Principes de développement AGILE / SCRUM

Les principes de développement Agile : le codage « be solid »

Principes de développement : BE SOLID !

J'ai retrouvé avec grand plaisir les principes que j'avais appris fin des années 90 (autant que je me souvienne le principe de **S**ubstitution de Barbara Liskov est encore plus ancien), rassemblés dans le "cri" : SOLID. Je voudrais préciser simplement ce que signifie à mon sens "être" SOLID.

Beauté du code

Sans lancer le débat sur l'art de coder - le *crafts(wo)manship* est-il un art ? - je me souviens d'un code-source (en Pascal je crois) que j'avais récupéré chez Borland. Et j'avais trouvé cela... *beau* ! Oui, comme une carte électronique, un bâtiment peuvent être (ou pas...) beaux. En ce sens, la notion d'architecture est parlante, trouver cet équilibre entre ingénierie et art.

Notez que ce n'est pas un raisonnement conscient qui me fait trouver un code beau ou pas, c'est une émotion qui naît, basée probablement sur ces fameux attracteurs engrangés depuis toutes ces années (voir Vision un sacré attracteur pour une équipe auto-organisée).

Autrement dit, une façon de dire la maintenabilité, la capacité à faire évoluer un code pourrait se traduire en beauté du produit logiciel.

C'est aussi un mode de vie : à faire quelque chose, autant que ce soit beau : **à le faire, autant que ce soit beau !**

Code Expressif

L'une des facettes du développement - ou conception - simple est le code expressif.

Voir à ce sujet Simple Design :

States every intention important to the programmers

page 57 Extreme Programming Explained 1999 Kent Beck.

D'ailleurs, différents refactorings sont consacrés à l'expressivité du code. Une facette du code expressif est la gestion des commentaires, les réserver en "plan B" lorsque je n'arrive pas à exprimer directement dans le code ce que je souhaite dire. L'un des problèmes du commentaire est que je ne sais pas écrire de test de non-régression sur un commentaire.

--

Et c'est là que se trouve cette facette du *crafts(wo)manship* qu'est la relation entre junior et senior, la durée nécessaire (les fameuses 10.000 heures) à l'acquisition de cette capacité à créer un code beau et expressif, qui respecte les principes SOLID.

- Beauté du code

- code Expressif

- SOLID

Développement agile : code expressif

L'une des facettes du développement agile est le **code expressif**.

Mais code expressif... Pour qui ?

Agile : gagnant / gagnant / gagnant

Aux deux acteurs principaux de l'agilité :

- Développeur
- Utilisateur (représentant)

ajoutons (c'est le cas dans l'Extreme Programming) un troisième larron : **le Développeur du Futur.** Ce Développeur qui, dans un futur plus ou moins proche, aura à faire évoluer, corriger, un code écrit aujourd'hui.
Voila donc les trois gagnants : l'Utilisateur, le Développeur, le Développeur du Futur. C'est dans ce contexte que se pose la question du **code expressif.**

Expressif ?

Un code source est expressif lorsqu'il exprime l'intention du Développeur. Plus précisément lorsque ce code permet de comprendre la conception telle que "pensée" par le Développeur.
Voici un exemple simple et concret, repris du "kata" que j'utilise parfois en formation : FizzBuzz.

```
public class FizzBuzz {

    public String listeFizzBuzz(int n) {

        String chaine = "";

        for(int j = 1; j <= n; j++)
        {
            if (j > 1)
                chaine = chaine + " ";
            if ((j % 3) == 0)
                if ((j%5)== 0)
                    chaine = chaine + "FizzBuzz";
                else
                chaine = chaine + "Fizz";
            else
                if ((j %5) ==0)
                    chaine = chaine + "Buzz";
                else
                    chaine = chaine + Integer.toString(j);
        }

        return chaine ;
    }
}
```

Une autre expression pourrait être

```
public class FizzBuzz {

    public String listeFizzBuzz(int n) {

        String chaine = "1";

        for(int j = 2; j <= n; j++)
        {
              chaine = chaine + " ";
       ...
```

Ce n'est pas le même algorithme qui est exprimé. Reste à savoir si ce deuxième algo est bien ce que je voulais dire ou bien une petite astuce de dev qui me permet de régler le cas de cet espace entre deux termes, cas qui apparait lors du test sur la valeur 2.

Les pratiques qui favorisent le code expressif

Plusieurs pratiques de développement agile favorisent le code expressif.

TDD

Le TDD et ses phases RED GREEN REFACTOR est une incitation forte au code expressif. Sous réserve de ne pas shunter "REFACTOR". J'y reviendrai plus tard.

Binôme

Le pair programming est aussi une incitation au code expressif. Disons simplement que le partenaire joue le rôle de "Développeur du Futur"... au présent. C'est un emprunt à la méthode XP.

Propriété collective du code

Encore une pratique favorable au code expressif. Lorsque les différents codes-sources tournent dans l'équipe, le feed-back généré améliore l'expression du code. Pourvu que ce feed-baxck soit effectivement exploité.

Enfant et Adulte

Le TDD (RED GREEN REFACTOR) m'a aidé à prendre conscience de l'ambivalence du développement. C'est à la fois un "jeu" et également une affaire sérieuse en terme de maintenabilité et qualité intrinsèque.

Développeur-Enfant

C'est un jeu. Lorsque j'écris un test qui représente une nouvelle étape de l'analyse-conception du produit et lorsque j'obtiens la barre verte, faut bien le dire... C'est excitant. C'est même enfantin en terme de plaisir rapide. Le TDD étant basé sur des cycles de quelques minutes.

Développeur-Adulte

Et le TDD ne se termine pas sur la barre verte. Reste le REFACTOR, outil indispensable au code expressif. C'est là que je remplace ma casquette "enfant" par une casquette "adulte" responsable. C'est quoi le code que je laisse au Développeur du Futur ?
Il ne s'agit donc pas uniquement d'une relation (code expressif, dette technique) entre code et Développeur du présent.

Déresponsabilisation

Mais il y a un problème. Nous sommes dans une société infantilisante. Est-ce un effet du matriarcat ambiant ? (Oups, n'allons pas trop loin non plus...)
Si je suis développeur et qu'un Chef de Projet m'impose un planning impossible, mis à part le fait qu'il se ridiculise, je suis déresponsabilisé.
Et donc j'en reste du mieux que je peux à la phase infantile de mon boulot si telle est la volonté du "système".

Agile = Responsable + Hédoniste

Vous voyez, nous retrouvons ici les deux piliers de l'agilité : responsabilité et hédonisme.
Si vous refactorez votre code de telle façon qu'il soit plus maintenable en étant plus expressif alors vous êtes responsable. Vous êtes même agile si au préalable vous avez pris votre pied grâce à la barre verte. Comme quoi, être agile c'est jouir/sourire plusieurs fois par heure. Sinon ...