

Rapport Test Driven Development

Continuation de projet

15/12/2017

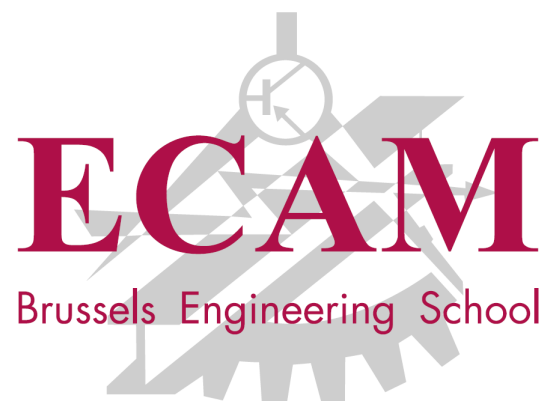
ALBERT Emile 14022@ecam.be

ANIZET Thomas 14164@ecam.be

LEKENS Amaury 14027@ecam.be

SELLESLAGH Tom 14164@ecam.be

WÉRY Benoit 14256@ecam.be



1 Métrique - Qualité structurelle du code

Ligne de Code *SLOC*

- Ligne physiques : Présentes dans le fichier
- Lignes logiques : Effectivement exécutées

Densité des commentaires par rapport aux lignes de code DC

- $DC = CLOC/SLOC$
- CLOC : Comment Line Of Code – Nombre de lignes de commentaires
- SLOC : Source Line Of Code – Nombre de lignes de code

Couverture de code

- Proportion de code couverte par des tests

Duplication de code

Couplage

- Couplage efférent "Ce" : Nombre références vers classe mesurée
- Couplage afférent "Ca" : Nombre types que la classe connaît

Instabilité

- Résistance d'un module au changement

$$\frac{C_e}{C_e + C_a} \quad (1)$$

Lack of Cohesion of Methods

- Manque de cohésion des méthodes

$$LCOM = 1 - \frac{\sum_F MF}{M * F} \quad (2)$$

Où M est le nombre méthodes, F est le nombre champs d'instance et MF est le nombre de méthodes appelant un champ donné

Nombre d'éléments

- Nombre d'éléments dans une classe
 - Paramètres ≤ 5
 - Variables ≤ 8
 - Surcharges ≤ 6

Conclusion

Interprétation pour évaluer (selon la norme ISO 9126) la qualité structurelle et fonctionnelle de notre code selon plusieurs critères

Qualité structurelle

- La performance Quantité de ressources utilisées (moyens matériels, temps, personnel), et la quantité de résultats délivrés. En font partie le temps de réponse, le débit et l'extensibilité - capacité à maintenir la performance même en cas d'utilisation intensive
- La maintenabilité
Effort nécessaire à corriger ou transformer le logiciel. En font partie l'extensibilité, c'est-à-dire le peu d'effort nécessaire pour y ajouter de nouvelles fonctions ;
- La portabilité
Aptitude d'un logiciel de fonctionner dans un environnement matériel ou logiciel différent de son environnement initial. En font partie la facilité d'installation et de configuration dans le nouvel environnement.

Qualité logicielle

- La capacité fonctionnelle
La capacité qu'ont les fonctionnalités d'un logiciel à répondre aux exigences et besoins explicites ou implicites des usagers. En font partie la précision, l'interopérabilité, la conformité aux normes et la sécurité
- La facilité d'utilisation
Effort nécessaire pour apprendre à manipuler le logiciel. En font partie la facilité de compréhension, d'apprentissage et d'exploitation et la robustesse - une utilisation incorrecte n'entraîne pas de dysfonctionnement
- La fiabilité
Capacité d'un logiciel de rendre des résultats corrects quelles que soient les conditions d'exploitation. En font partie la tolérance aux pannes - la capacité d'un logiciel de fonctionner même en étant handicapé par la panne d'un composant (logiciel ou matériel)

2 Convention de codage

Les conventions de codage ont été légèrement revues par rapport aux conventions décidées par le groupe précédent. En effet, par nos modifications, nous avons souhaité formaliser l'aspect général du code (emplacement et forme des commentaires)

2.1 Nom de classe

- UpperCamelcase

```
1 public class XmlHttpRequest() {}
```

2.2 Nom méthode

- lowerCamelcase

```
1 public void supportsIpv6nIos() {}
```

2.3 Constante

- CONSTANT_CASE = ... tout est écrit en majuscule

2.4 Nom variable

- lowerCamelcase

```
1 int supportsIpv6nIos = 6;
```

2.5 Variable/Objet nouvelle classe

- A name in the form used for classes followed by the capital letter.

2.6 Indentation

- 4 espaces

2.7 Nombre de caractère par ligne maximum

- 100

2.8 Commentaires

- Les commentaires en anglais se trouvent toujours au dessus du code concerné et suivent le format ci-dessous :

```
1 Comment on one line:
2  /* ... */
3
4 Comment on several lines:
5  /* ...
6   * ...
7   */
```