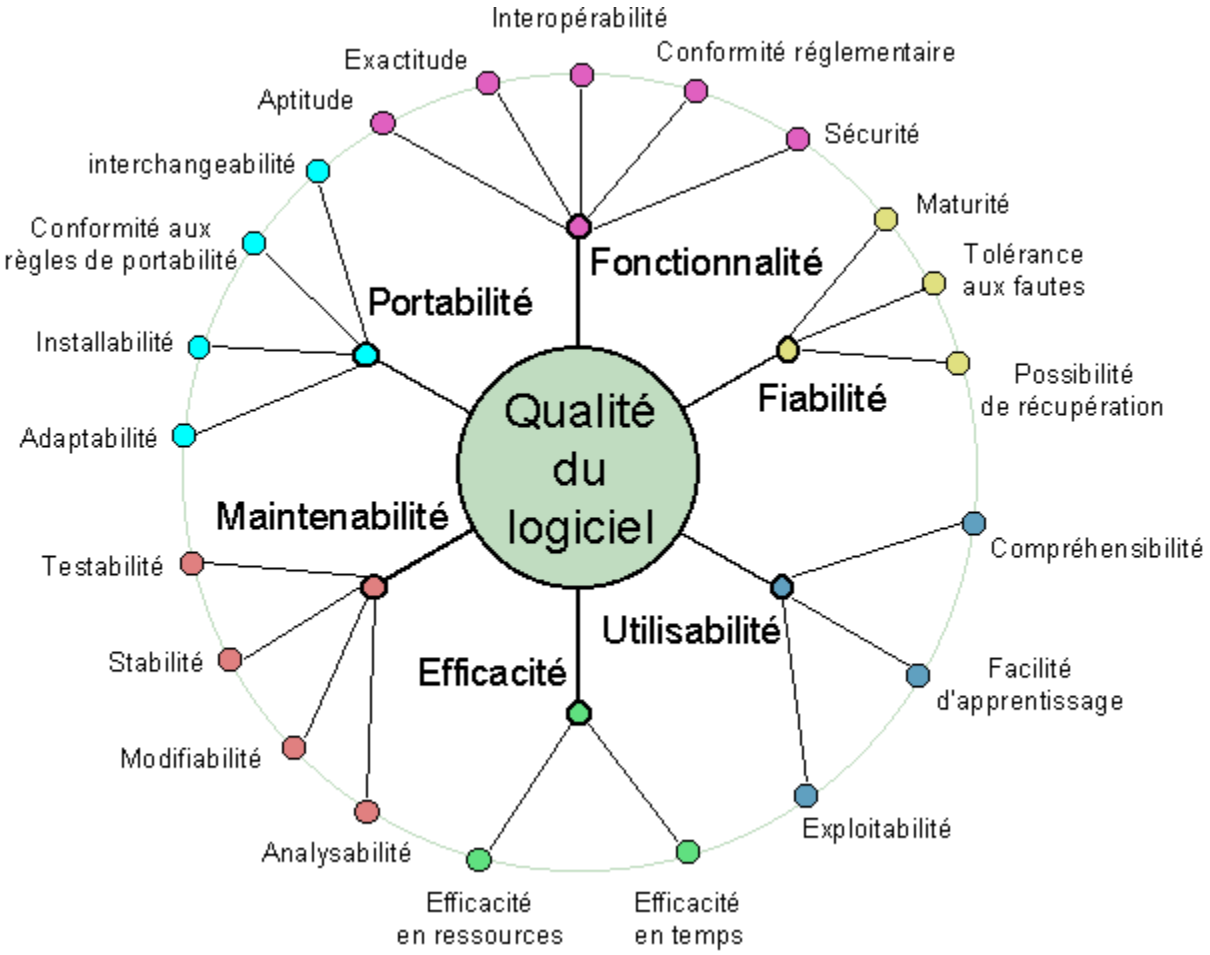
– $DC(C_i)$ le nombre de descendants de la classe i

Outils de métriques du code

- JDepend
- Eclipse Metrics
- PMD
- Sonar
- Et aussi en phase expérimental
  - EvalMetrics : outil développer au laboratoire LSII (ENSAO).

Classification des métriques selon le modèle ISO en utilisant des propriétés de qualité et de design

Modèle ISO 9126



Classification des métriques selon le modèle ISO en utilisant des propriétés de qualité et de design

Les propriétés utilisées dans notre classification :

taille du code,

abstraction,

encapsulation,

modularité,

couplage,

cohésion,

composition,

héritage,

polymorphisme,

complexité

…

Exemple du sous caractéristique (maintenabilité) du modèle ISO selon notre classification