

# Art du refactoring

---

Emmanuel Conrardy, Arolla  
@manu\_the\_chene

# Refactoring, c'est quoi ?

---

Restructurer  
Nettoyer  
Améliorer  
Ajuster  
Adapter

...

le système sans changer son comportement





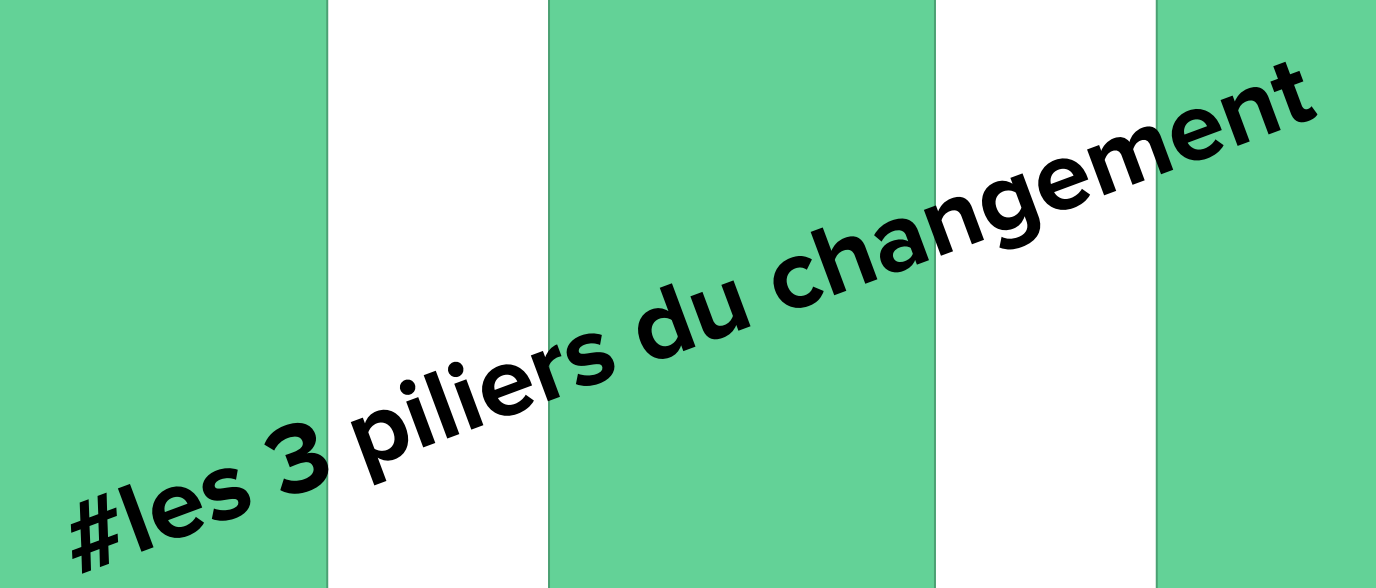
Pour



répondre à un besoin d'évolution.  
reprendre un code legacy.  
éviter une refonte risquée.  
soutenir la stratégie technique.  
assurer la maintenabilité du système.  
...

# Refactoring, pourquoi ?

---

The image features three identical, solid green rectangular pillars standing side-by-side on a white background. They are positioned at approximately one-third, two-thirds, and three-quarters of the way across the frame. A bold, black, sans-serif text is superimposed diagonally across the middle pillar, starting from the bottom left and extending towards the top right.

**#les 3 piliers du changement**



**Produit**

Si il est utilisé, il va  
changer



**Produit**

Si il est utilisé, il va  
changer



**Orga**

Si il doit changer, il va être  
conduit.





**Produit**

Si il est utilisé, il va  
changer



**Orga**

Si il doit changer, il va être  
conduit.



**Qualité**

Si il doit être conduit, il  
devrait être faisable.

# Produit

#UX #Designer  
#marketing

Si il doit être utilisé, il va  
changer

# Orga

#Manager  
#Scrum Master

Si il doit changer, il va être  
conduit.

# Qualité

#Developpeur  
#Testeur

Si il doit être conduit, il  
devrait être faisable.

# Produit

#UX #Designer  
#marketing

Si il doit être utilisé, il va  
changer

# Orga

#Manager  
#Scrum Master

Si il doit changer, il va être  
conduit.

# Qualité

#Developpeur  
#Testeur

Si il doit être conduit, il  
devrait être faisable. #the why !

#Architecte  
#CEO

#Manager  
#Scrum Master



On est tous dans le  
même camp !

#Utilisateur

#UX  
#Designer  
#marketing

#Developpeur  
#Testeur

Le refactoring assure  
que le changement  
reste faisable.

Ainsi on évite que que  
nos équipes passent  
de  à   
quand on ne peut plus  
changer notre système

Refactoring, quand ?

---

Est-il temps de nettoyer ?





#adding new  
function

Régulièrement.  
Refactoring quotidien.

#fix a bug

#review

#DDD

Stratégiquement.  
Refactoring planifié.

#long terme

#Architecture

Alertes :

“On a pas le temps !”

“On n’est pas autorisé !”

#boy-scout rule

Pour le quotidien pas  
besoin d'autorisation.

#just do it

#TDD

#team-play

Pour le stratégique  
agissez en équipe.

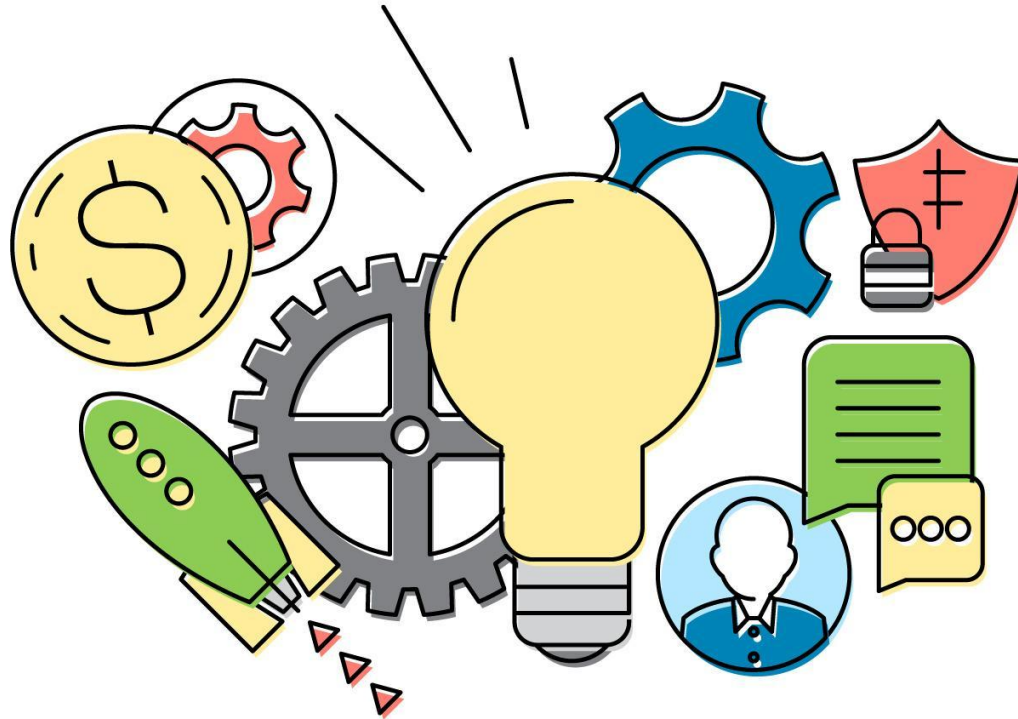
#pas de  
sous-marin

#Soutient  
hiérarchique

Refactoring, où ça ?

---

Par où on commence ?



Qu'est ce qui apporte  
de la valeur à nos  
utilisateurs ?



Qu'est ce qui change  
fréquemment ?

Quel est le besoin ?

#stay focus

Où ne devrions pas faire  
de refactoring ?

#ROI

#KPI

#intuition

Quelle a été votre  
première idée ?

#risque

#Experience

Au sein d'une équipe,  
savoir par où  
commencer appartient à  
l'équipe et chacun peut  
apporter ses idées.

Refactoring, comment ?

---

#Code smell

#primitive  
obsession

#Nested If

#feature envy

#lazy class

Step 0 : acquérir le savoir

#Design pattern

#value object

#strategy

#factory method

#adapter

Step 0 : acquérir le savoir



#Kata

#TDD

#Mikado method

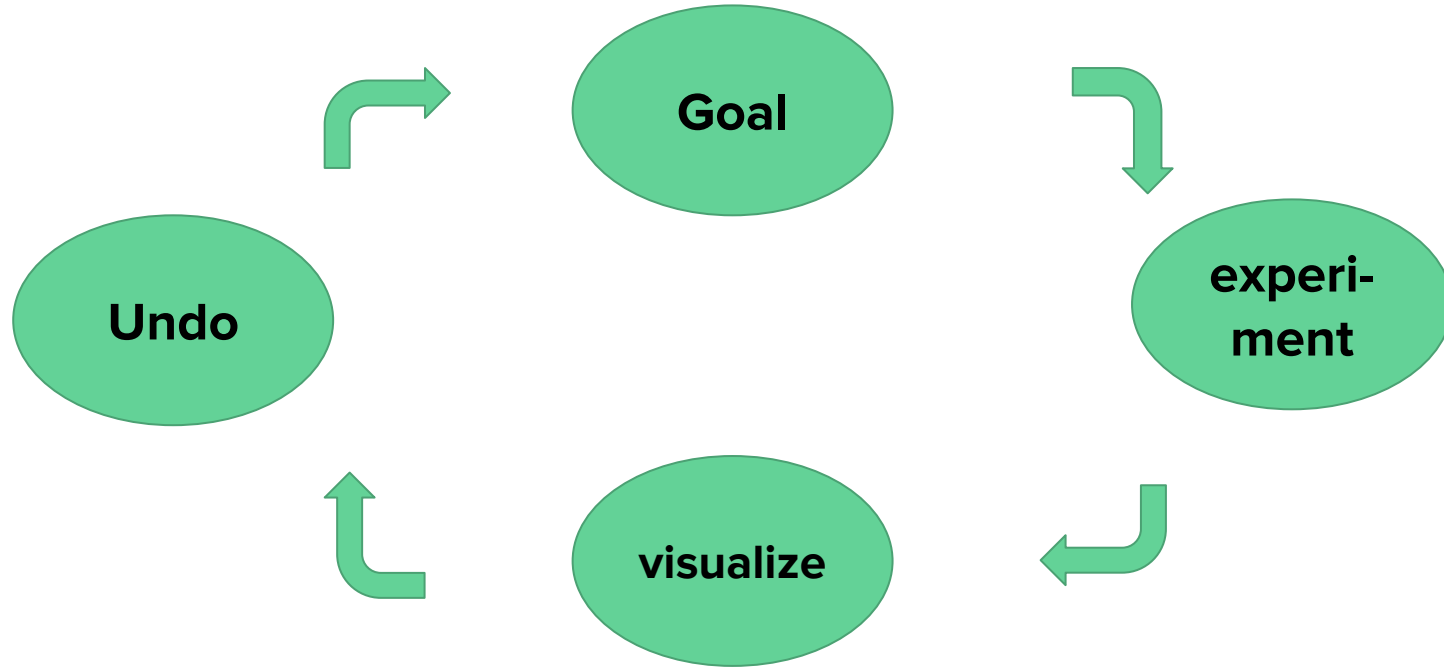
#micro-commit

#golden master

#IDE

Step 1 : acquérir le savoir faire

# #Mikado method



Step 1 : acquérir le savoir faire

# #Mikado method

#baby-step

**Undo**



**Goal**

#needs



**experiment**

#learn

#scrach  
refactoring

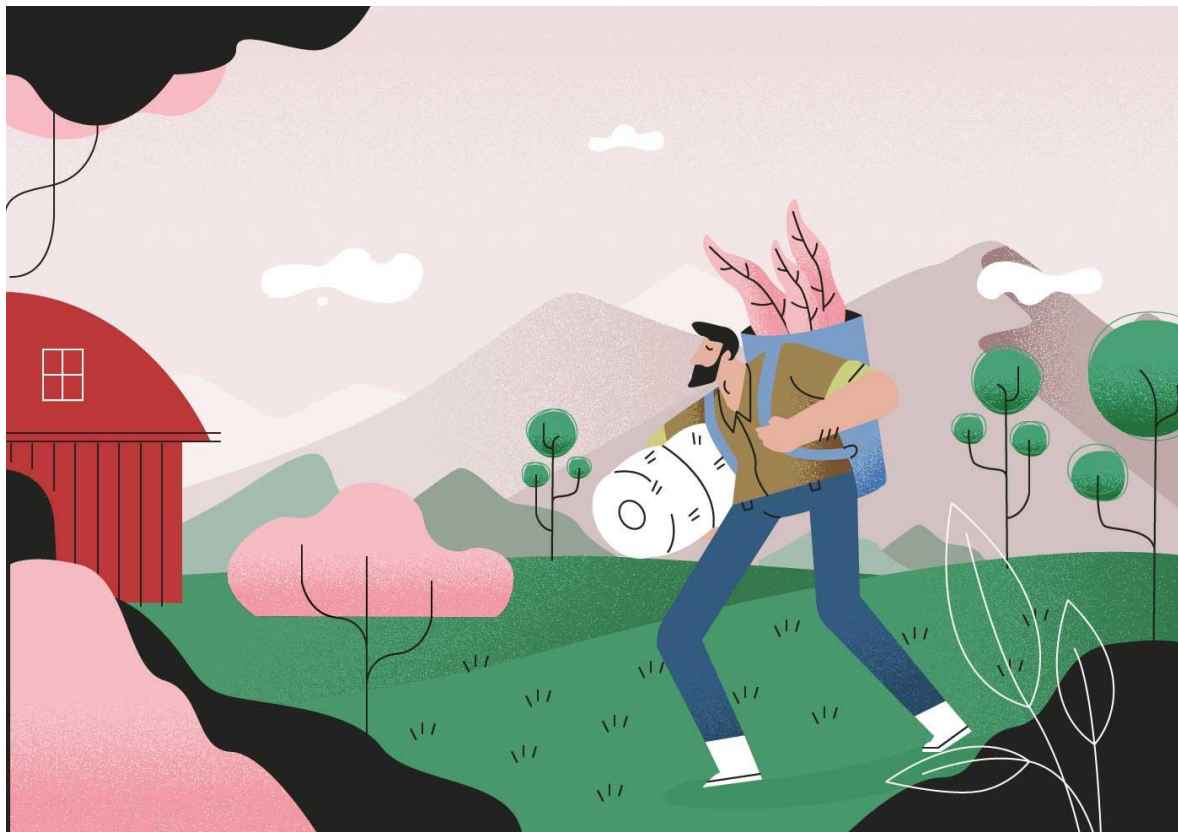


**visualize**



#focus

Step 1 : acquérir le savoir faire

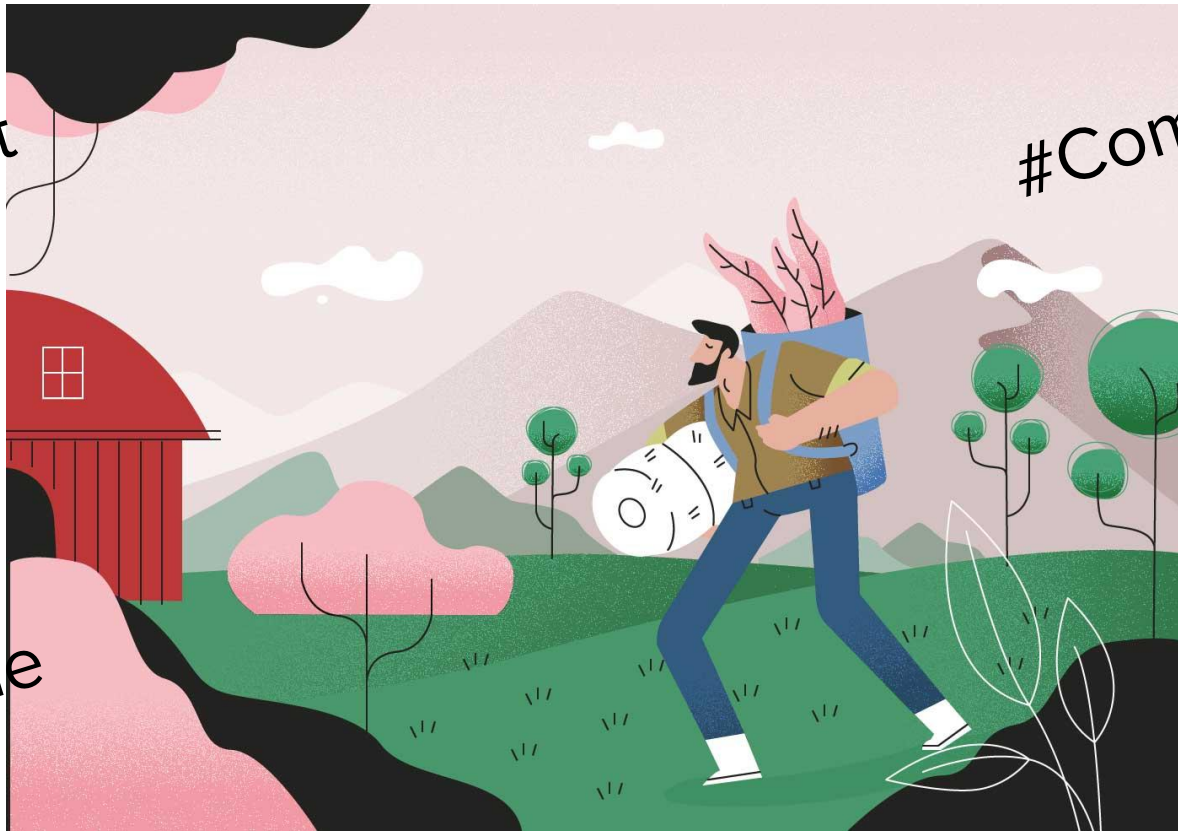


Step 2 : Acquérir le savoir-être

#Respect

#Compassion

#Discipline



Step 2 : Acquérir le savoir-être

#résilience

# Art of Refactoring

---



# Art of Refactoring

#boy-scout rule  
#baby-step  
#Respect  
#Kata  
#intuition  
#Design pattern  
#Nested If  
#adding function  
#Code smell  
#TDD  
#Mikado method  
#team-play  
#Compassion  
#value object  
#DDD  
#micro-commit  
#needs  
#strategy  
#primitive obsession  
#focus  
#KPI  
#adapter  
#learn  
#review  
#IDE  
#golden master  
#ROI  
#just do it  
#Discipline  
#risque  
#scrach  
#refactoring  
#feature envy  
#factory method  
#pas de sous-marin  
#Soutient hiérarchique  
#Architecture  
#résilience  
#Experience  
#lazy class  
#long terme

## références et inspirations :

- images : <https://www.vecteezy.com>
- mikado : <https://www.schibsted.pl/blog/zen-of-refactoring/>
- natural course of refactoring :  
<https://www.slideshare.net/BNSIT/natural-course-of-refactoring-mixit-lyon>
- Quand et pourquoi :  
<https://www.arolla.fr/blog/2018/06/quand-refactorer-et-pourquoi/>



On code ?

---