# A52 – Qualité de Développement

Romain Orhand

*rorhand@unistra.fr*

Université de Strasbourg

IUT Robert Schuman

Institut universitaire de technologie

Université de Strasbourg

# Exercice - Rappel



De quels éléments du précédent TD vous souvenez-vous ?

# Retours sur Kanban oueb – Partie 1



1. La présence des fonctionnalités
2. Des comportements étranges pour les fonctionnalités implémentées ?
3. Accessibilité de votre site kanban oueb
4. Persistance des données
5. Aspect graphique
6. Respect des consignes
7. La communication avec le client

# Sommaire





# Premières petites questions



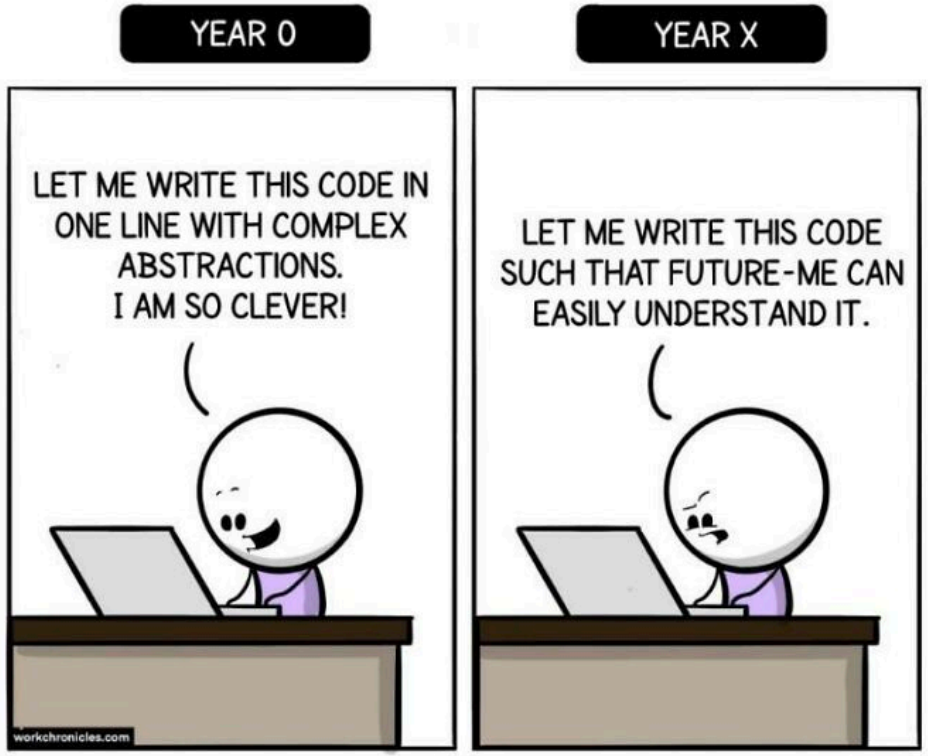
Pouvez-vous me redéfinir ce qu'est la maintenabilité pour la qualité logicielle ?

Qu'avez-vous vu dans votre cursus pour travailler la maintenabilité de votre code ?

# Maintenabilité



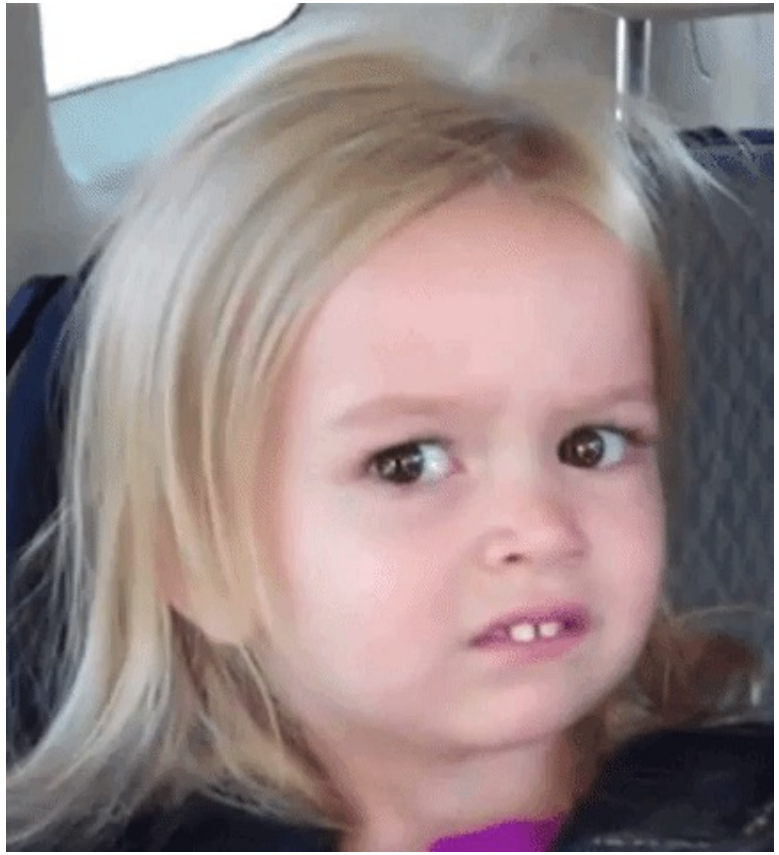
La Maintenabilité représente le degré d'efficacité avec lequel un produit ou un système peut être modifié pour l'améliorer, le corriger ou l'adapter aux changements de l'environnement et des exigences.



# Indicateurs de maintenabilité



1. Le nombre de lignes de code
2. La complexité cyclomatique
3. La mesure de complexité d'Halstead



# Nombre de lignes de code

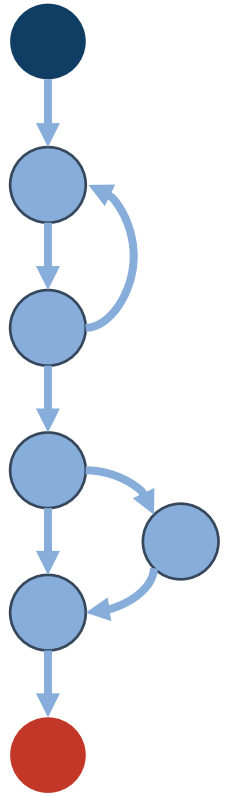


```c
#include <stdio.h>

int main(int argc, char ** argv) {
    // Pas envie qu'il y ait plus d'un argument.
    if (argc > 2) exit(1);
    // Là, je suis content.
    printf("Hello world !\n");
    return 0;
}
```

# Complexité cyclomatique



```c
#include <stdio.h>

int main(int argc, char ** argv) {
    // Comptons pour compter
    int cpt = 0;
    scanf("%d",&cpt);
    do {
        printf("Je sais compter jusque %d\n", cpt);
        cpt++;
    } while (cpt < 10);
    // Yolo
    if (argc > 2) cpt++;
    // Là, je suis content.
    printf("Bye bye world !\n");
    return 0;
}
```



# Mesures de complexité d'Halstead



On a besoin du :

- nombre d'opérateurs uniques $n_1$ ;
- nombre d'opérandes uniques $n_2$ ;
- nombre total d'opérateurs $N_1$ ;
- nombre total d'opérandes $N_2$.

On peut calculer :

- la longueur du programme $N = N_1+N_2$ ;
- le vocabulaire du programme $n = n_1+n_2$ ;
- le volume du programme $V = N*\log_2(n)$ ;
- la difficulté $D = n_1 * N_2 / n_2$.

On peut alors avoir **l'effort** qui le produit du **volume** par la **difficulté**.

# Extensibilité



L’extensibilité désigne la capacité d'un système ou d'un logiciel à être facilement **étendu ou élargi sans nécessiter de modifications majeures** de sa structure existante.

# Autres petites questions



Un code maintenable est-il facilement extensible ?

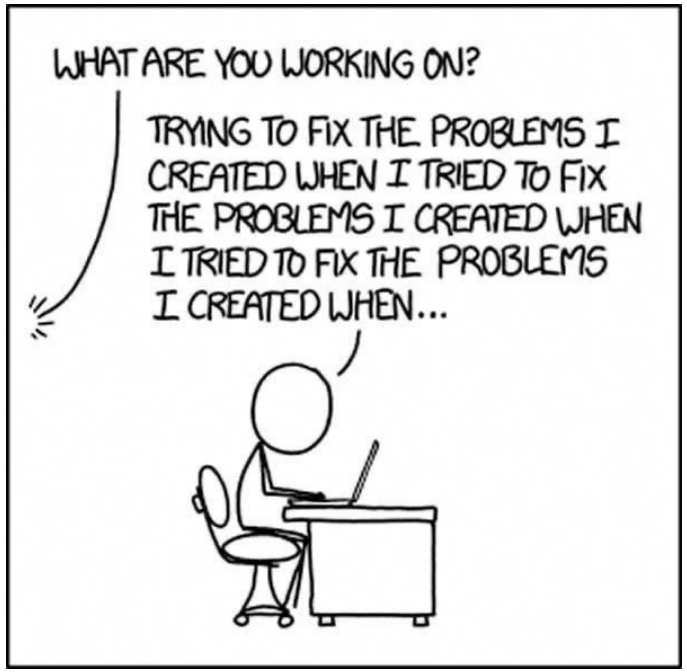
Un code extensible est-il facilement maintenable ?

# La dette technique, c’est quoi ?



« Sortir une première itération de code, c'est comme s'endetter. Une petite dette accélère le développement tant qu'elle est remboursée rapidement par une réécriture... Le danger survient lorsque la dette n'est pas remboursée. Chaque minute passée sur un code pas tout à fait correct compte comme un intérêt sur cette dette. »

— Ward Cunningham, 1992



# Maîtriser sa dette technique

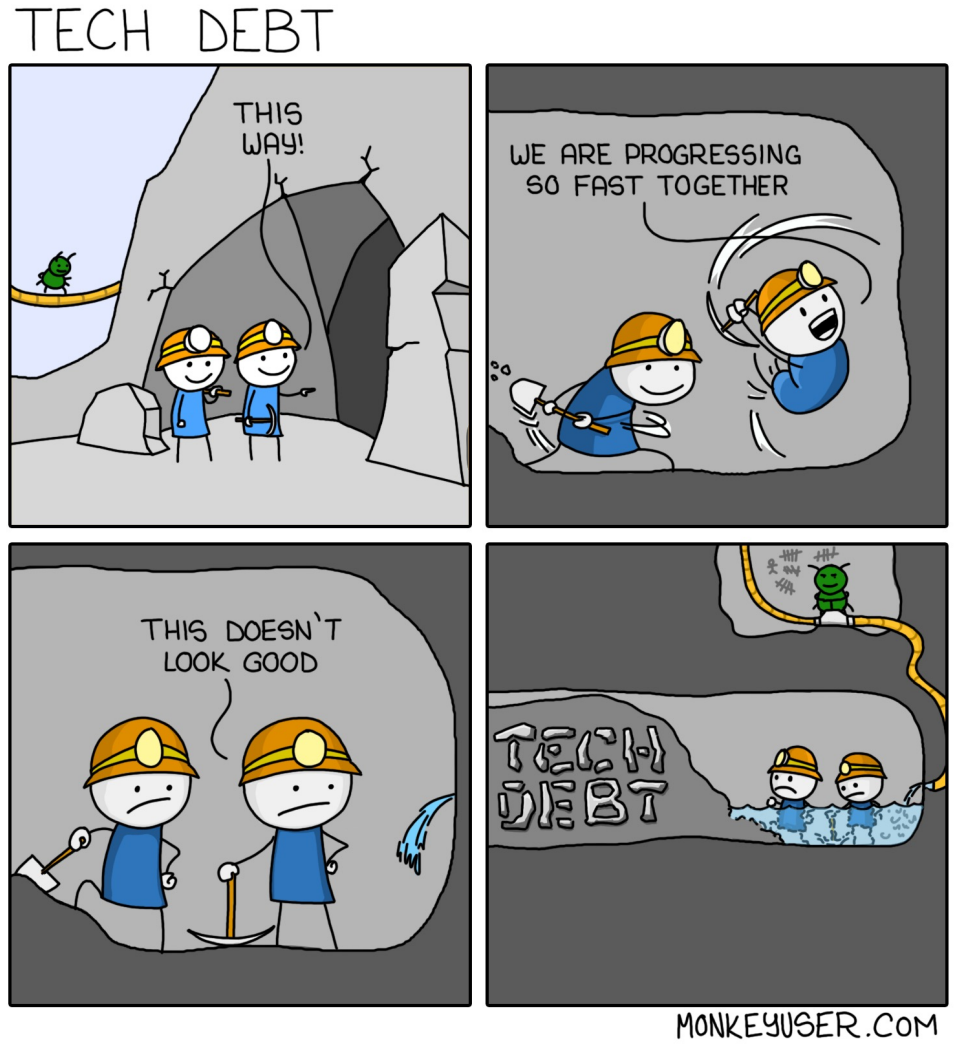


Tout code induit une dette technique qui doit être **quantifiée**.

Une manière d'y parvenir est de **quantifier les efforts** que votre équipe de développement consacre à **maintenir les fonctionnalités existantes dans l'état où elles se trouvaient auparavant**, puis de **chercher à réduire** ces efforts.

Il faut alors **identifier** les choix qui ont été faits et qui ne conviennent pas complètement, et **planifier** le travail permettant de passer outre ces choix.

Et comment peut-on **éviter** la dette technique ou la **réduire** ?



# Sommaire





# Portabilité



*La Portabilité* est le degré auquel un système ou un composant peut être transféré d'un environnement matériel, logiciel ou autre environnement opérationnel ou d'utilisation à un autre.

Trois questions :

- Est-ce que mon logiciel peut facilement tourner sur d'autres plateformes ? Quelles sont les modifications que cela implique ?

- Est-ce que mon logiciel s'installe facilement ? La souffrance de l'utilisateur est-elle minimisée ?

- Est-ce que mon logiciel peut remplacer des logiciels concurrents sans soucis ?

# Portabilité : Adaptabilité



Une idée à retenir : il faut essayer de garder à l'esprit, quand on code, **le nombre de lignes qui ne passeront pas sur d'autres systèmes**, car elles témoignent de l'ampleur des modifications qu'il faudra apporter. Elles sont donc, si possible, à minimiser !

$$M_{port}(p) = 1 - C_{port}(p)/C_{new}(p)$$

$$M_{elem}(p) = 1 - \sum_{i=1}^{N_{el}} C_{el}(el_i)/(N_{el} * N_{engines})$$

*Des questions ? Non ? Merci !*

# Portabilité : Installabilité et Remplaçabilité



*Pour l'installabilité :*

$$ESR(e) = N_{succ}/N_{total}$$

$$IE(e) = \begin{cases} 0 & \text{if } NDS = 0 \\ \frac{AIT(e)}{NDS(e)} & otherwise \end{cases}$$

*Et pour la remplaçabilité ?*

# Encore une petite question



Qu’avez-vous vu dans votre BUT qui vous permette de favoriser la portabilité de vos applications ?

# Sommaire





# Densité de commentaires



On commence avec la densité de commentaires.

Faut-il que cette métrique soit élevée, ou faible ?
Que nous apprend-elle réellement ?

# Couverture de test

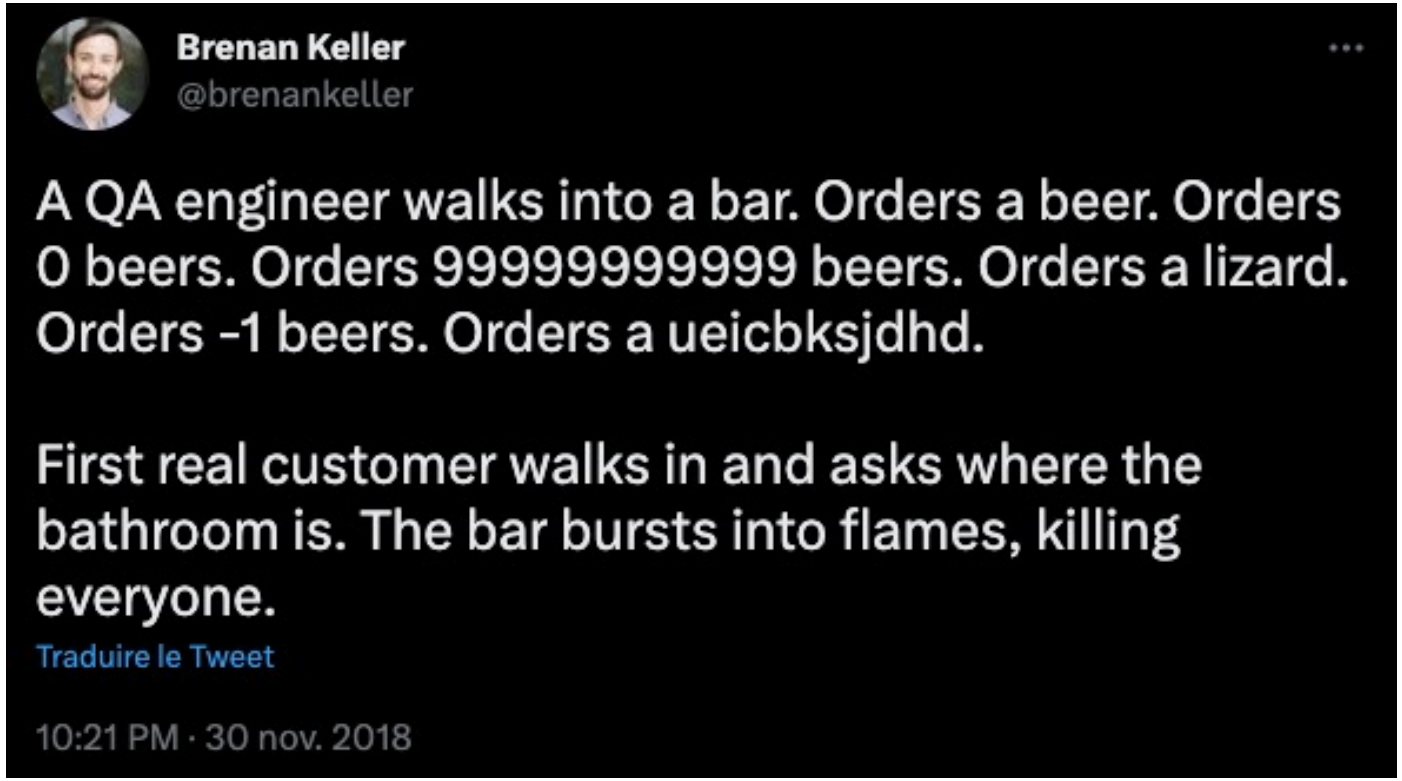


Deux couvertures différentes :

-> la **couverture du code** qui quantifie la part du code traversée par des tests ;

-> la **couverture des cas** qui indique combien de cas d'utilisation sont couverts par des tests.

Vers quelle couverture aller ?



Assez explicite, idéalement que pouvons-nous chercher à obtenir ?

# Sommaire





Fouillez dans la documentation *GitLab*
pour trouver des outils vous permettant d'évaluer votre code.

https://docs.gitlab.com/ee/ci/testing/code_quality.html

https://docs.gitlab.com/ee/user/application_security/sast/

Mettez en place *GitLab code quality*

sur votre projet Kanban oueb !

Mettre en place *GitLab accessibility testing*

sur votre projet Kanban oueb !

Regardez si d'autres éléments dans la documentation *GitLab* pourraient vous sembler pertinents à mettre en place.

Si vous en trouvez, vous savez quoi faire maintenant !

# Conclusion