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 ?

Qu’est-ce que la revue de code ?

Que permet-elle ?

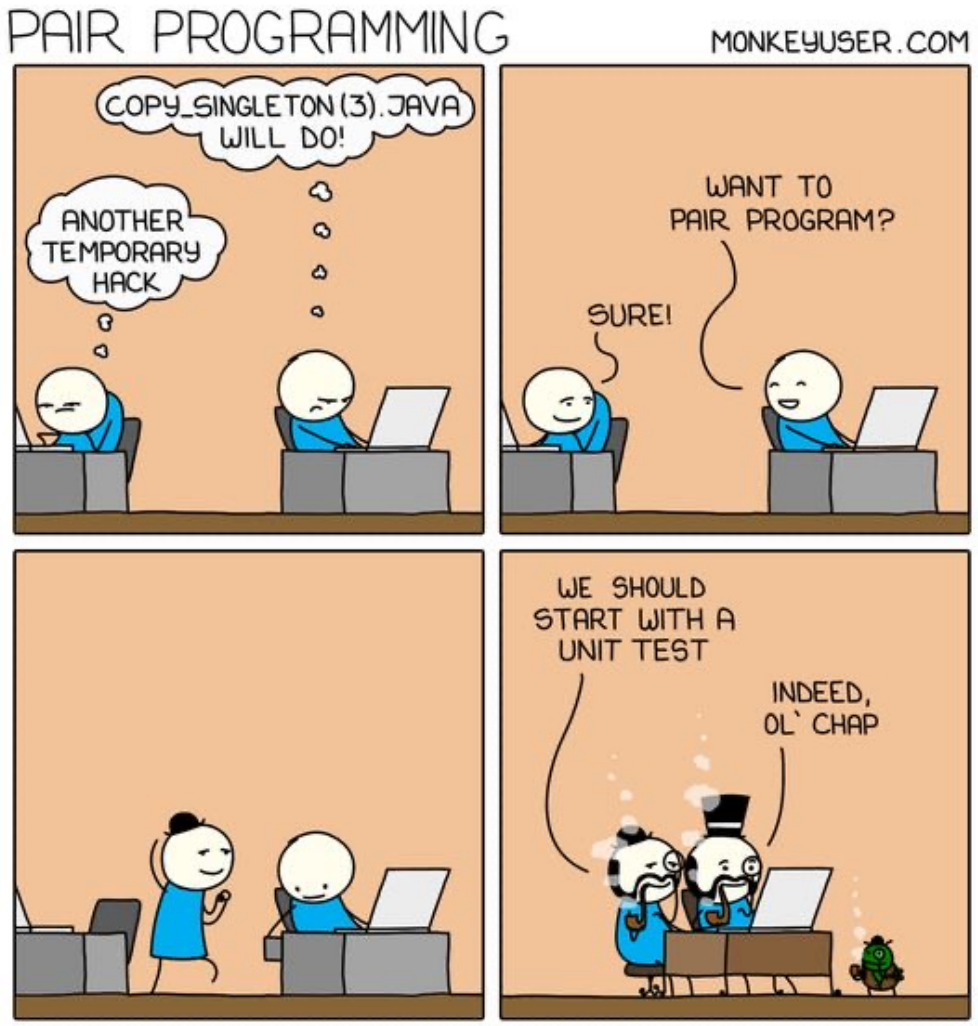
Sommaire

La revue de code

Il s'agit d'une **technique d'inspection du logiciel**.

Autrement dit, il s'agit d'une technique (in)formelle **d'évaluation** par laquelle des produits ou documents **informatiques** sont examinés en détail par **une personne ou un groupe autre que son auteur** pour y **détecter des erreurs**.

Vous pratiquez déjà la revue de code !



Pourquoi la revue de code ?

- ✓ Pour du code **lisible**, **propre**, **sans bugs**, **maintenable** et **documenté**.
- ✓ C'est alors l'occasion rêvée pour poser des questions, y répondre et discuter : vous **communiquez** et **prenez du recul** !
- ✓ Permet de **partager des connaissances**, qu'elles soient relatives au projet ou à la technique.
- ✓ Si un projet est mieux connu, il est plus facile **d'estimer** le temps nécessaire pour développer de nouvelles fonctionnalités.
- ✓ Il est alors aussi possible de former de nouvelles recrues sur votre projet, de les faire **monter en compétence** plus vite.

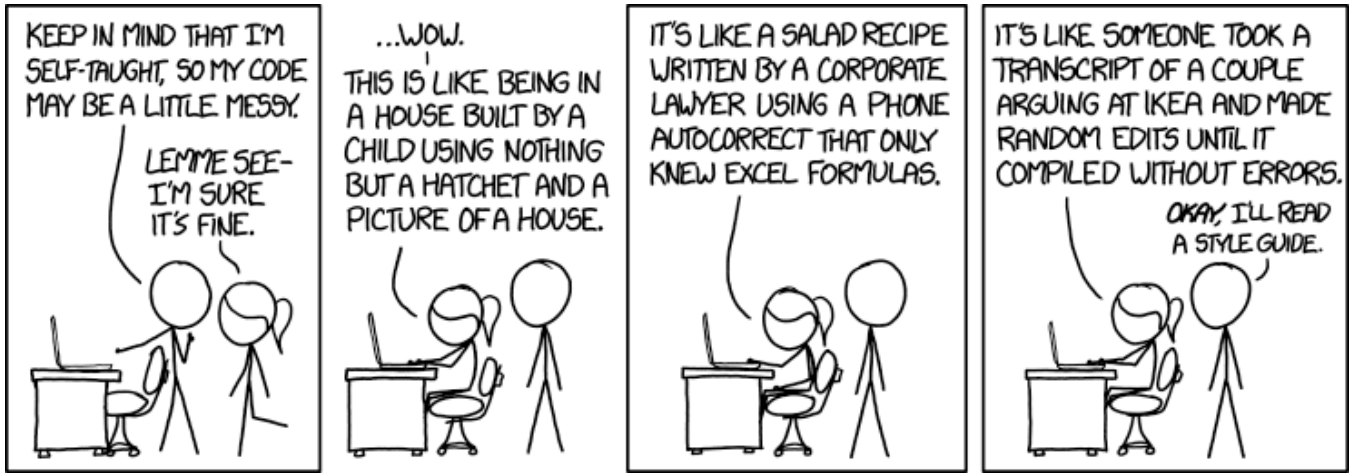
Mais attention…

La revue de code prend du **temps** !

Elle se **prépare** et doit être **méthodique**.

Elle doit être faite avec **respect**.

Source : xkcd



Petite question

Selon vous, qui peut faire la revue de code ?

Exercice 2

A quel moment pouvez-vous faire une revue de code ?

Quelques techniques de revues de code

L’inspection « Fagan » ou inspection formelle.

L’analyse par-dessus l’épaule.

La programmation en binôme.

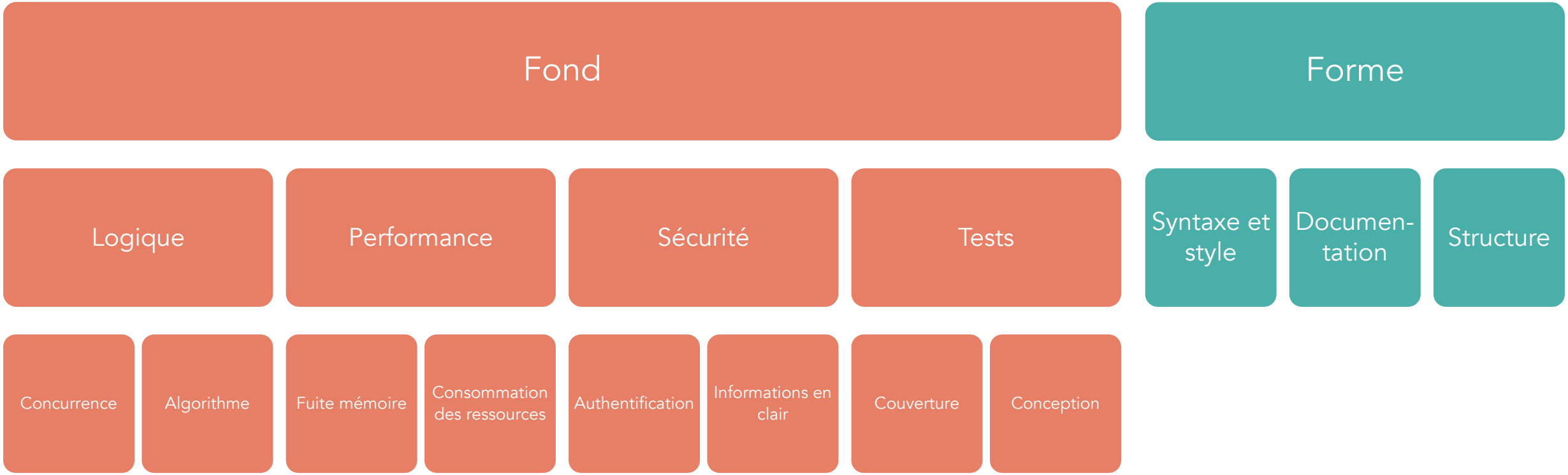
La notification par email.

Sommaire

Exercice 3

Concrètement, quelles erreurs pouvez-vous essayez de trouver lors d'une revue de code ?

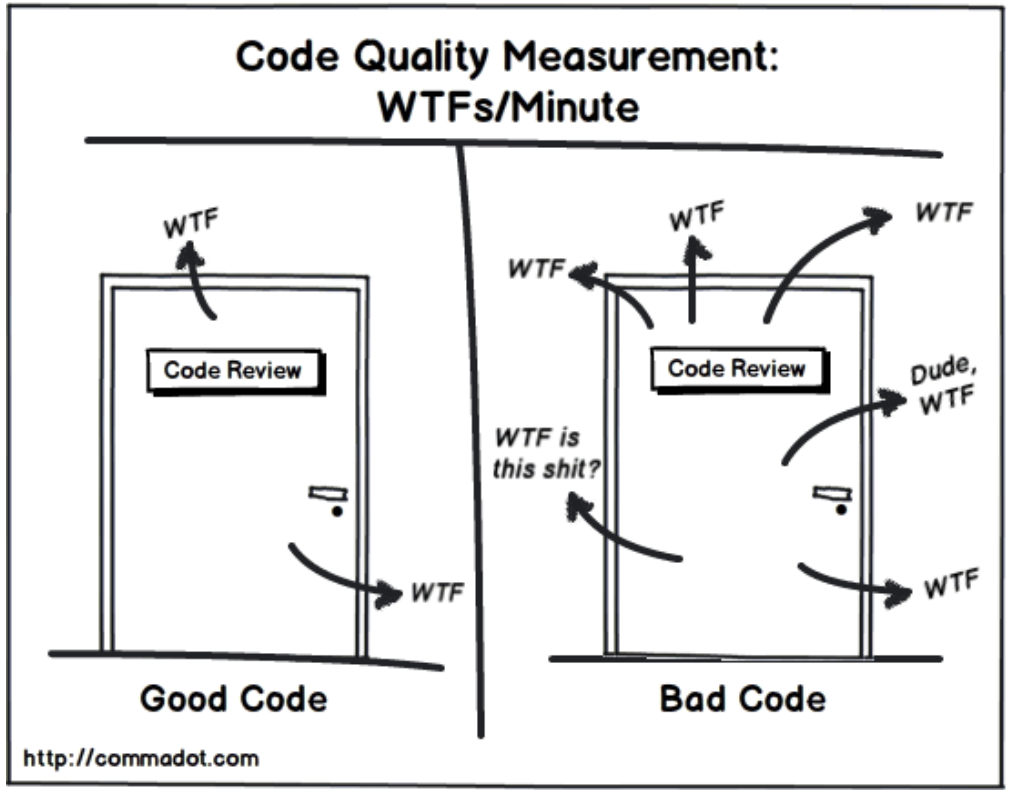
Des erreurs de fond et de forme



Et des mauvaises pratiques..

Au choix, des *antipatterns* ou des *code smells*.

En connaissez-vous ?



Exercice 4


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void my_function(char * input){
    char * buffer = malloc(10);
    strcpy(buffer, input);
    for (size_t i = 0; i < sizeof(buffer)/2; i++){
        char tmp = buffer[i];
        buffer[i] = buffer[sizeof(buffer)-1-i];
        buffer[sizeof(buffer)-1-i] = tmp;
    }
    printf("Reverse string of %s is %s\n", input, buffer);
}

int main(int argc, char ** argv) {
    if (argc < 2) {
        printf("Usage: %s <input string>\n", argv[0]);
        exit(1);
    }
    my_function(argv[1]);
    return 0;
}
```

Il est demandé à ce qu'une chaine de caractères de longueur 10 au plus soit inversée, puis affichée.

A vous de faire la revue de code et de faire les corrections.

Mettez-vous en équipe, la même que pour le projet A52, et faites rapidement un dépôt GitLab avec ce bout de code où vous pousserez vos modifications dans une branche *fix*.

Exercice 5

Vous devez avoir un dépôt GitLab avec le bout de code précédent, tous les membres de votre projet A52 ajoutés sur ce dépôt et des modifications selon votre revue de code.

Maintenant, vous allez ajouter un membre d'un groupe de l'autre groupe de TD, afin que cet autre groupe puisse faire une revue de code par le biais d'une *merge request* de la branche *fix* vers la branche *main*.

Cette fois-ci, ce sont vos propositions et modifications qui seront revues.

Exercice 6


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void reverse_onplace_string(char * input){
    char * reverse_string = malloc(10);
    if(reverse_string != NULL) {
        strncpy(reverse_string, input, 10);
        size_t length = (10 < strlen(input)) ? 10 : strlen(input);
        for (size_t i = 0; i < length/2; i++){
            char tmp = reverse_string[i];
            reverse_string[i] = reverse_string[length-1-i];
            reverse_string[length-1-i] = tmp;
        }
        reverse_string[9] = '\0';
        printf("Reverse string of %s is %s\n\n", input, reverse_string);
        free(reverse_string);
    }
}

int main(int argc, char ** argv) {
    if (argc != 2) {
        printf("Usage: %s <input string>\n", argv[0]);
        exit(1);
    }
    reverse_onplace_string(argv[1]);
    return 0;
}
```

Même demande, code différent.

A vous de faire juste la revue de code !

Pas de modifications,

que de la relecture !

Sommaire

Déjà, quelques conseils de bon sens

Par la suite, peu importe avec qui vous travaillerez : soyez bienveillant.

Relisez votre code avant de demander à ce qu'il soit relu par quelqu'un d'autre.

Faites des petites revues de code.

Explicitez vos messages de *commits*.

Planifiez vos revues de code.

Des exemples de questions à se poser

Si vous fouillez sur le net, vous pouvez trouver des *check lists* (ou listes de contrôle) pouvant vous aider. Par exemple : *https://github.com/mgreiler/code-review-checklist*

- A-t-on de bons tests (unitaires, …) ?
- Le code est-il compatible avec les conventions de l'équipe et de l'entreprise ?
- Le code est-il lisible, compréhensible, consistant ?
- Les patterns utilisés et l'organisation du code sont-ils judicieux ?
- Le code créé-t-il une faille de sécurité ?
- Pourrait-on avoir un problème de performance ?
- Pourrait-on rencontrer des problèmes difficilement reproductibles (fuites mémoires, gestion des erreurs, multithreading, deadlocks, …) ?
- La documentation doit-elle être ou est-elle mise à jour ?

Des outils ?

Dépend de la méthode de revue de code que vous choisissez !

Sinon, les gestionnaires de versions (*Git* et compagnie), *GitHub*, *GitLab*, *Review Board*, *Gerrit*, *Code Collaborator*, *Mondrian*, *CodeFlow*, les *linters*, tout outil de communication (*mails*, *discord*, *slack*, …), les outils d'Intégration Continue (*Jenkins*, *GitLab-CI*, …) et encore bien d'autres.

Sommaire

Exercice 7

Allez fouiller dans à l'url suivante : *https://docs.gitlab.com/ee/ci/environments/*

a) Qu'avez-vous appris ou retenu de cette documentation ?
b) Est-ce que cela peut vous être utile pour faire de la revue de code ?
c) Si oui, comment ? Si non, pourquoi ?

Exercice 8

Allez fouiller dans à l'url suivante : *https://docs.gitlab.com/ee/user/project/pages/*

a) Qu'avez-vous appris ou retenu de cette documentation ?
b) Est-ce que cela peut vous être utile pour faire de la revue de code ?
c) Si oui, comment ? Si non, pourquoi ?

Vous aurez bientôt l'occasion de jouer avec du déploiement …

Pour la prochaine fois

Documentation

Mais avant …