

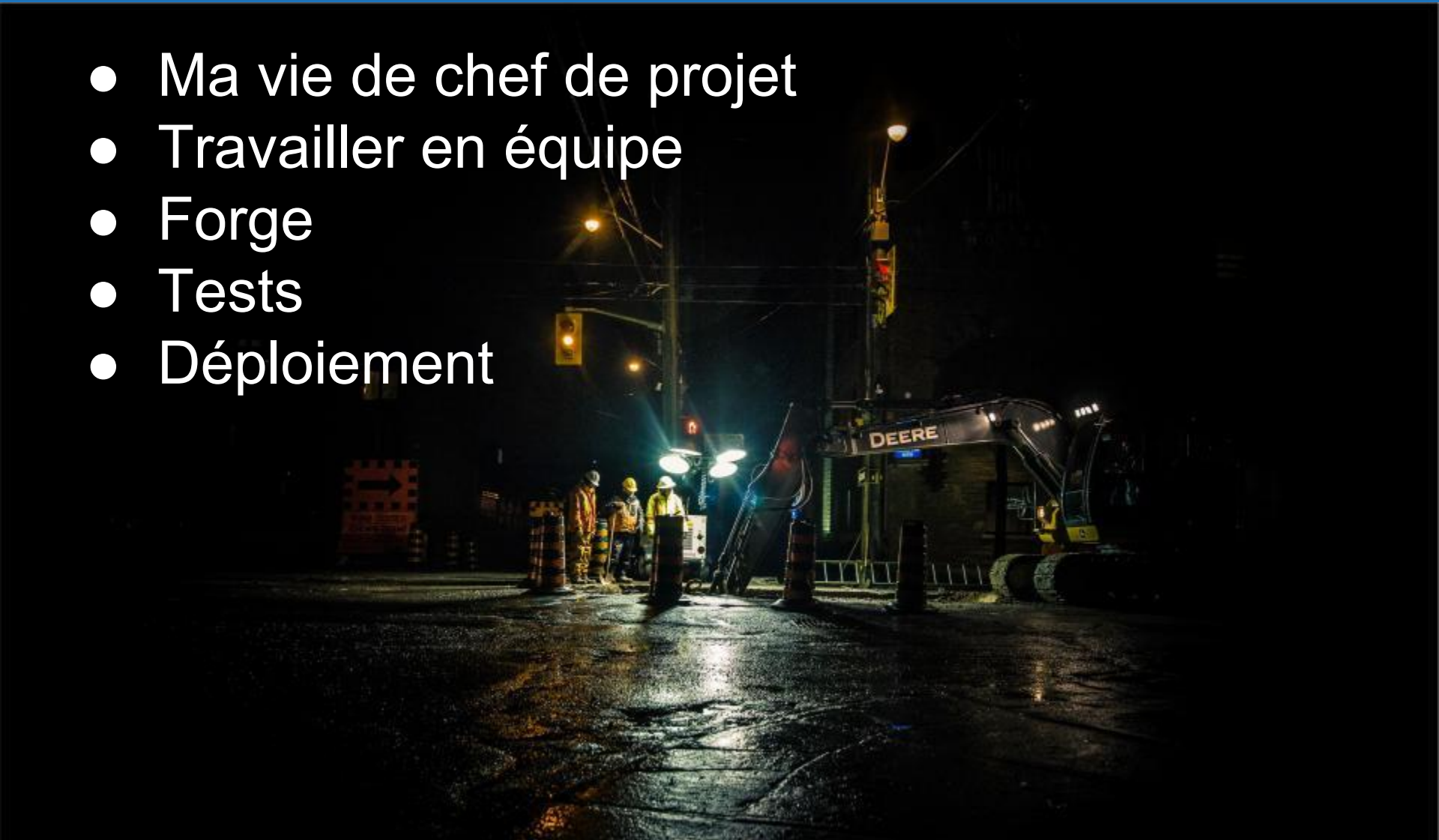
A nighttime photograph of a city skyline. In the center is a tall, dark building with many lit windows. To its left, a construction crane is visible. In the foreground, a bridge with a metal railing and lights runs diagonally across the frame. The sky is dark with some clouds.

# Autour de la webapp

Forge logicielle

# Forge logicielle - TDD

- Ma vie de chef de projet
- Travailler en équipe
- Forge
- Tests
- Déploiement



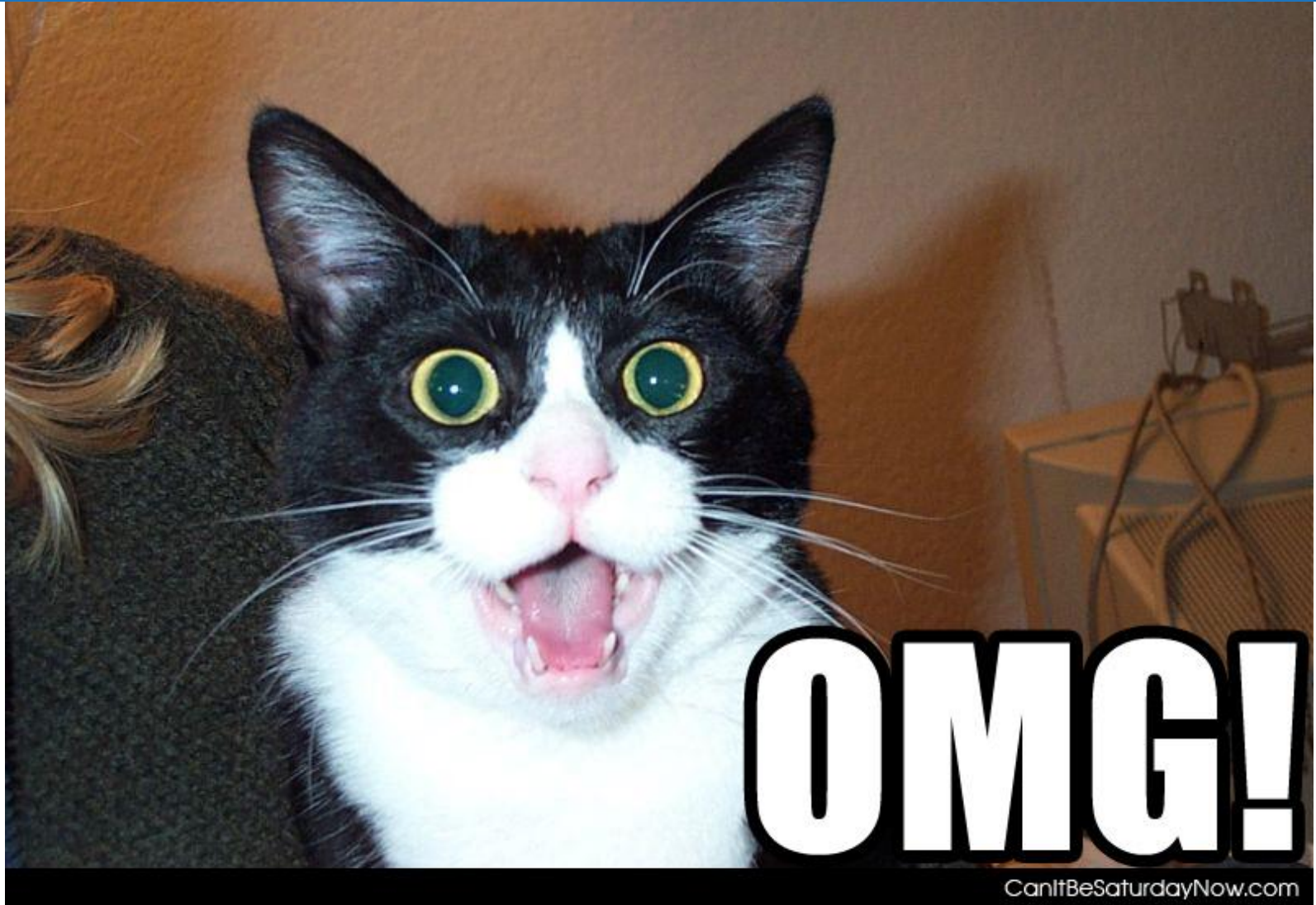
# Ma vie de chef de projet



“J’ai une super idée, on va se faire une machine à cambrer les bananes! On va défoncer la concurrence!”



# Ma vie de chef de projet

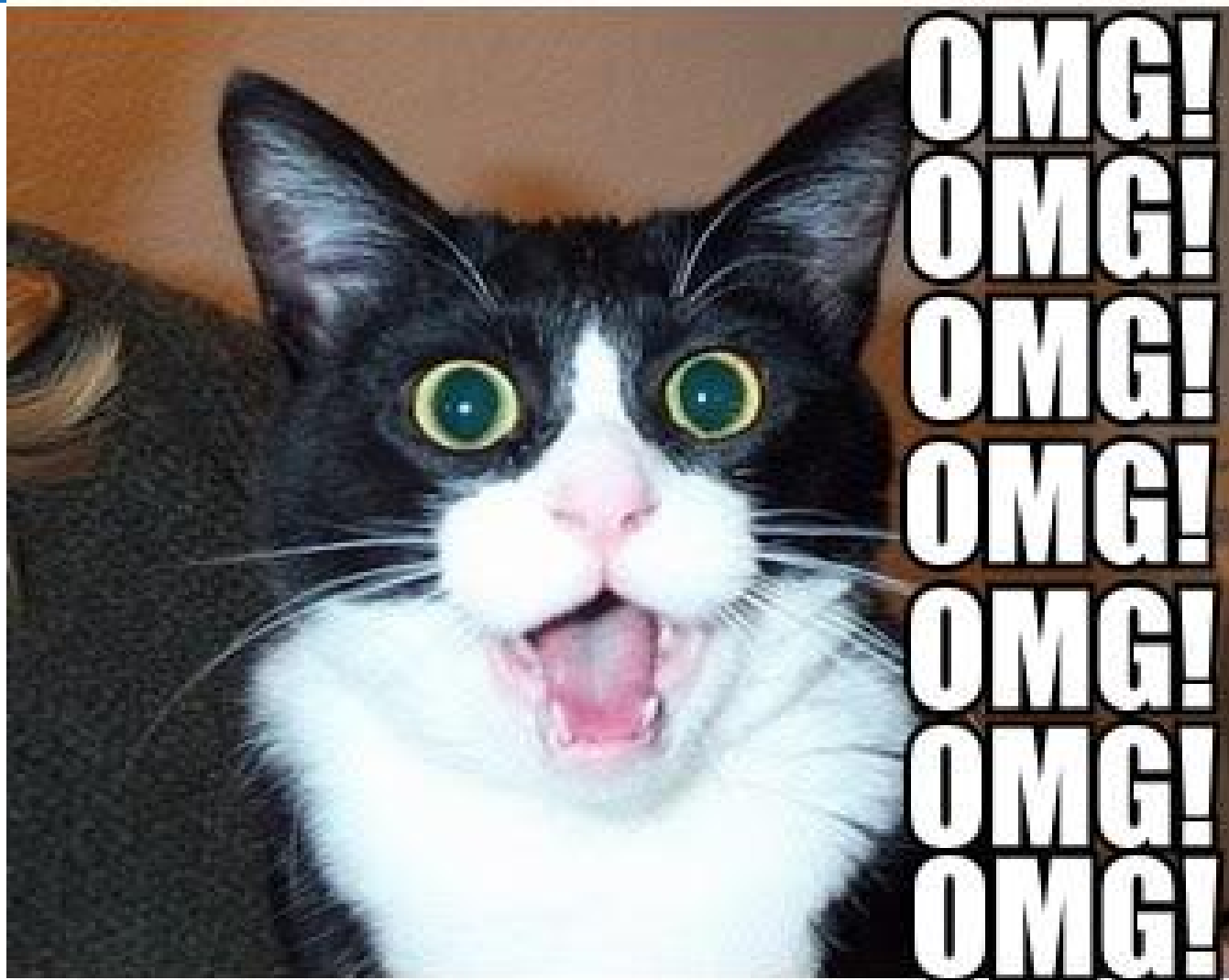


# Ma vie de chef de projet



“et on a la campagne d’octobre qui commence en septembre, il nous faut cette machine!”

# Ma vie de chef de projet



# Ma vie de chef de projet

## L'inventaire :

- un chiffrage
- s'organiser
  - des développeurs ressources
  - planifier
  - suivre
- construire
  - tester
  - mesurer
  - corriger
- livrer

# Ma vie de chef de projet

**CLIENT**

« Un cahier des charges ? Pas la peine, faites directement un site qui permet tout, comme ça je ne vous dérange plus. »

@webAgencyFAIL

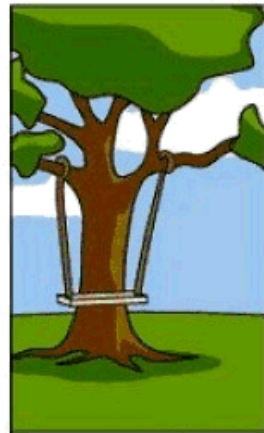
PIWEE



# Ma vie de chef de projet



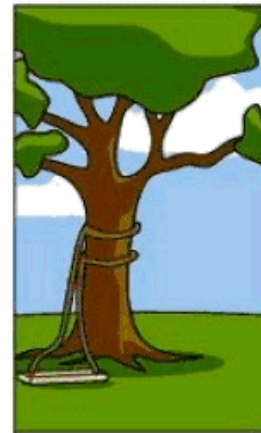
How the customer  
explained it



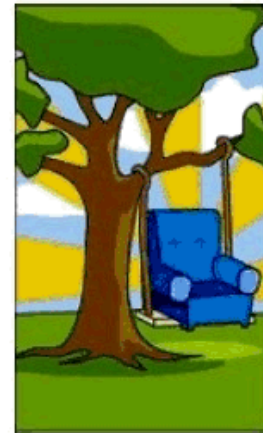
How the project leader  
understood it



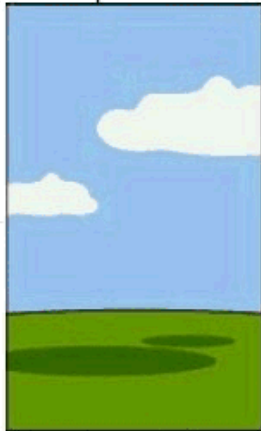
How the engineer  
designed it



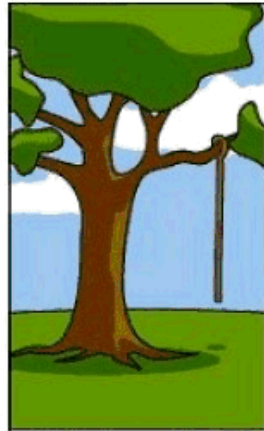
How the programmer  
wrote it



How the sales  
executive described it



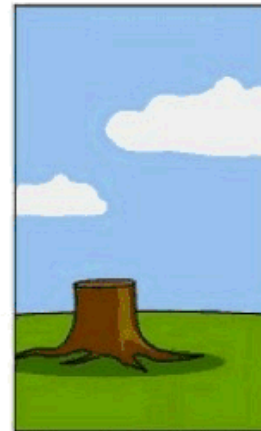
How the project was  
documented



What operations  
installed



How the customer  
was billed



How the helpdesk  
supported it



What the customer  
really needed

# Ma vie de chef de projet

## Approche Lean

Not like this....



1



2



3



4

Like this!



1



2



3

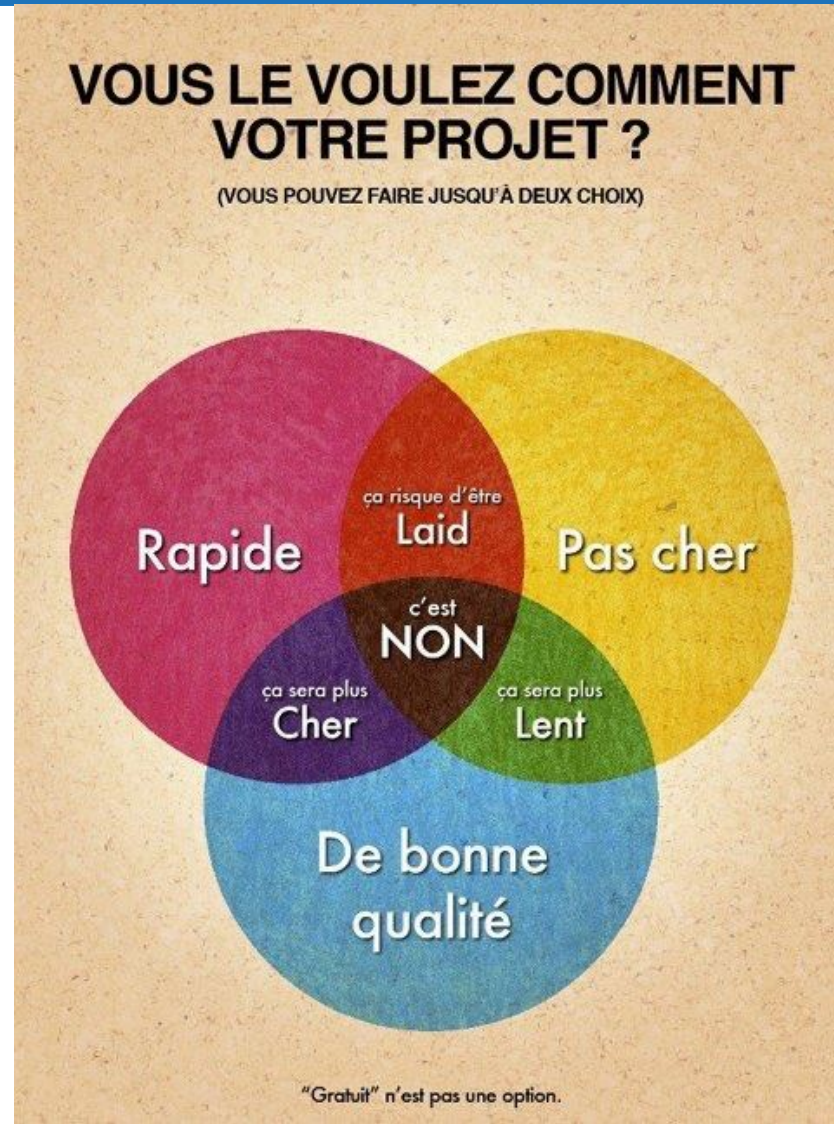


4



5

# Ma vie de chef de projet



# Ma vie de chef de projet

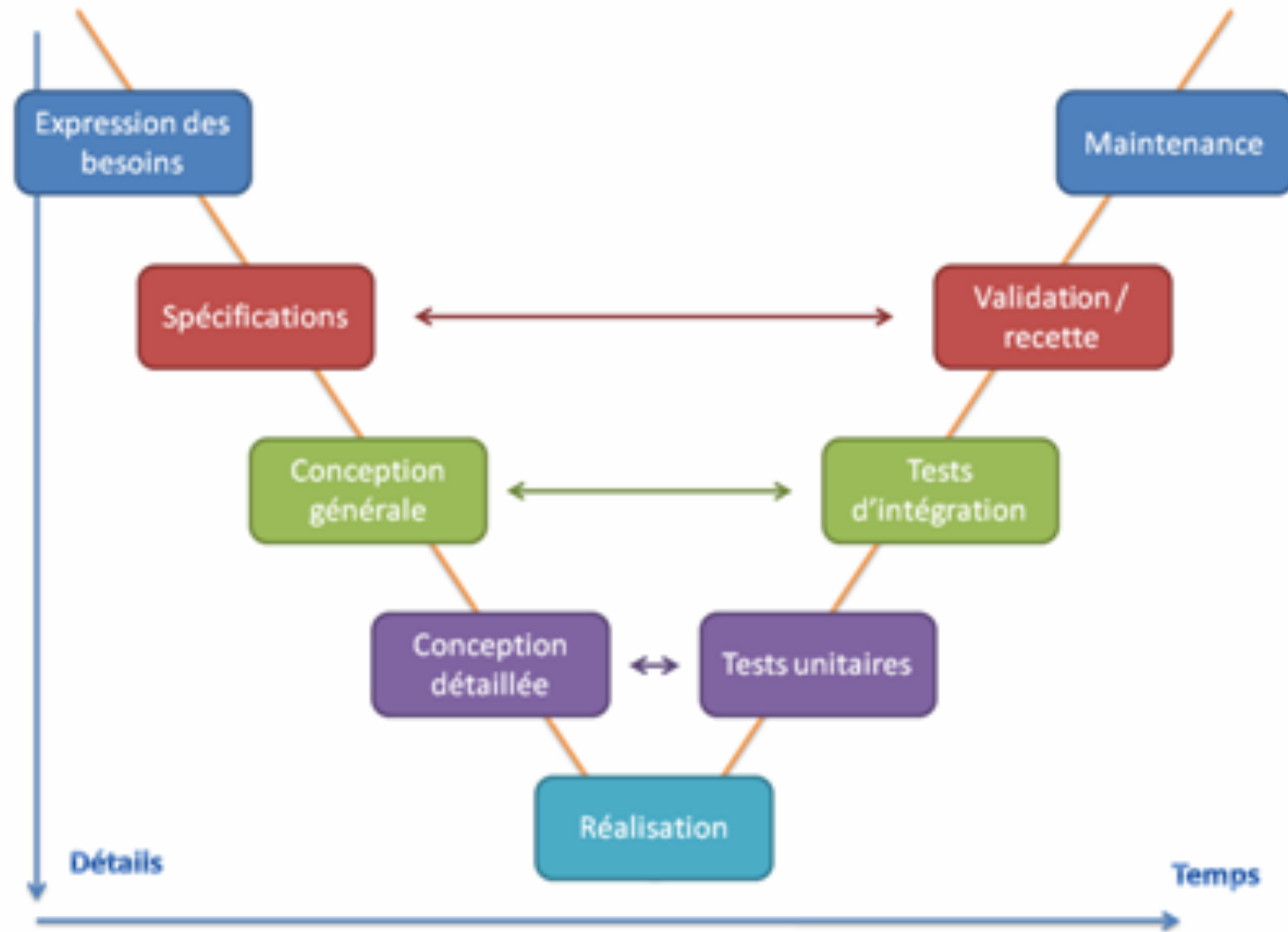
Chiffrer = faire un devis

... Cycle en V “classique”

- comparer avec des projets similaires
- découper en tâches atomiques
- appliquer des ratios (des abaques peuvent exister)
  - pilotage
  - recette
  - tests
  - documentations
- Effet tunnel



# Ma vie de chef de projet



# Ma vie de chef de projet

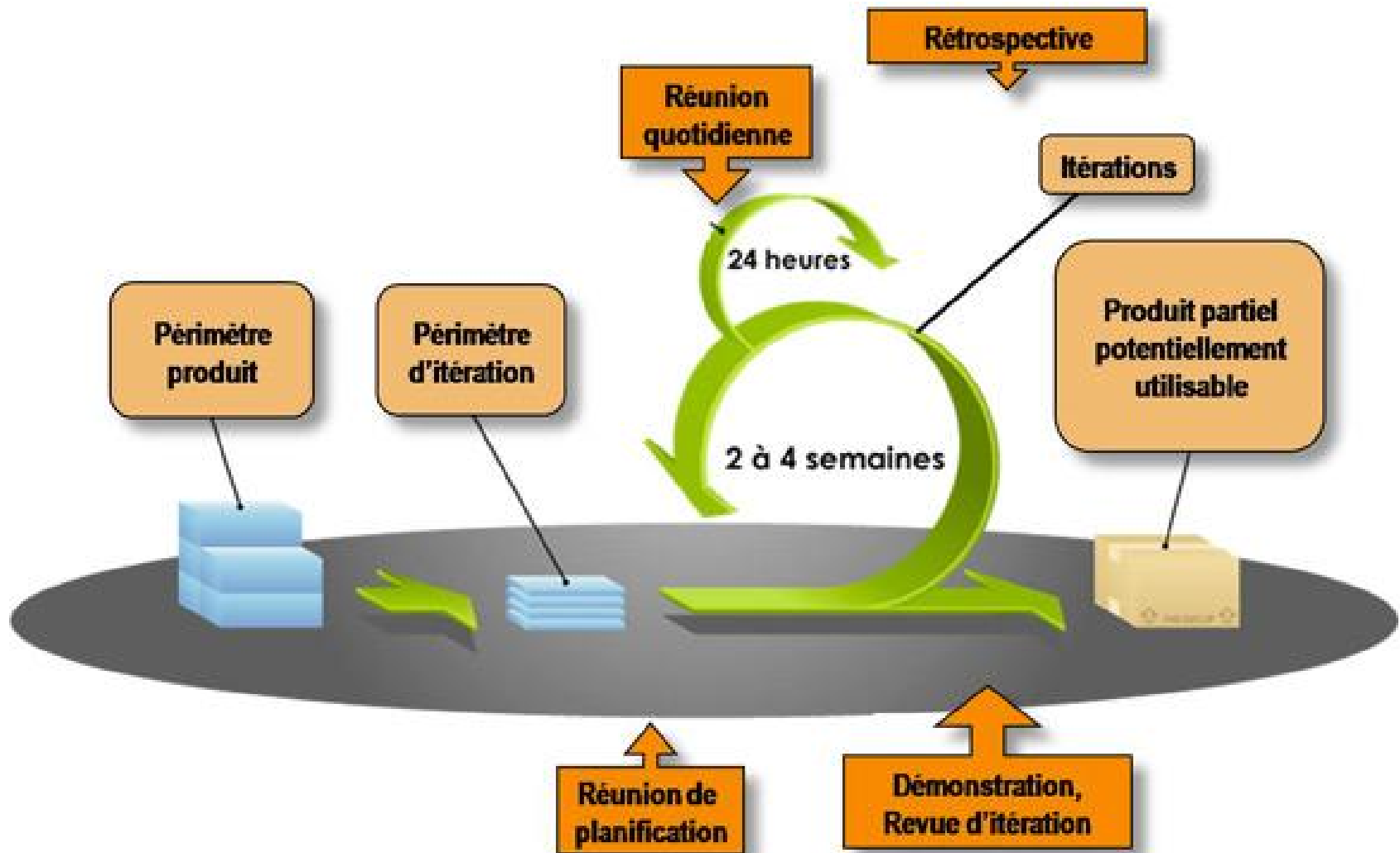
Chiffrer = faire un devis

... méthode agile = périmètre fixe

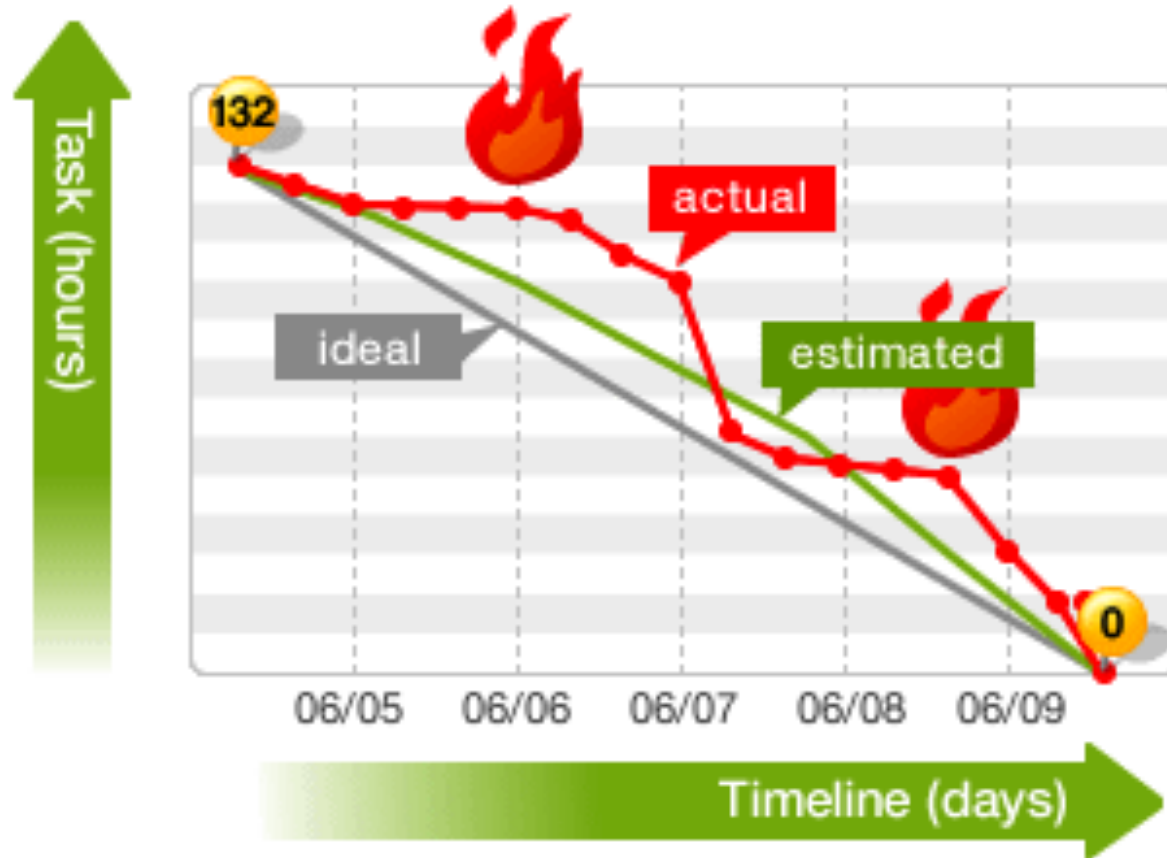
- découper en tâches atomiques
- catégoriser par valeur business
- chiffrer en points de complexité
- impliquer le client
- beaucoup de refactoring
- risque de manque de recul



# Ma vie de chef de projet



# Ma vie de chef de projet





# Ma vie de chef de projet

Client peu présent, peu mature -> Cycle en V

Client impliqué et réceptif -> Agile

Un chiffrage c'est dur à faire

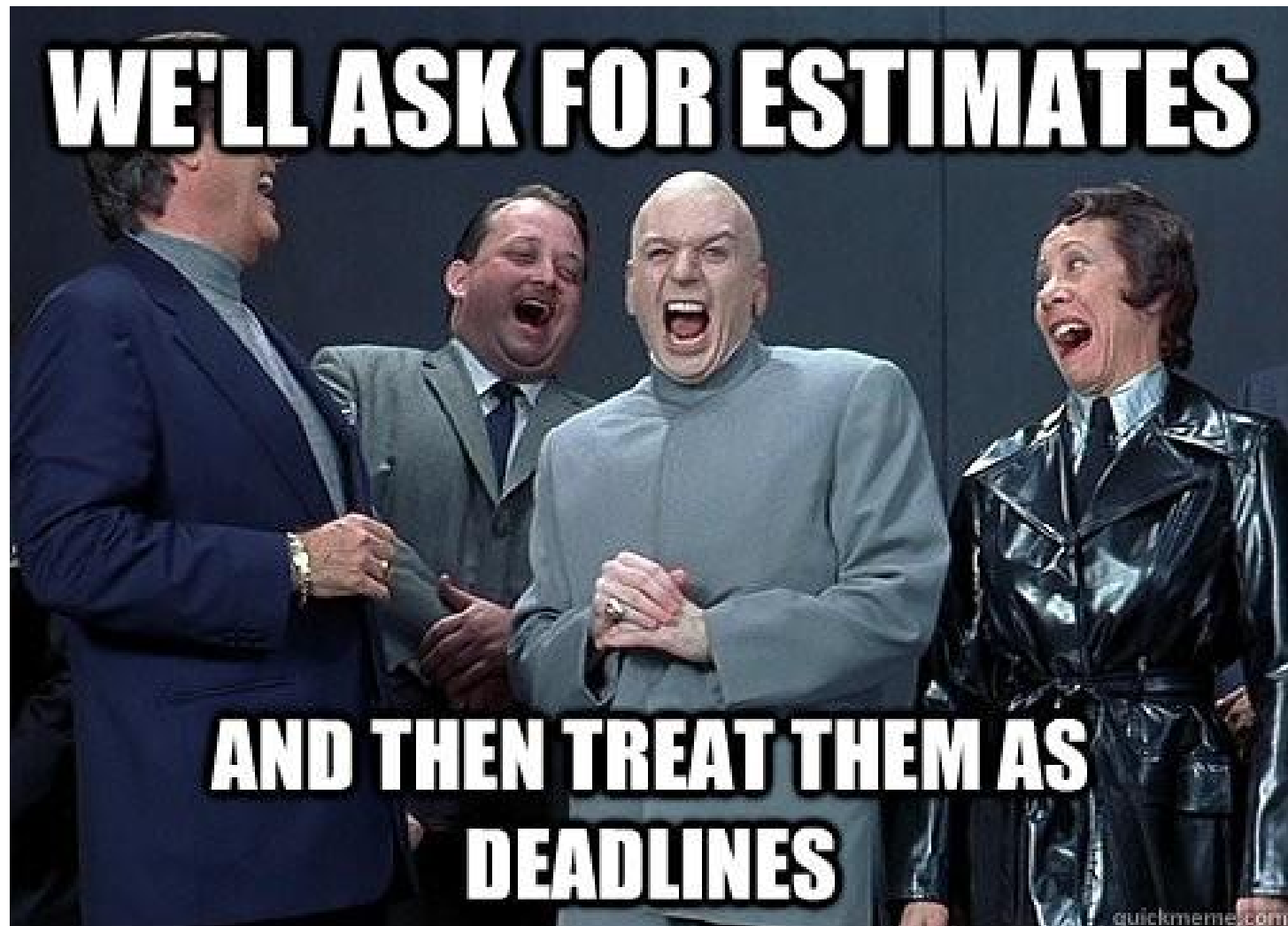
- abaques
- expérience
- analogie
- découpage en tâches atomiques

# Ma vie de chef de projet

découper en tâches atomiques



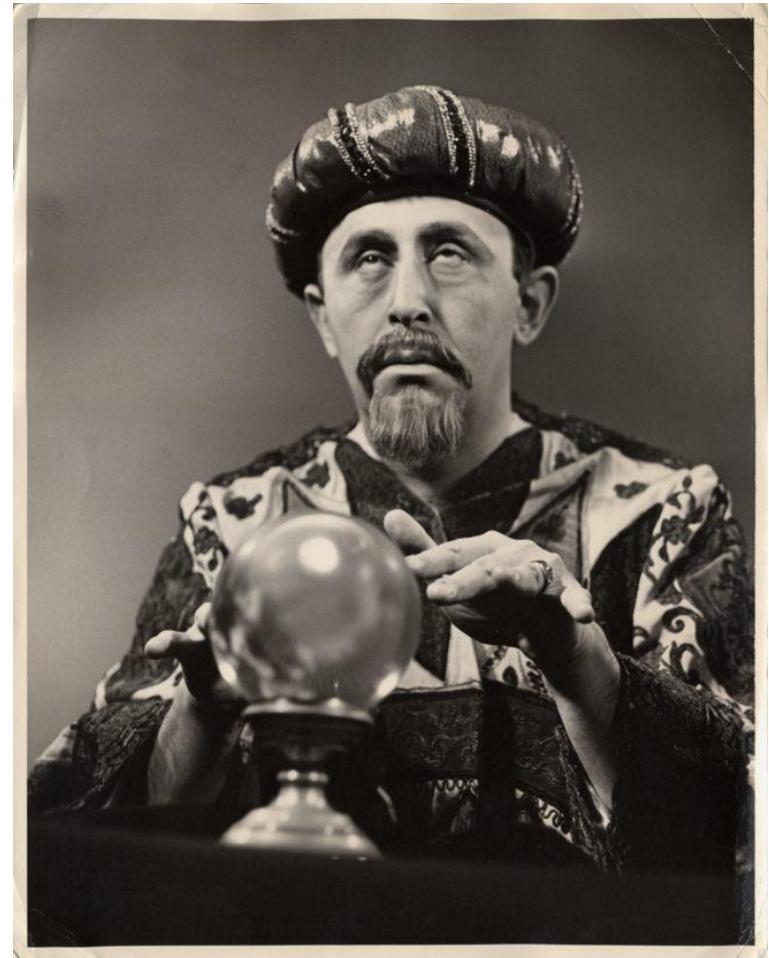
# Ma vie de chef de projet



# Ma vie de chef de projet


Les estimations ne sont que des estimations

- suivi des consommés
- maîtrise des risques
  - plans d'actions
- pourcentage d'erreur
  - indisponibilités
  - problèmes techniques
  - ...





# Travailler en équipe - Forge

- 
- A group of people are working together at a wooden table. There are laptops, notebooks, and a glass of iced tea on the table. The background is a solid blue color.
- Référentiel exigences
  - SCM
  - Documentation
  - Construction
  - Tests
  - Déploiements
  - Bug tracker

# Travailler en équipe - Forge

Une forge?

- Amélioration de la qualité des logiciels
- Amélioration de la traçabilité : du développement à la production
- Garantir la pérennité
- Amélioration de la productivité des développements

# Travailler en équipe - Forge

Un référentiel des exigences

- tracer les demandes client
- lister les epics / user-stories
- lier le code à la user-story et inversement
- aide à définir et comprendre le besoin du client
- priorisation des développements

Jira / IceScrum / ...

# Travailler en équipe - SCM

- Source Control Management

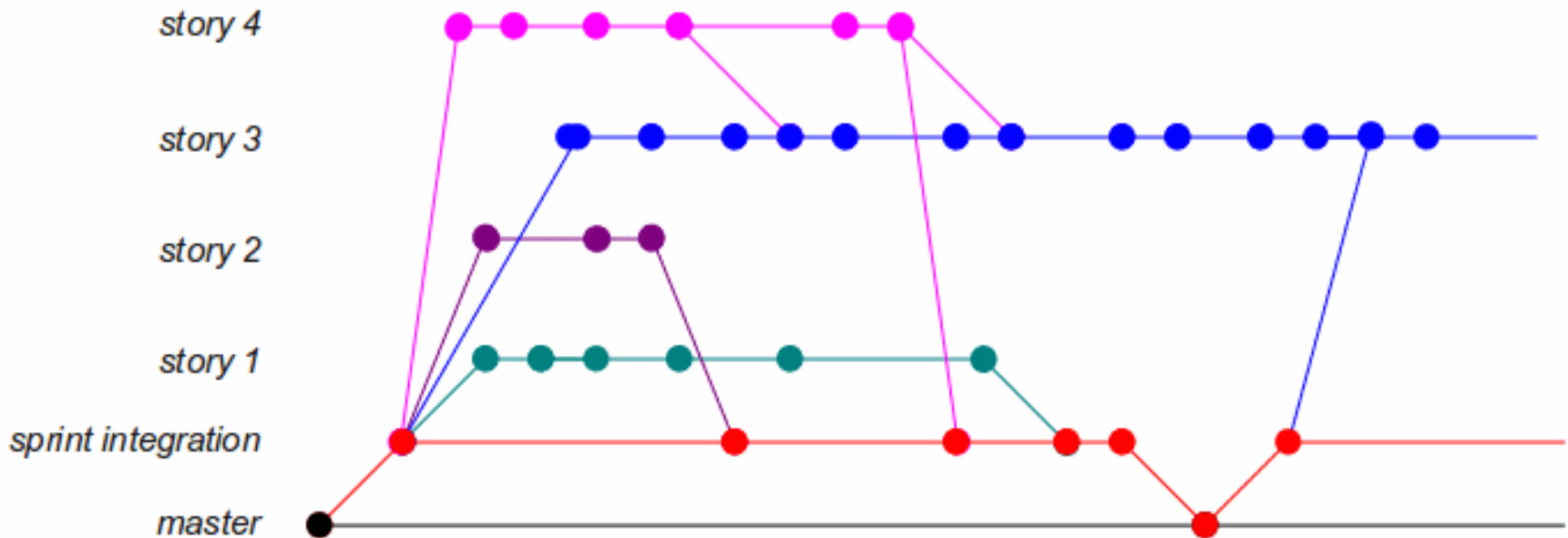
- Git
- Subversion
- CVS
- Mercurial
- Clearcase
- ...





# Travailler en équipe - SCM

- Branches
- Tags
- Commits



# Travailler en équipe - SCM

- Versionning (SEMVER)



# Travailler en équipe - Doc

- JavaDoc / JSDoc / APIdoc
- Wiki

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”  
– Martin Fowler

“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.”  
– Martin Golding

# Travailler en équipe - Construire

- Gestion de dépendances

Hey Jean-Claude, il paraît que tu as développé une super lib pour requêter le web-service...

Pas de soucis, je te l'envoie par mail

Et il y a besoin de quelles autres libs pour la faire fonctionner ?

HTTPClient, Jackson, euuuuh, Log4j, et euuuuh

Quelles versions?

... j'ai une réunion, j'te laisse

# Travailler en équipe - Construire

- Gestion de dépendances

Hey Jean-Claude, il paraît que tu as mis à jour ta super lib ...

Oui, hier, tu as quelle version?

la 0.2.3.00.25

Ah, on en est à la 5.25.665.14 ...

les librairies dépendantes sont à monter en version aussi?

... j'ai une réunion, j'te laisse



# Travailler en équipe - Construire

- Gestion de dépendances
- Archétypes
- Java
  - Dépôts Maven
    - Maven
    - Ivy
    - Gradle
- Javascript
  - NPM (Node Package Manager)
  - Yeoman
  - Bower (Basé sur Github)

**maven**



# Travailler en équipe - Construire

```
<project>
...
<dependencies>
  <dependency>
    <groupId>group-a</groupId>
    <artifactId>artifact-a</artifactId>
    <version>1.0</version>
    <exclusions>
      <exclusion>
        <groupId>group-c</groupId>
        <artifactId>excluded-artifact</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>group-a</groupId>
    <artifactId>artifact-b</artifactId>
    <version>1.0</version>
    <type>bar</type>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</project>
```

The Maven logo, featuring the word "maven" in a bold, italicized sans-serif font. The letter "a" is orange, while the other letters are black.

# Travailler en équipe - Construire



```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    apt "org.androidannotations:androidannotations:$AAVersion"  
    compile "org.androidannotations:androidannotations-api:$AAVersion"  
    compile 'com.android.support:appcompat-v7:22.1.1'  
    compile 'com.google.android.gms:play-services:7.3.0'  
    compile 'com.android.support:support-v4:22.1.1'  
    compile 'org.apache.commons:commons-lang3:3.4'  
    compile 'com.google.code.gson:gson:2.3.1'  
    compile 'com.android.support:cardview-v7:21.0.3'  
    compile('au.com.datasymphony:EasyFlow:1.3.1') {  
        exclude group: 'junit'  
        exclude group: 'slf4j-api'  
        exclude group: 'slf4j-log4j12'  
    }  
}
```

# Travailler en équipe - Construire

- Gestion de dépendances
- Dépôt public sur le Web
- Cache d'entreprise (optionnel)
- Cache local
- Résolution des dépendances transitives ou non
- SEMVER



# Travailler en équipe - Construire

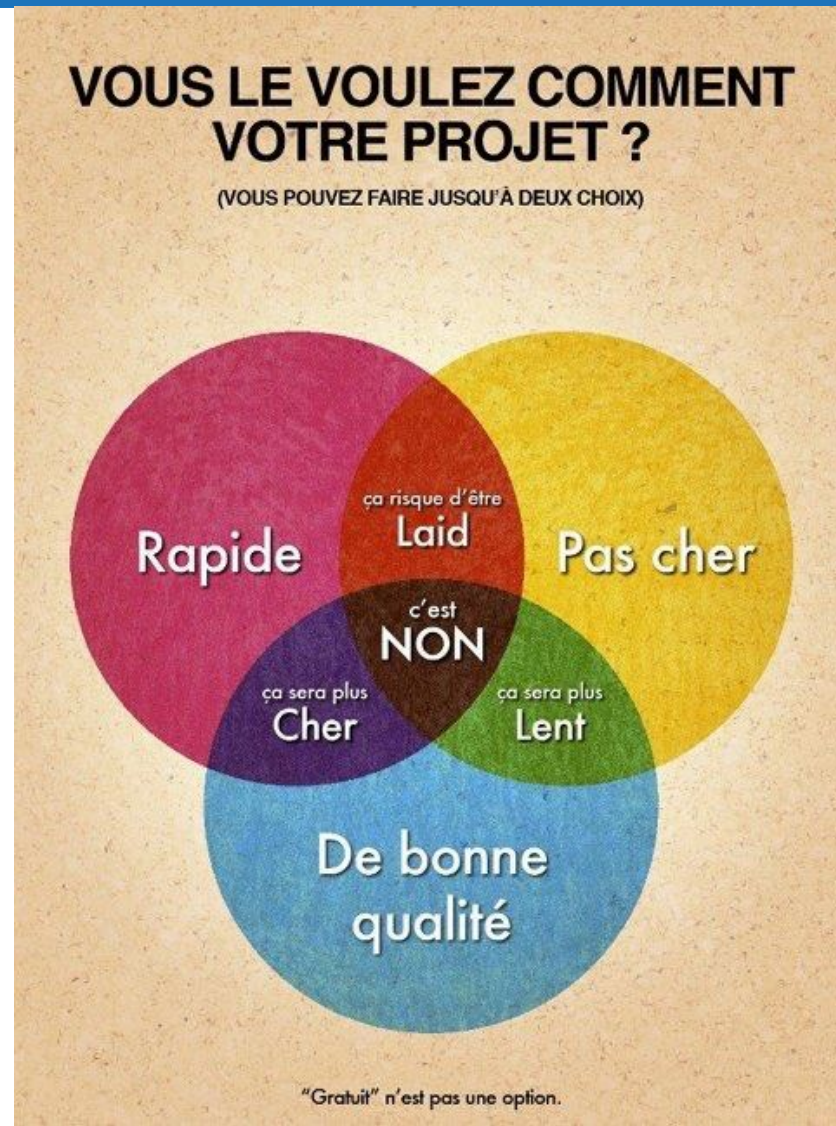
- Compiler / assembler
- Java
  - Maven
  - Gradle
  - Ant
  - ...
- Javascript
  - Gulp
  - Grunt

**maven**





# Travailler en équipe - Qualité

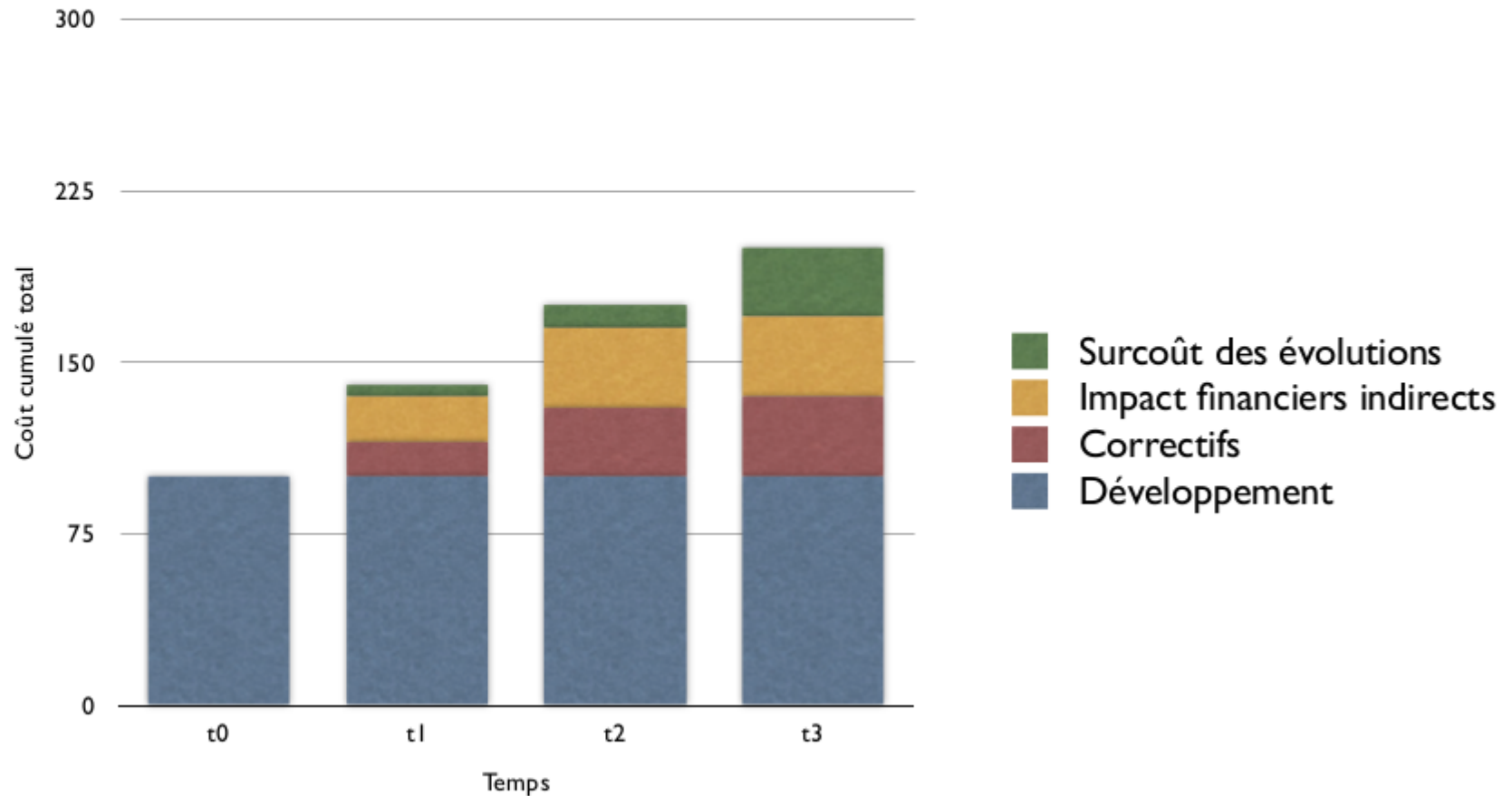


# Travailler en équipe - Qualité

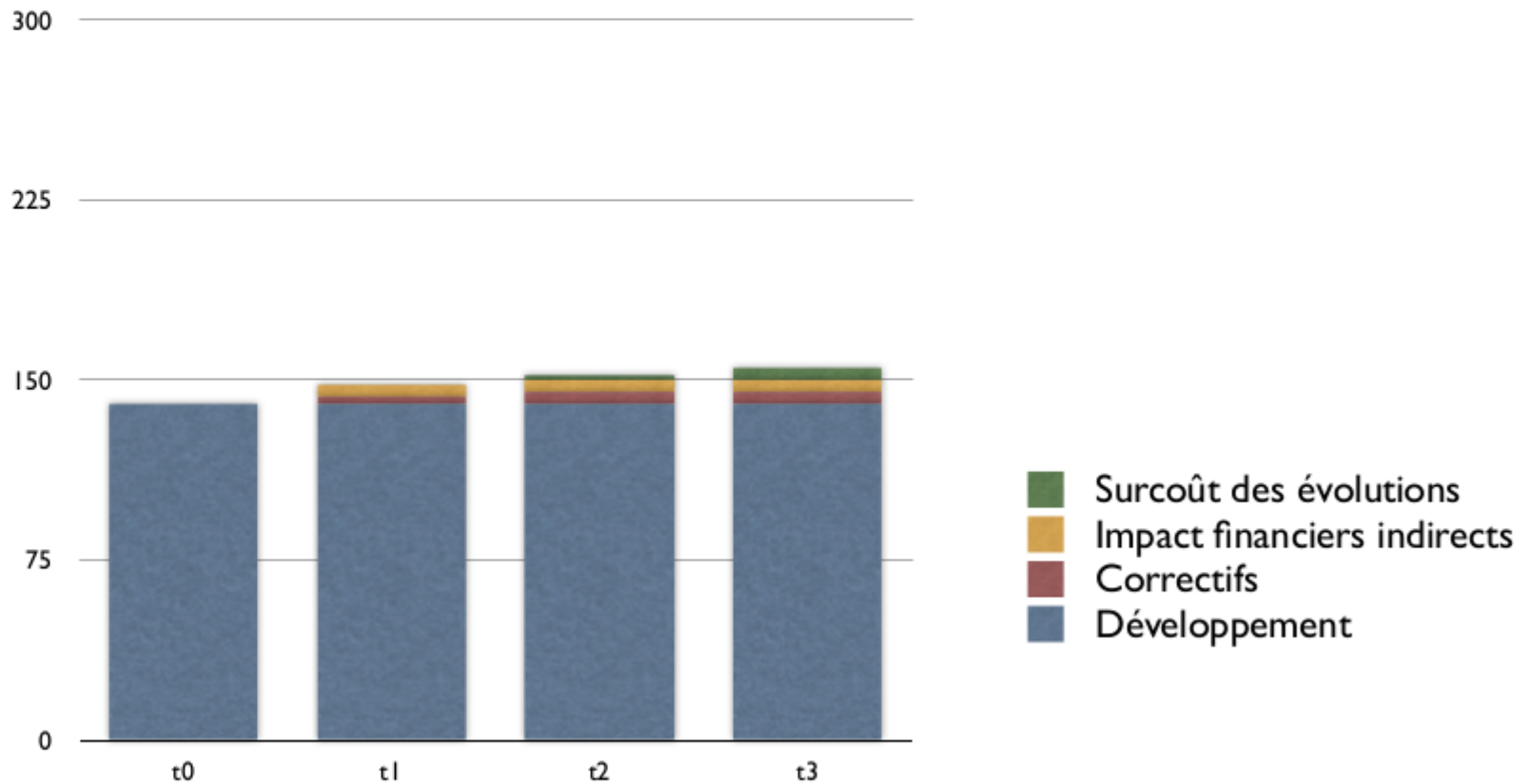
“La qualité fait chuter ma productivité”



# Travailler en équipe - Qualité

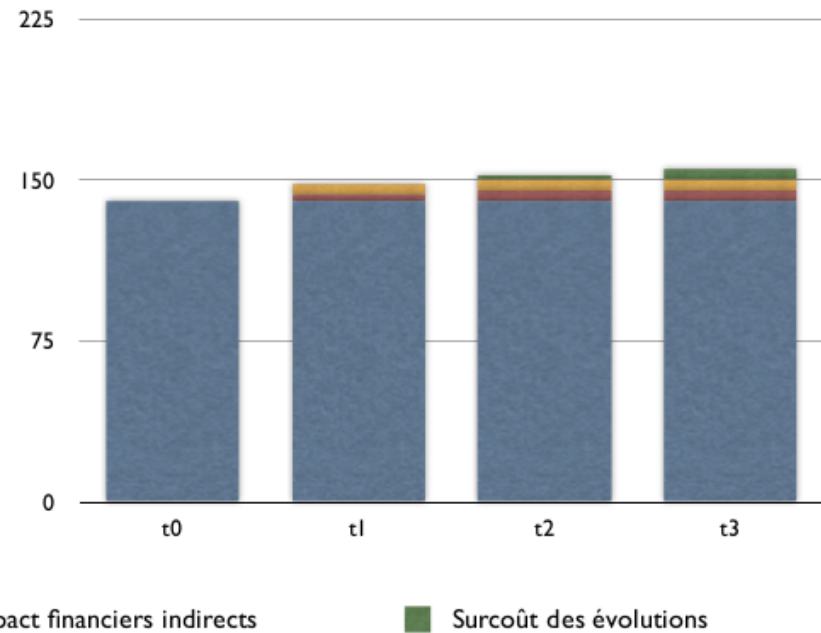
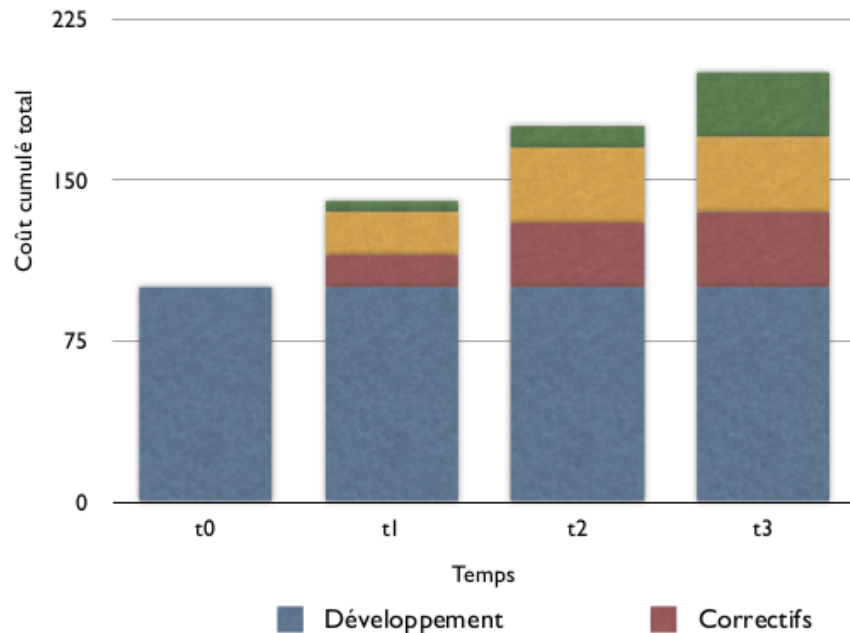


# Travailler en équipe - Qualité



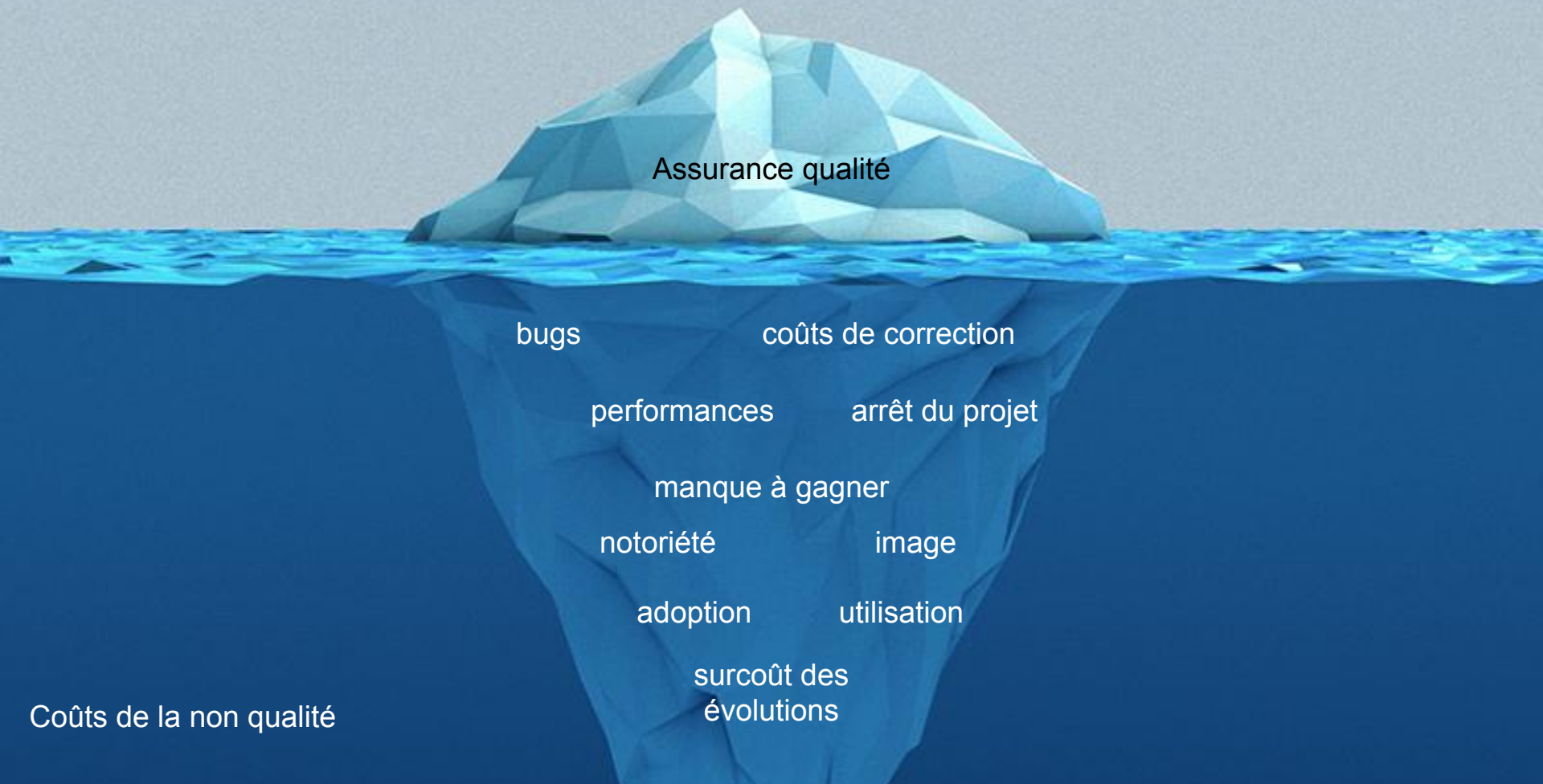
# Travailler en équipe - Qualité

Qui a été le plus productif?



# Travailler en équipe - Qualité

Coûts de la qualité



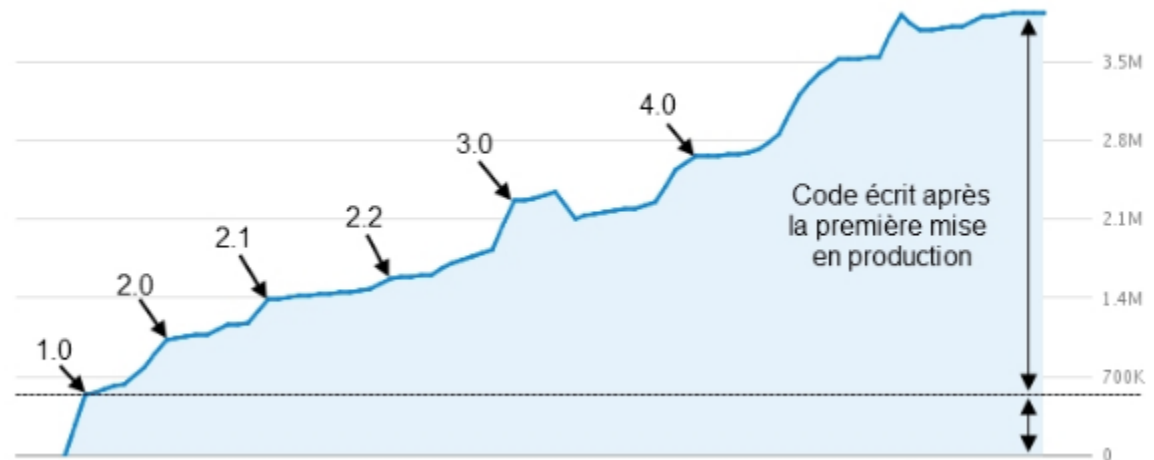
Coûts de la non qualité



# Travailler en équipe - Qualité

Votre logiciel va évoluer :  
60% à 80% des coûts en maintenance  
dont 75% évolutive

Source: Software Maintenance, par G. Ganfora



# Travailler en équipe - Qualité

- Augmenter la durée de vie de nos applications :
  - Améliorer la maintenabilité
  - Améliorer la documentation
  - Réduire le code dupliqué
  - Respecter des pratiques communes de développement
- Détecter au plus tôt certains bugs classiques
- Améliorer les performances par la détection d'anti-patterns

# Travailler en équipe - Qualité

## Pair programming

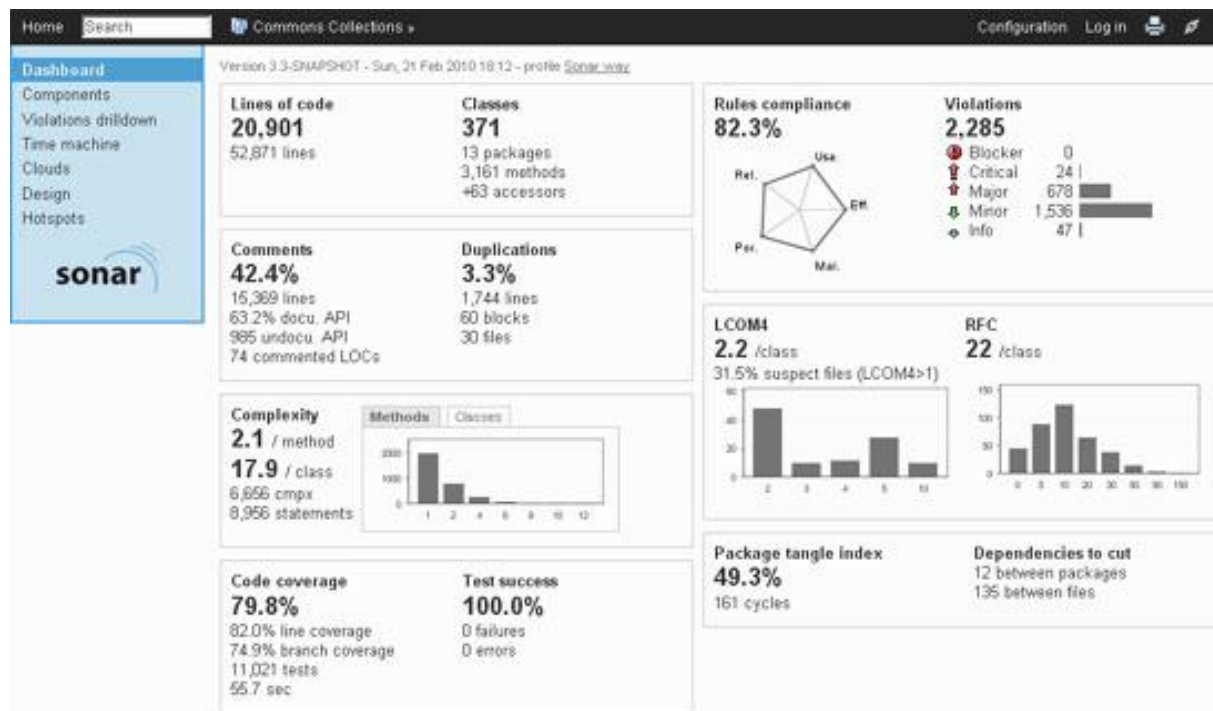
Tu as oublié un  
point-virgule...



C'est du Groovy !

# Travailler en équipe - Qualité

## Analyse statique / dynamique (Sonar)



# Travailler en équipe - Qualité

## Analyse statique / dynamique (Sonar)

- **Apports**

- Vérification des pratiques de codages
- Détection de bugs
- Référentiel qualité du code source

- **Non apports**

- Ne valide pas la performance de l'application
- Ne valide pas la sécurité
- Ne valide pas la capacité du logiciel à répondre au besoin fonctionnel

# Travailler en équipe - Qualité

## Revue de code (Crucible)

The screenshot displays the Crucible web interface for a code review. The top navigation bar includes links for Dashboard, Source, Projects, People, and Reviews, along with a user profile for Edwin Dawson and a search bar. The main header shows the project name 'Testing Project > TEST-75' and the review title 'Blink Java Example Review', with the author 'Edwin Dawson' and creation date '23 May 2010'. A sidebar on the right offers options to 'Create Snippet' and 'Tools'. The central area contains a Java code snippet for a 'Blink' applet. To the right of the code, a comment thread is visible. The first comment, by Brendan Humphreys, suggests 'consider using HTML5'. The second comment, by Seb Ruiz, points out a potential failure when 'blinkFrequency' is an empty string and suggests pulling the logic out into an 'if' statement for better readability. This second comment is marked as a 'Defect'.

Dashboard Source Projects People **Reviews** ☆ Edwin Dawson Search

Testing Project > TEST-75 **Blink Java Example Review** + Create Snippet ⚙ Tools

Author: Edwin Dawson Created: 23 May 2010

Click on source lines to add an inline comment.

```
01. import java.awt.*;
02. import java.util.*;
03.
04. public class Blink extends java.applet.Applet {
05.     private Timer timer;           // Schedules the blinking
06.     private String labelString;    // The label for the window
07.     private int delay;             // the delay time between blinks
08.
09.     public void init() {
10.         String blinkFrequency = getParameter("speed");
11.         delay = (blinkFrequency == null) ? 400 :
12.             (1000 / Integer.parseInt(blinkFrequency));
13.         labelString = getParameter("lbl");
14.         if (labelString == null)
15.             labelString = "Blink";
16.         Font font = new java.awt.Font("Serif", Font.PLAIN, 24);
17.         setFont(font);
18.     }
19.
20.     public void start() {
21.         timer = new Timer();        //creates a new timer to schedule the blinking
22.         timer.schedule(new TimerTask() { //creates a timertask to schedule
23.             // overrides the run method to provide functionality
24.             public void run() {
25.                 repaint();
26.             }
27.         });
28.     }
29. }
```

**Brendan Humphreys:** 19:31  
consider using HTML5  
[Reply](#)

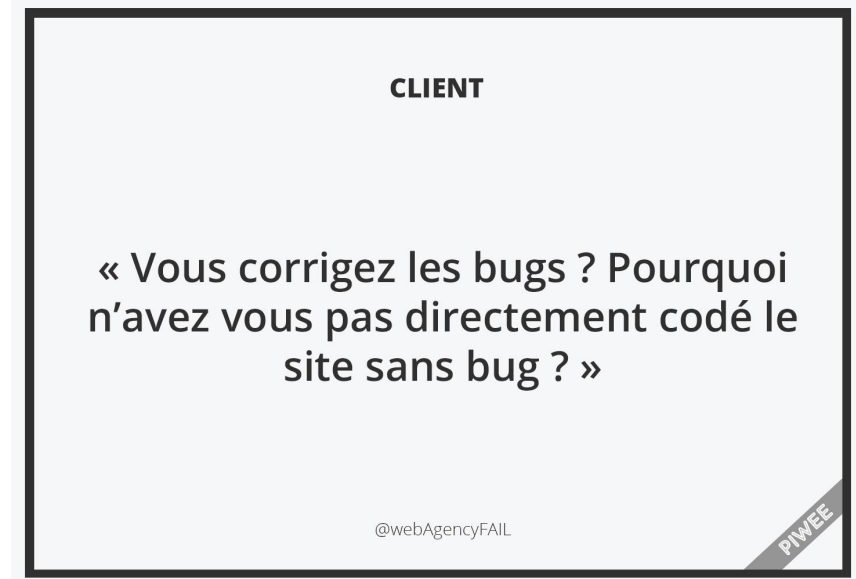
**Seb Ruiz:** 19:34  
Absolutely  
[Reply](#)

**Seb Ruiz:** 19:32  
This will fail when `blinkFrequency` is an empty string. Consider pulling this out into an if statement for greater readability.  
[Reply](#) **Defect**



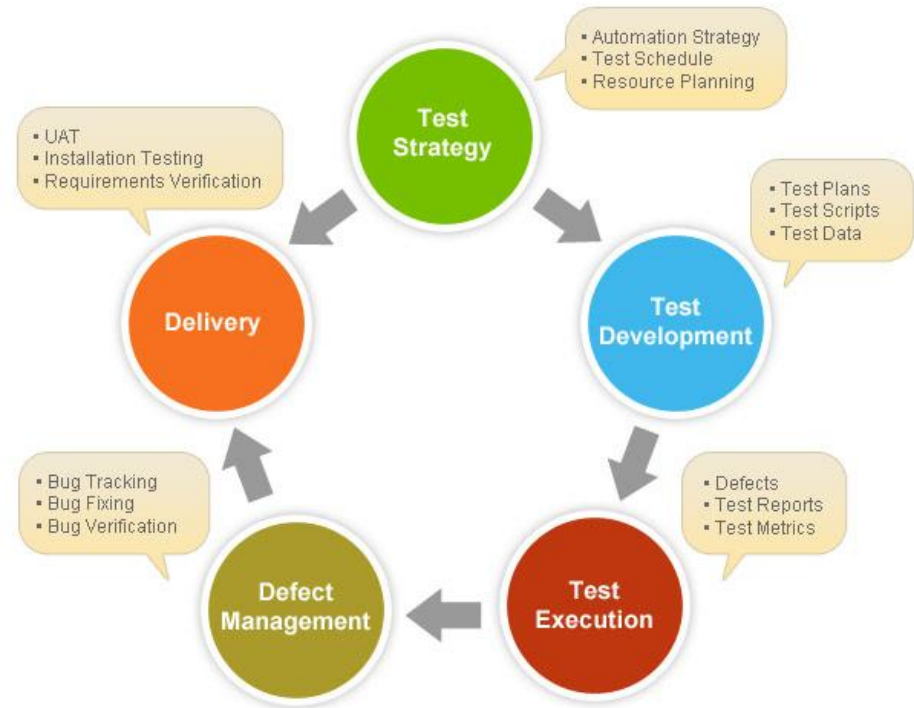
# Travailler en équipe - Tester

- Tests unitaires
  - Junit
  - Selenium
  - Karma
  - ...
- Tests end to end
- Tests fonctionnels
- Bug trackers
  - Jira
  - Mantis
  - ...



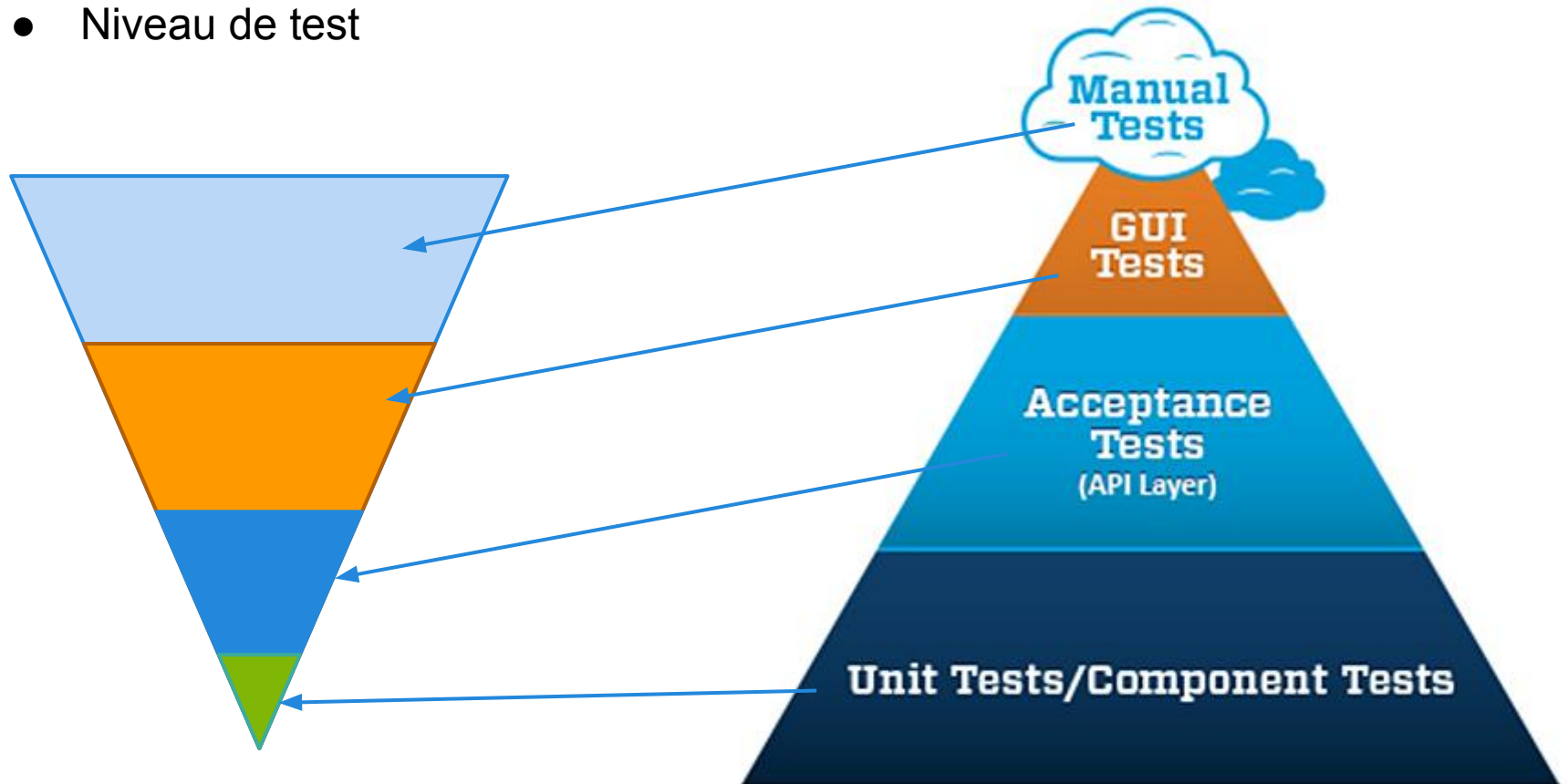
# Travailler en équipe - Tester

- Tests unitaires
  - Junit
  - Selenium
  - Karma
  - ...
- Tests end to end
- Tests fonctionnels

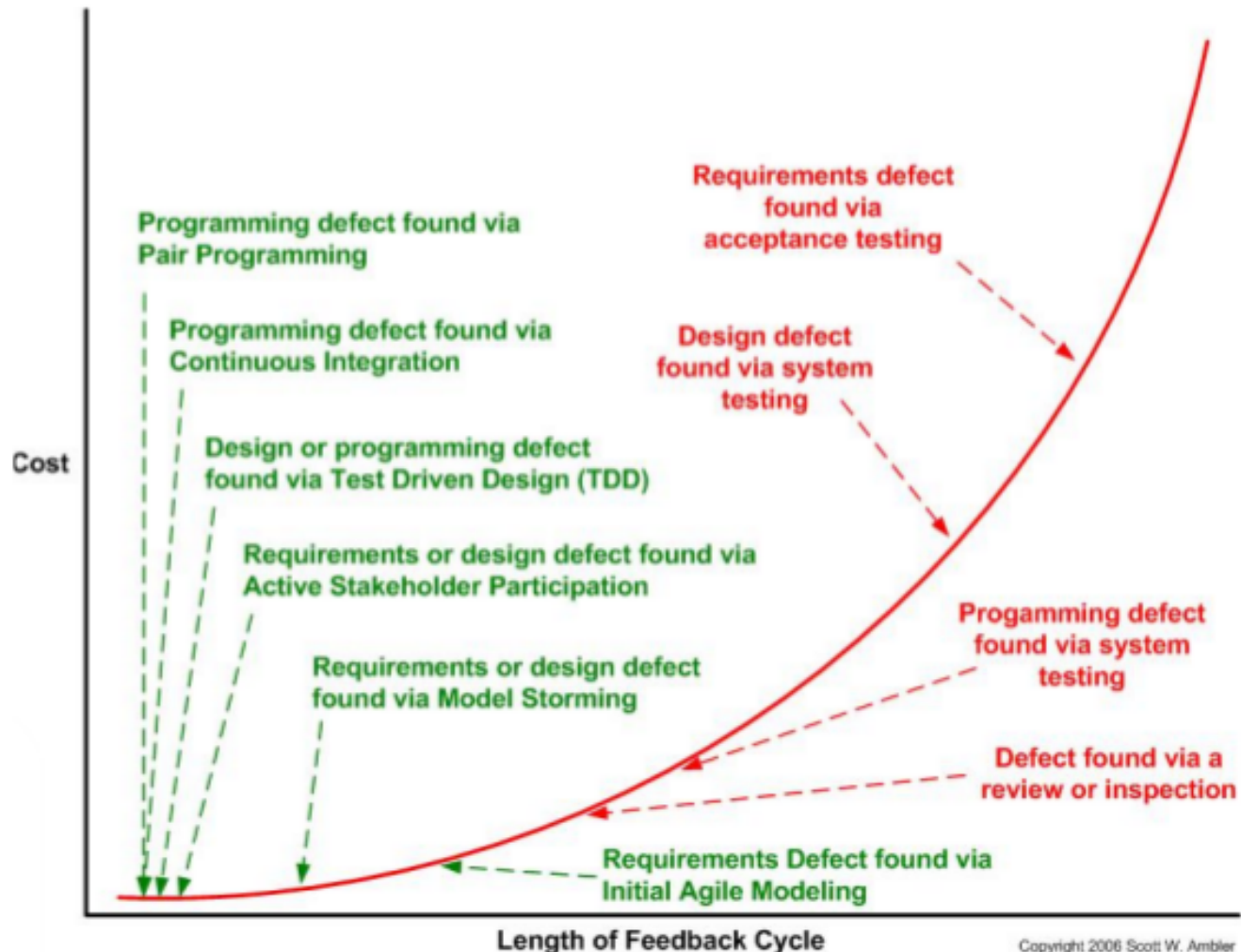


# Travailler en équipe - Tester

- Stratégie de test
- Niveau de test



# Travailler en équipe - Tester



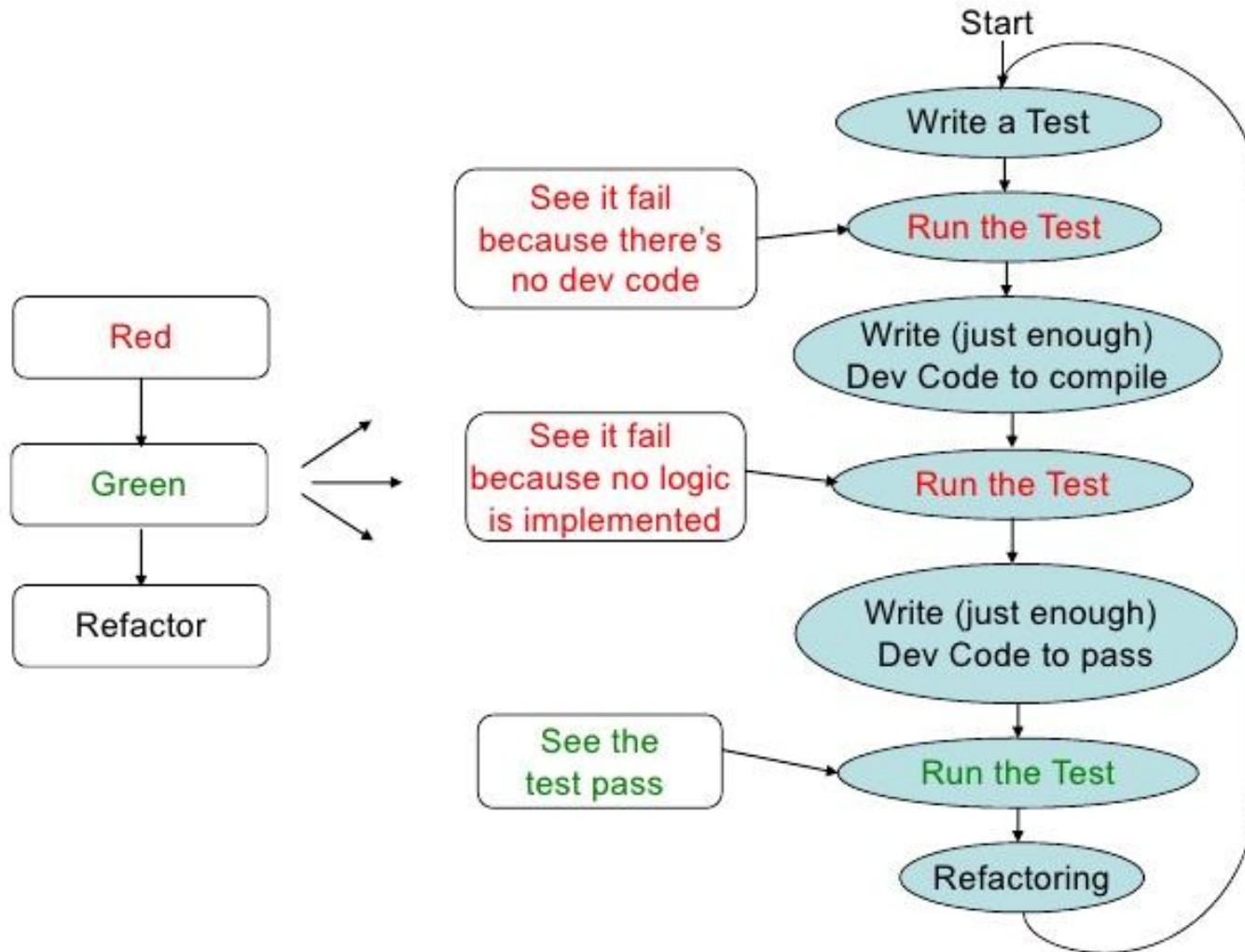
# Travailler en équipe - TDD

## Test Driven Development

- Approche du dev par les tests
  - On code les tests puis on fait le dev
  - Comment utiliser un composant vs comment l'implémenter
  - Ne pas écrire une ligne de code sans un test ko
  - Éliminer les duplications



# Travailler en équipe - TDD



# Travailler en équipe - TDD

- Développeur
  - Tests unitaires
  - Implémentations des BDD
- Client (MOA / Product owner)
  - Tests BDD d'acceptation

```
@Test
public void newArrayListsHaveNoElements() {
    assertThat(new ArrayList().size(), is(0));
}

@Test
public void sizeReturnsNumberOfElements() {
    List instance = new ArrayList();
    instance.add(new Object());
    instance.add(new Object());
    assertThat(instance.size(), is(2));
}
```



# Travailler en équipe - BDD

- Business Driven Développement
- Tests décrits avec une syntaxe et un vocabulaire fixe
- Traduit par le développeur

Feature: Book search

To allow a customer to **find** his favourite books quickly, the library must offer multiple ways to search **for** a book.

Scenario: Search books by publication year

Given a book with the title 'One good book', written by 'Anonymous', published **in** 14 March 2013

And another book with the title 'Some other book', written by 'Tim Tomson', published **in** 23 August 2014

And another book with the title 'How to cook a dino', written by 'Fred Flintstone', published **in** 01 January 2012

When the customer searches **for** books published between 2013 and 2014

Then 2 books should have been found

And Book 1 should have the title 'Some other book'

And Book 2 should have the title 'One good book'

# Travailler en équipe - BDD

## ● Junit + Cucumber

```
public class BookSearchSteps {
    Library library = new Library();
    List<Book> result = new ArrayList<>();

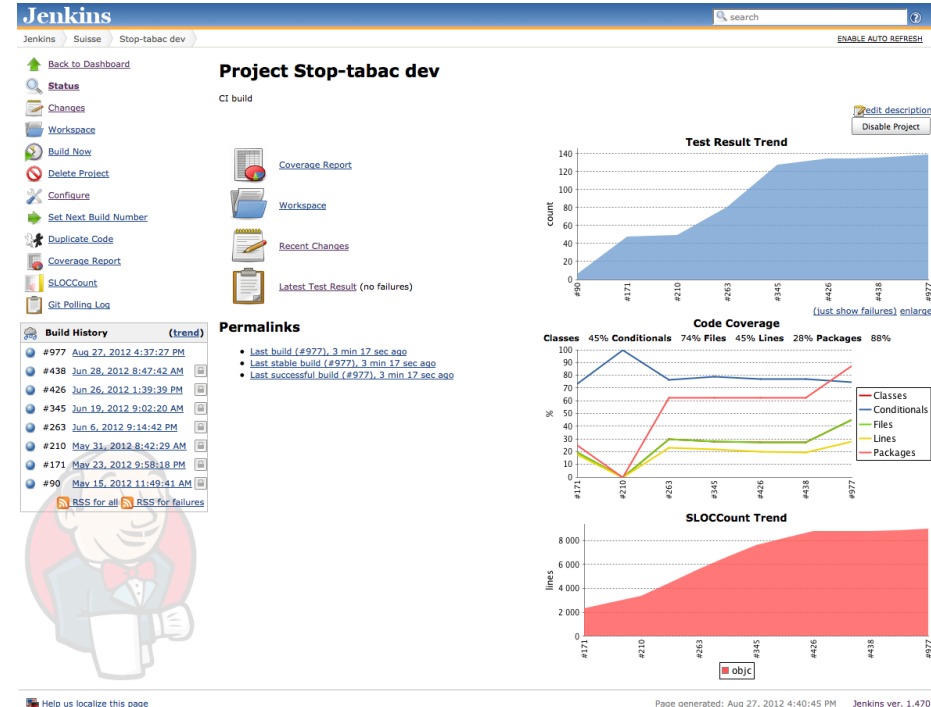
    @Given("+.book with the title '(.+)', written by '(.+)', published in (.+)"
    public void addNewBook(final String title, final String author, @Format("dd MMMMM yyyy") final Date
published) {
        Book book = new Book(title, author, published);
        library.addBook(book);
    }
    @When("^the customer searches for books published between \\d+ and (\\d+)$")
    public void setSearchParameters(@Format("yyyy") final Date from, @Format("yyyy") final Date to) {
        result = library.findBooks(from, to);
    }
    @Then("(\\d+) books should have been found$")
    public void verifyAmountOfBooksFound(final int booksFound) {
        assertEquals(result.size(), booksFound);
    }
    @Then("Book (\\d+) should have the title '(.+)'"
    public void verifyBookAtPosition(final int position, final String title) {
        assertEquals(result.get(position - 1).getTitle(), title);
    }
}
```

# Travailler en équipe - Intégration continue

- Déléguer les tâches répétitives

- Compilation
- Assemblage
- Tests
- Livraison

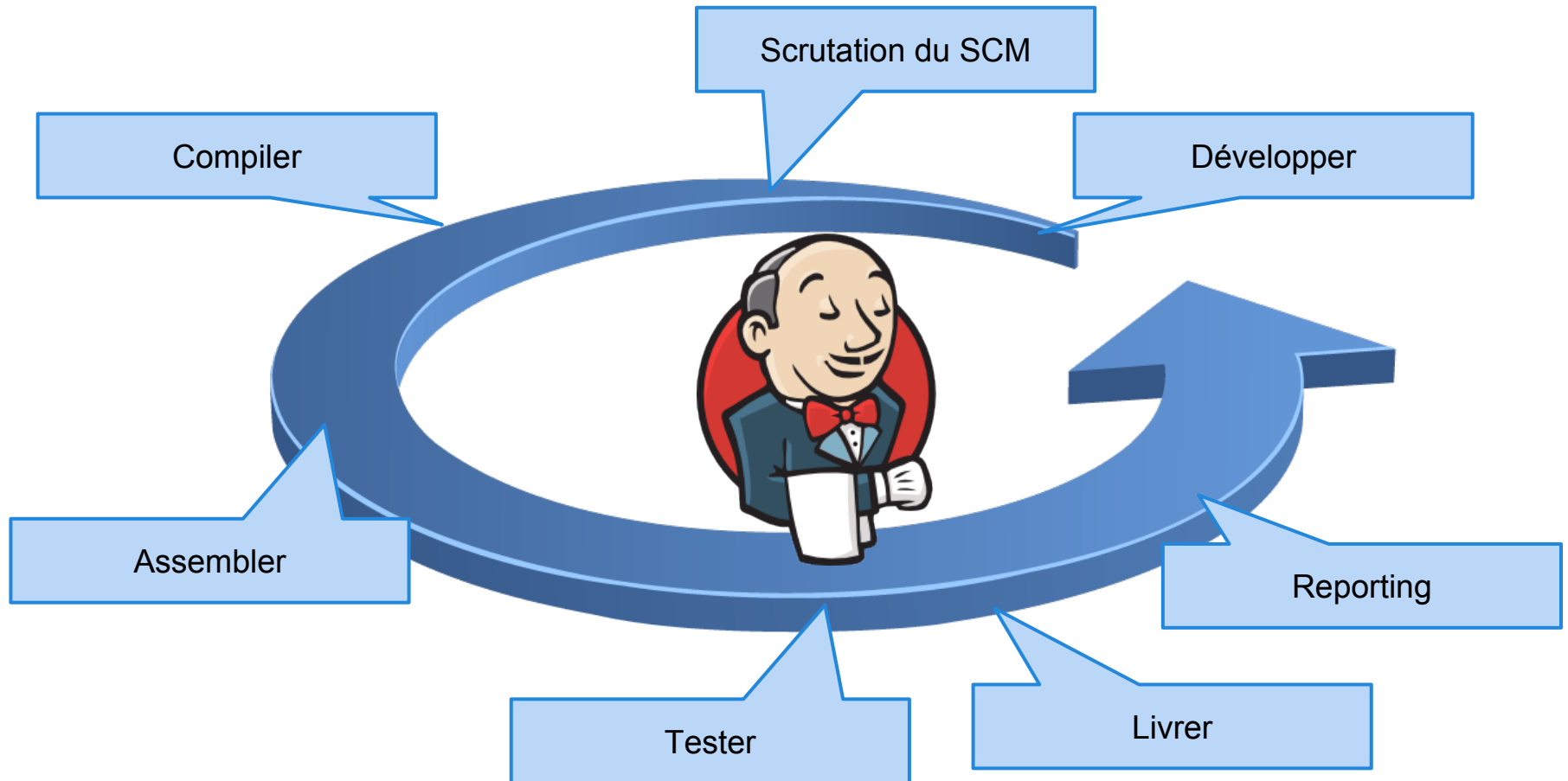
- Reporting
- Partager l'information
- Factuel



# Travailler en équipe - Intégration continue

- Automatiser la construction et le lancement des tests
  - De manière périodique
  - Sur la base de la version courante sur le SCM
- Avertir l'équipe en cas de problèmes
  - Mail aux personnes concernées
- Présenter les résultats à l'équipe
  - Qualimétrie, résultats des tests, rapports de compilation
- Suivre l'impact des développements sur la qualité du build
  - Tendance des constructions

# Travailler en équipe - Intégration continue

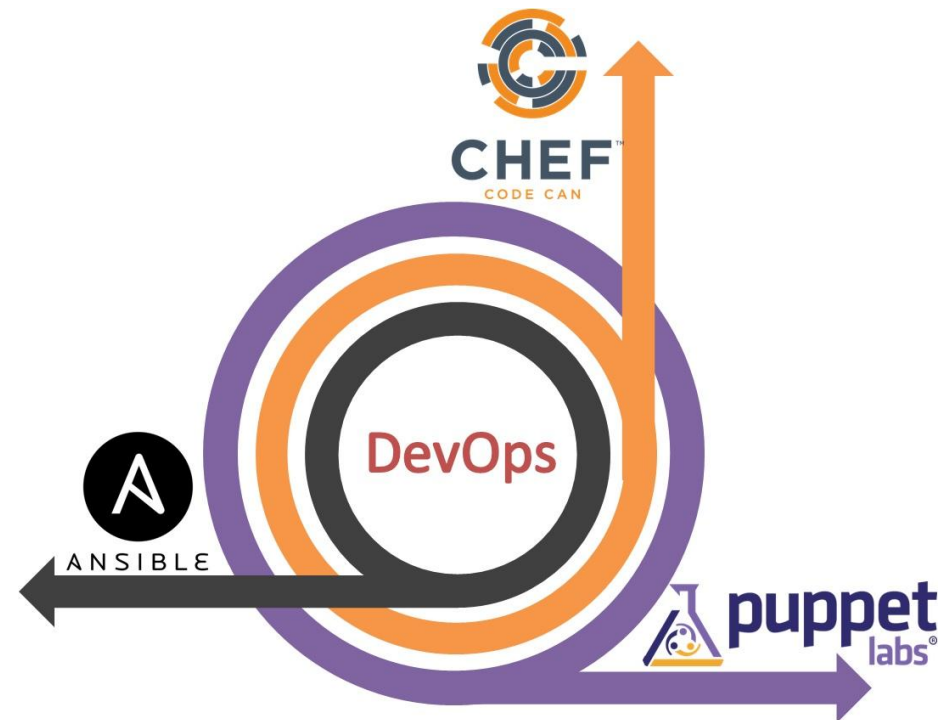


# Travailler en équipe - Intégration continue

- Mise en œuvre des bonnes pratiques de développements
  - Commit régulier : retour rapide sur les erreurs
  - Écriture des tests unitaires : éviter les non-regressions
- Passer d'une démarche personnelle à une démarche d'équipe
- Vérifier en continu l'état des développements
- Libérer des tâches répétitives
- Avoir un référentiel de construction lors des développements

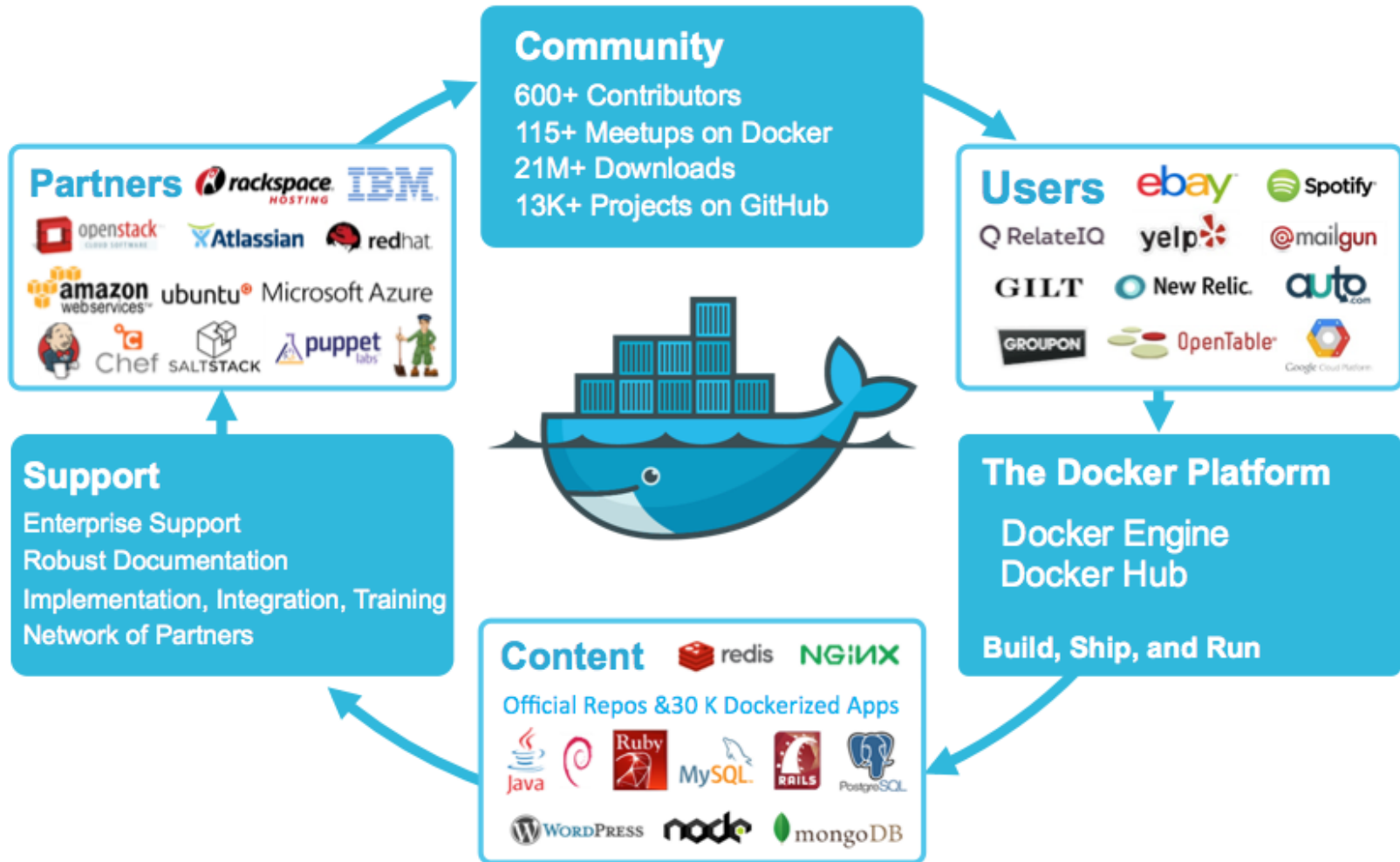
# Livraisons

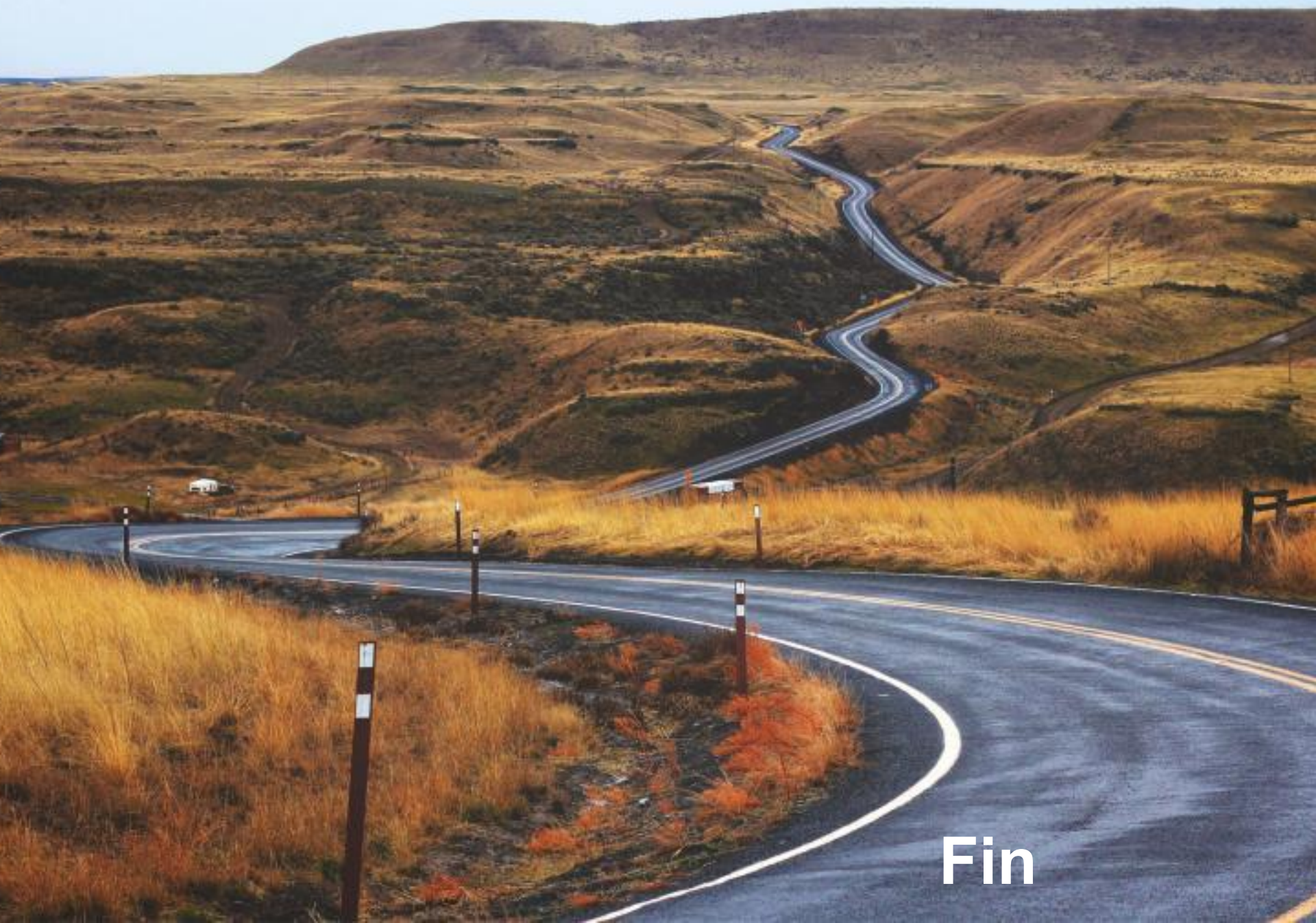
- A la main
- Dev/Ops
- Automatisée
  - OpenStack
  - OpenShift
  - Ansible
  - Puppet/Chef
  - ...
- Docker





# Livraisons - Docker





Fin