Métriques logicielles

Mesurer et évaluer quantitativement un logiciel

Charbel Daoud - IMT Mines Alès

Quelques rappels sur ce qu'est un logiciel

Un logiciel est un produit : (définition à la manière d'une grammaire algébrique)

- **Particulier** ⇒ s'appuie sur du physique pour de l'**immatériel**
- **Immatériel** ⇒ le rend **puissant** et **vulnérable**
- **Puissant** ⇒ Configurable et adaptable
- **Vulnérable** ⇒ Dépendant d'un environnement d'exécution
- Point faible majeur ⇒ L'HUMAIN !

Qualité logicielle

Définition : Appréciation globale du logiciel par rapport à un ensemble de caractéristiques définies par une ou plusieurs normes.

Selon l'AFNOR : Aptitude d'un produit ou d'un service à satisfaire les besoins des utilisateurs

Selon l'IEEE :

- Le degré avec lequel un système, un composant ou un processus satisfait à ses exigences spécifiées.
- Le degré avec lequel un système, un composant ou un processus satisfait aux besoins ou attentes de ses clients/usagers

Qualité logicielle

Le référentielle ISO 25010 définit 8 indicateurs de qualité logicielle :

- la capacité fonctionnelle : est-ce que le logiciel répond aux besoins fonctionnels exprimés ?
- l'utilisabilité : est-ce que le logiciel nécessite peu d'effort à l'utilisation ?
- la fiabilité : est-ce que le logiciel maintient son niveau de service dans des conditions précises et pendant une période déterminée ?
- efficience de la performance : est-ce que le logiciel requiert un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences ?
- la maintenabilité : est-ce que le logiciel nécessite peu d'effort à son évolution par rapport aux nouveaux besoins ?
- la portabilité : est-ce que le logiciel peut être transféré d'une plate-forme ou d'un environnement à un autre ?
- la sécurité : est-ce que le logiciel est en capacité de protéger ses informations et celles qu'il manipule ?
- la compatibilité : est-ce que le logiciel peut communiquer correctement avec d'autres composants dans des contextes matériels ou logiciels partagés ?

Pourquoi faire de la qualité logicielle ?

Therac-25, 5 morts officiels par syndrome d'irradiation aiguë, plusieurs blessés graves par irradiations massives

Illustration provenant du site hackaday.com

```
Magnet:
   Set bending magnet flag
   repeat
      Set next magnet
      Call Ptime
      if mode/energy has changed, then exit
   until all magnets are set
   return
```

Pourquoi faire de la qualité logicielle ?

Vol inaugural d'Ariane 5 VA-501 : 370 millions de dollars (pour un feu d'artifice…)

Pourquoi faire de la qualité logicielle ?

Système de gestion des paies de l'armée Française nommé Louvois : Milliers de soldes non versées ou mal versées. Diverses répercussions financières et sociales sur plusieurs milliers de personnes

Que retenir de ces échecs ?

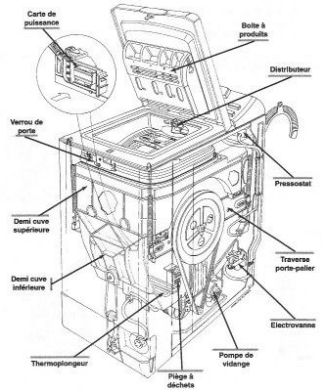
Le bug peut avoir des conséquences qui dépassent de très loin son périmètre fonctionnel et l'endroit virtuel/matériel où le logiciel s'exécute.

Ces conséquences peuvent être :

- Financières
- Sociales
- Humaines
- Matérielles

Maîtriser la qualité par l'évaluation quantitative

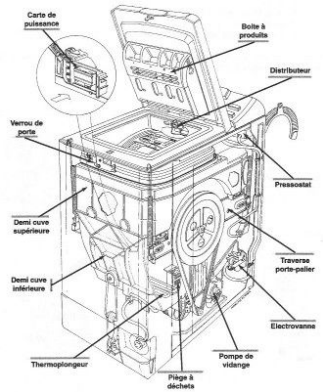
Un objet physique, sans protection contre le *reverse engineering* ou la réparation, a l'ensemble de ses composants observables



Un logiciel, contexte boîte blanche ou sources ouvertes, a l'ensemble de ses lignes de code observables

Maîtriser la qualité par l'évaluation quantitative

Un objet physique, sans protection contre le *reverse engineering* ou la réparation, a l'ensemble de ses composants observables



Un logiciel, contexte boîte blanche ou sources ouvertes, a l'ensemble de ses lignes de code observables

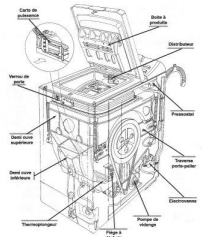
Dans les 2 cas il est possible d'évaluer ces deux objets par des mesures / métriques

Science de la mesure appelée métrologie

Maîtriser la qualité par l'évaluation quantitative

Définition : une métrique est une mesure permettant de quantifier une propriété/un aspect d'un logiciel

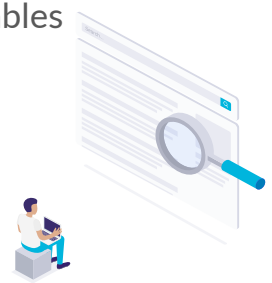
Un objet physique, sans protection contre le *reverse engineering* ou la réparation, a l'ensemble de ses composants observables



Dans les 2 cas il est possible d'évaluer ces deux objets par des mesures / métriques

Science de la mesure appelée métrologie

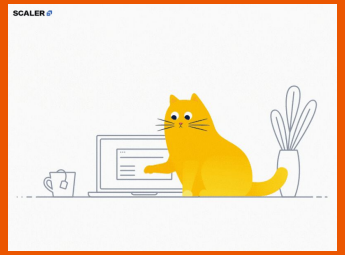
Un logiciel, contexte boîte blanche ou sources ouvertes, a l'ensemble de ses lignes de code observables



Types de métriques

Dans le domaine logiciel il y a 2 grandes familles de métriques :

- les métriques liées au code (*code metrics*) :
  - métriques de complexité
  - métriques dédiées à la POO
  - métriques de maintenabilité
- les métriques liées au processus de développement (*process metrics*) :
  - nombre de développeurs d'un projet
  - âge du projet
  - nombre de *commits*
  - nombre de version
  - nombre de construction sur plate-forme CI réussies
  - etc..



Métriques liées au code : les métriques de complexité

Métriques liées au code : les métriques de complexité

- 1976 : Thomas McCabe invente la première métrique logicielle, la complexité cyclomatique. Elle mesure le nombre de chemin linéairement indépendants dans une portion de code en utilisant le graphe de flot de contrôle (bloc conditionnel, méthode, une classe, etc…). **Toujours utilisée dans l'industrie.**

Métriques liées au code : les métriques de complexité

- 1977 : Maurice Howard Halstead invente une métrique permettant de mesurer 6 aspects du programme. **Utilisé par Microsoft pour calculer un indice de maintenabilité dans Visual Studio**)

  Le calcul de cet indice est basé sur des jetons (éléments textuels du code) qui représentent les opérandes et les opérateurs.

  - le volume du programme : estimation du nombre de bits nécessaire pour coder le programme
  - le niveau de difficulté : la difficulté dépend des opérateurs et des opérandes utilisées pour coder le programme
  - le niveau du programme : l'inverse du niveau de difficulté
  - l'effort à l'implémentation : volume du programme multiplié par la difficulté
  - le temps d'implémentation : effort divisé par un facteur 18
  - estimation du nombre de bugs potentiels : effort d'implémentation divisé par l'habileté du développeur

Métriques liées au code : les métriques de complexité

Le calcul de ces métriques est basé sur des éléments textuels du code qui représentent les opérandes et les opérateurs.

- ***n1*** = le nombre d'opérateurs uniques
- ***n2*** = nombre d'opérandes uniques (termes, constantes, variables)
- ***N1*** = nombre total d'apparition des opérateurs
- ***N2*** = nombre total d'apparition des opérandes

On peut ainsi définir :

- le vocabulaire du programme : **n = n1 + n2**
- la taille du programme : **N = N1 + N2**
- le volume du programme : **V = N * log2(n)**
- le niveau de difficulté : **D = (N1/2) * (N2/n2)**
- le niveau du programme : **L = 1/D**
- l'effort d'implémentation : **E = V * D**
- le temps d'implémentation : **T = E/18**
- le nombre de bugs : **B = (E**(2/3)) / S**
  - **S** = facteur d'habilité du développeur

Métriques liées au code : les métriques de complexité

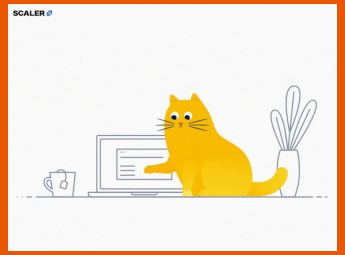
Exemple tiré de Wikipedia sur le calcul de la métrique de Halstead

```
main()
{
  int a, b, c, avg;
  scanf("%d %d %d", &a, &b, &c);
  avg = (a+b+c)/3;
  printf("avg = %d", avg);
}
```

The distinct operators (η_1) are: `main` , `()` , `{}` , `int` , `scanf` , `&` , `=` , `+` , `/` , `printf` , `,` , `;`

The distinct operands (η_2) are: `a` , `b` , `c` , `avg` , `"%d %d %d"` , `3` , `"avg = %d"`

- $\eta_1 = 12, \eta_2 = 7, \eta = 19$
- $N_1 = 27, N_2 = 15, N = 42$
- Calculated Estimated Program Length: $\hat{N} = 12 \times log_2 12 + 7 \times log_2 7 = 62.67$
- Volume: $V = 42 \times log_2 19 = 178.4$
- Difficulty: $D = \frac{12}{2} \times \frac{15}{7} = 12.85$
- Effort: $E = 12.85 \times 178.4 = 2292.44$
- Time required to program: $T = \frac{2292.44}{18} = 127.357$ seconds
- Number of delivered bugs: $B = \frac{2292.44^{\frac{2}{3}}}{3000} = 0.05$



Métriques liées au code : les métriques dédiées à la POO

Les métriques dédiées à la POO : métriques de CK

- Années 90 ⇒ Avènement de la programmation orientée objet (POO)
- 1994 : Chidamber et Kemerer inventent une suite de métriques spécifiquement conçus pour la POO. Ces métriques sont appelées métriques de CK ou *CK-metrics*. Ces métriques évaluent 5 éléments sur le programme que nous allons détailler par la suite. **Elles sont encore aujourd'hui des références pour la mesure de la qualité en POO.**

Les métriques dédiées à la POO : métriques de CK

- 1994 : Les métriques de CK évaluent 5 éléments sur le programme :
    - le nombre d'enfants pour une classe (*Number Of Children (NOC)*) : nombre de sous-classes dépendant immédiatement d'une classe mère donnée. Un nombre de sous-classes élevé peut indiquer un problème de conception ou de performance.
    - le couplage entre classes (*Coupling Between Object (CBO)*) : mesure la dépendance entre les classes du logiciel en s'appuyant sur les attributs, les paramètres et les retours de méthodes. Un couplage élevé sera en défaveur d'une bonne maintenabilité et augmentera la complexité de réutilisabilité.
    - le manque de cohésion des méthodes (*Lack Of Cohesion Of Methods (LOC)*) : évaluer dans quelle mesure les méthodes d'une classe sont liées les unes aux autres. une cohésion élevée est à privilégier car elle favorise l'encapsulation et donc une meilleure réutilisabilité.

Les métriques dédiées à la POO : métriques de CK

- le poids des méthodes par classe (*Weighted Methods per Class (WMC)*) : somme de la complexité (cyclomatique ou nombre de lignes de code) des méthodes appartenant à une classe. Permet d'avoir un ordre de grandeur sur la complexité globale de la classe, sa compréhension et l'effort exigé pour la maintenir.

- la profondeur d'héritage (*Depth Inheritance Tree (DIT)*) : distance entre la classe racine dans l'arbre d'héritage et un noeud héritant de cette dernière. Permet de mesurer une complexité dans l'arboresence d'héritage et la diffiiculté à comprendre le comportement lié à des héritages multiples.

Les métriques dédiées à la POO : métriques de Abreu *etal.*

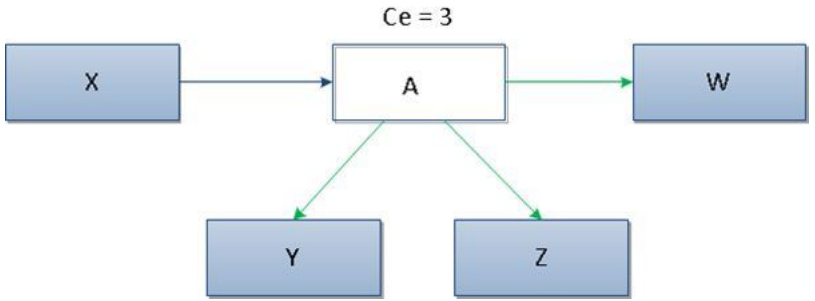
- 1995 : Abreu *etal.* inventent une suite de métrique nommée MOOD (*Metrics for Object Oriented Design*) pour évaluer 4 éléments . Contrairement à celles de CK, elles ne sont pas calculées pour une classe en particulier mais pour l'entièreté du projet. **Relativement peu utilisées comparativement aux métriques de CK.**
  - l'encapsulation
  - le couplage maximal
  - le polymorphisme
  - le pourcentage de méthodes héritées

Les métriques dédiées à la POO : métriques de Martin *etal.*

- 2003 : Martin *etal.* produisent une suite de métriques dédiées aux *packages*. Elles permettent de mesurer 6 aspects sur les packages. **Ces métriques sont intégrées dans des produits industriels comme SonarQube.**
  - le nombre de classes et d'interfaces dans les *packages* : permet de connaître la répartition classes/interfaces dans les différents packages pour équilibrer au mieux.
  - le couplage efférent : nombre de packages qui utilisent un package donné
  - le couplage afférent : nombre de packages qui sont utilisés par un package donné
  - le niveau d'abstraction des *packages* : nombre de classes abstraites divisé par les nombre de classes abstraites + le nombre de classes concrètes.
  - l'instabilité des packages : couplage efférent divisé par le couplage efférent + le couplage afférent.

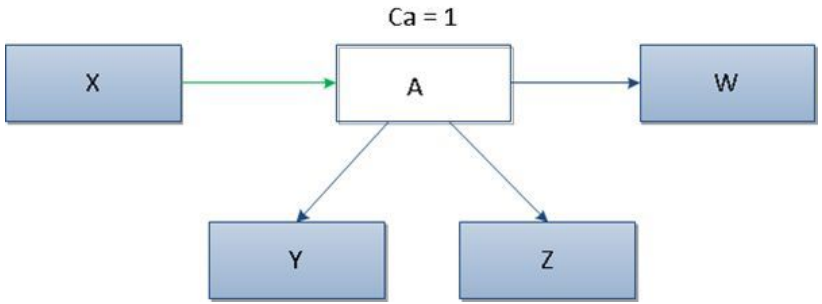
Les métriques dédiées à la POO : métriques de Martin *etal.*

- Ce = Couplage efférent



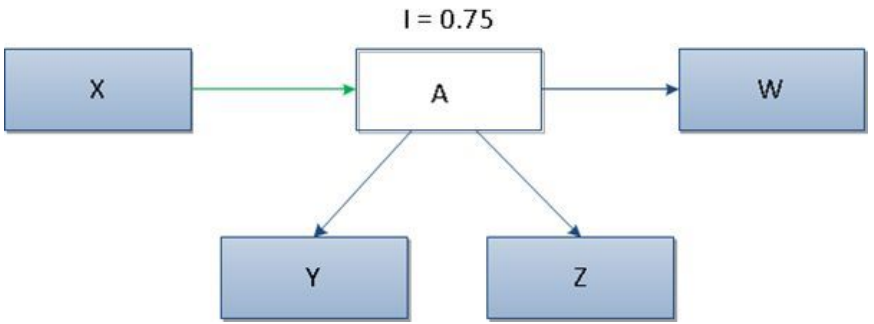
Les métriques dédiées à la POO : métriques de Martin *etal.*

- Ce = Couplage efférent
- **Ca = Couplage afférent**



Les métriques dédiées à la POO : métriques de Martin *etal.*

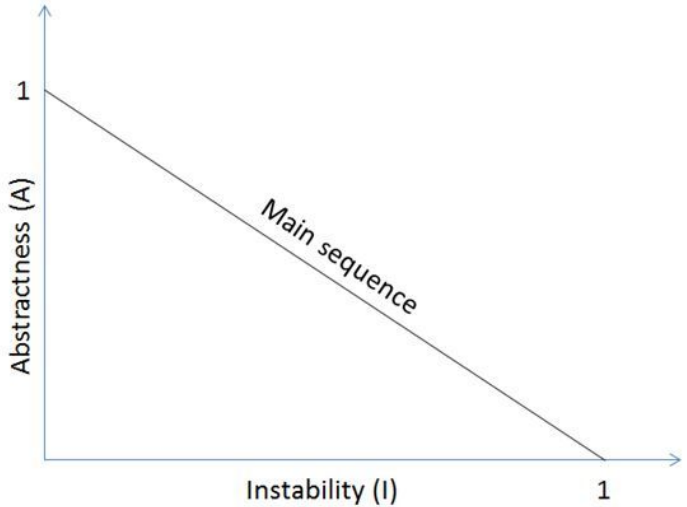
- Ce = Couplage efférent
- Ca = Couplage afférent
- **I = Instabilité des packages (Ce/Ce+Ca)**

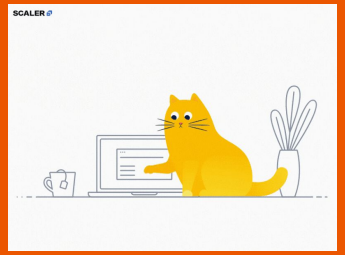


Les métriques dédiées à la POO : métriques de Martin *etal.*

- Ce = Couplage efférent
- Ca = Couplage afférent
- I = Instabilité des packages
- A = Nb classes abstraites / (Nb classes abstraites + Nb classes concrètes)

Plus l'abstraction est grande et plus l'instabilité est petite. Une grande instabilité d'un package sera révélateur du fait que celui-ci contient un grand nombre de classes concrètes.





Métriques liées au code : les métriques de maintenabilité

Les métriques de maintenabilité

Ces métriques sont des indicateurs de la maintenabilité d'un projet/code logiciel. Nous n'entrerons pas dans le détail de leur calcul car elles peuvent être relativement complexes à calculer (agrégation de métriques) mais il est bon de savoir qu'elles existent.

Les métriques de maintenabilité : Oman et Hagemeister

- 1992 : Oman et Hagemeister inventent la première métrique de maintenabilité dédiée à un système logiciel. Elle se base entre autre sur :
  - le nombre de changements effectués
  - la modularité
  - la complexité
  - le couplage
  - etc.

Les métriques de maintenabilité : Indice de maintenabilité de Microsoft

- 2021 : Microsoft développe un indice de maintenabilité dédiée à un système logiciel. Cet indice est intégré à l'éditeur Visual Studio. C'est un indice créé par agrégation de métriques. Cette indice se base sur :
    - la complexité cyclomatique
    - le volume de Halstead
    - le nombre de lignes de code

https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-maintainability-index-range-and-meaning?view=vs-2022

Courage c’est presque fini…

Métriques liées au processus de développement

Les métriques liées au processus de développement

Ces métriques vont permettre de monitorer les éléments qui gravitent autour du développement du projet. Ce sont généralement des mesures assez simples qui portent notamment sur :

- nombre de développeurs d'un projet
- âge du projet
- nombre de *commits*
- nombre de version
- nombre de construction sur plate-forme CI réussies
- etc..

Jenkins

Bitbucket

Exploitation de ces métriques

Ces métriques peuvent être mesurées et exploitées en utilisant 2 principaux moyens :

- Utiliser un plugin dans votre IDE permettant de calculer une ou plusieurs de ces métriques (ex: le plugin **MetricsTree** pour IntelliJ IDEA)
- Les calculer à la construction du projet via Maven (ex : **CK Calculator** sur Maven)
- Utiliser un outil de gestion de qualité type **SonarQube** intégrant le calcul de différentes métriques présentées précédemment.

La plupart des outils proposent de définir ou ont des seuils prédéfinis pour chacunes des métriques calculées.

Fin... (enfin)