EKOD

L'ÉCOLE DES MÉTIERS DU DIGITAL

GO FOR IT !

Une école

CCI LE MANS SARTHE

Lenny Louis

- IUT de Laval promotion 2022
- ESIEA promotion 2025
- 3 ans chez Orange à Rennes
- Ingénieur logiciel indépendant
- Formateur indépendant

Structure du cours

- CM : **2 heures**
- TD : **12 heures**

Que signifie qualité pour vous ?

Introduction à la qualité logicielle

- **Qualité logicielle** : La capacité d'un logiciel à satisfaire les attentes des utilisateurs et à respecter les exigences spécifiées

- **Objectifs du cours** :
  - Comprendre l'importance de la qualité logicielle.
  - Découvrir les différents types de tests et techniques de test
  - Apprendre les étapes de mises en œuvre d'un test

Pourquoi faut-il tester les logiciels ?

Objectif des tests

- Détecter les erreurs avant que les utilisateurs ne les rencontrent.
- Améliorer la confiance dans le logiciel.
- Réduire les risques de défaillance en production.
- Fournir des informations aux décideur pour une meilleure visibilité.

Exemples d’échecs logiciels

Orange : Le 2 juin 2021 une mise à jour logicielle rend impossible la communication vers les services d’urgence.

Un bug présent dans le module d’interconnexion entre le réseau IP et les numéros analogiques.

Bilan : 5 morts

CrowdStrike : Le 19 juillet 2024 une mise à jour de configuration de la plateforme Falcon contient une entrée non prévue.

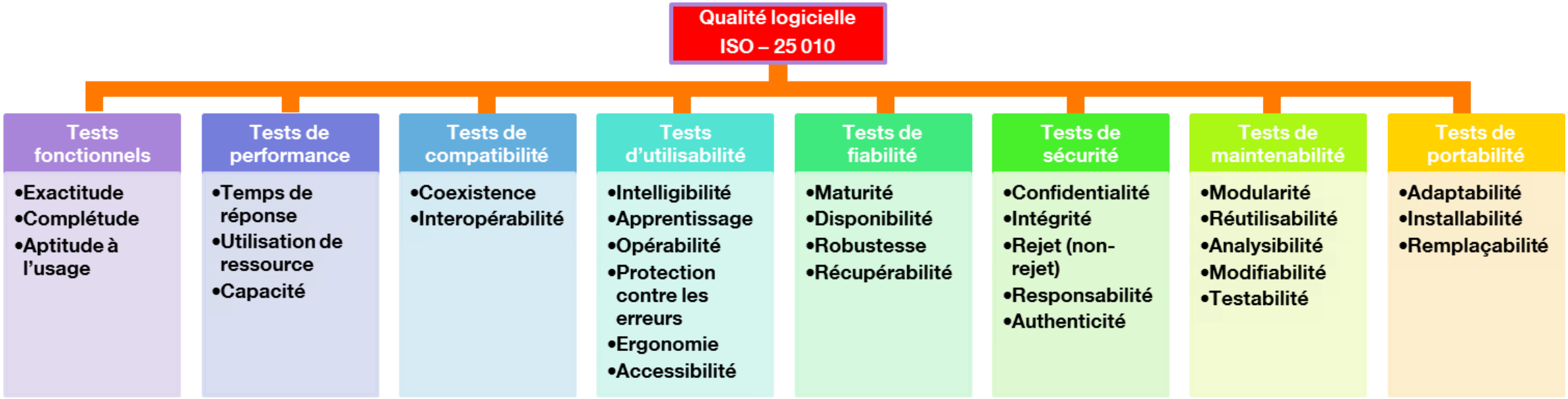
Ce fichier étant utilisé par le système d’exploitation perturbe la communication entre les processus. Environs 8,5 millions ordinateurs ont plantés.

Bilan : Vols, moyens de paiements bloqués, annulation d'opérations chirurgicales…

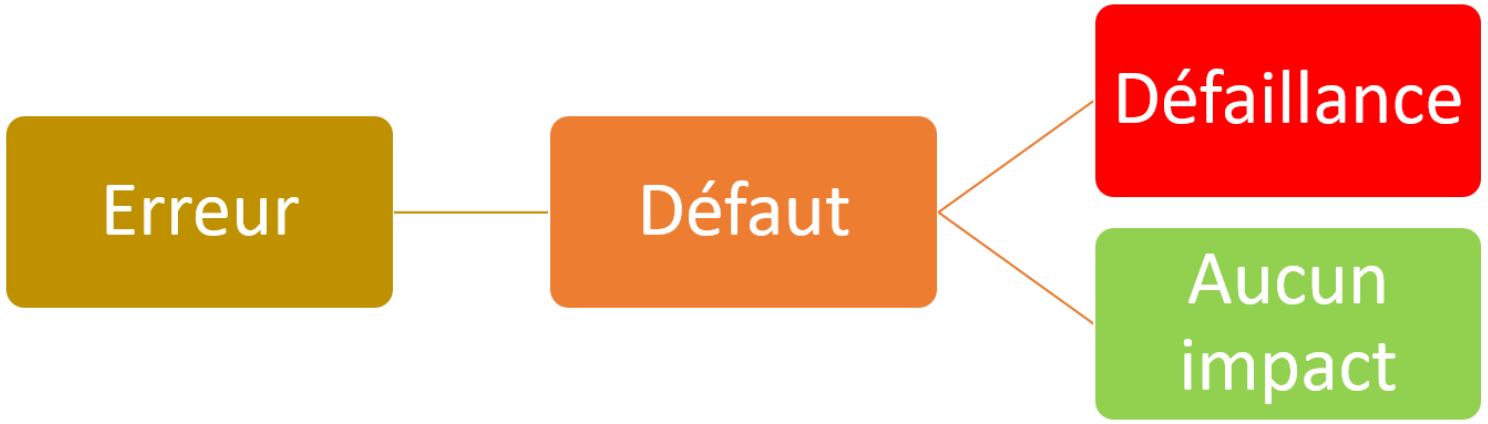
EKOD L’ÉCOLE DES MÉTIERS DU DIGITAL — Une école CCI LE MANS SARTHE

Qu’est-ce qu’un logiciel de qualité ?

- Les 8 critères qualité selon la norme **ISO 25010** :



Erreur, défaut, défaillance



Les 7 principes fondamentaux du test

- Analogique avec une crise sanitaire

1. Les tests montrent la présence de défaut

- **Principe :** Les tests peuvent démontrer que des défauts existent, mais ils ne prouvent jamais qu'un logiciel est sans défaut.

- **Analogie :** Dépister des individus mettra en évidence que le virus est encore présent mais ne garantira pas sa disparition.

2. Les tests exhaustifs sont impossibles

- **Principe :** Tester tous les scénarios est irréaliste car il existe une infinité de combinaisons possibles dans les logiciels.

- **Analogie :** Impossible de dépister tous les individus simultanément dans le monde entier.

3. Tester tôt

- **Principe :** Identifier et corriger les défauts tôt dans le cycle de développement réduit les coûts et prévient les risques en production.

- **Analogie :** Un dépistage rapide permet une prise en charge plus précoce, limitant la propagation.

4. Regroupement de défauts

- **Principe :** Les défauts se regroupent souvent dans certains modules ou parties du code, en raison de faiblesses spécifiques.

- **Analogie :** Les infections se concentrent souvent dans certains clusters (ou zones).

5. Le paradoxe du pesticide

- **Principe :** Répéter les mêmes tests conduit à ne plus détecter de nouveaux défauts, car les défauts restants se trouvent ailleurs.

- **Analogie :** Tester toujours les mêmes groupes de personnes risque de laisser passer des cas positifs dans d'autres groupes.

6. Les tests dépendent du contexte

- **Principe :** Les méthodes de test et leur rigueur dépendent du contexte d'utilisation du logiciel.

- **Analogie :** Le dépistage est plus strict pour les personnes vulnérables ou en contact avec des cas positifs.

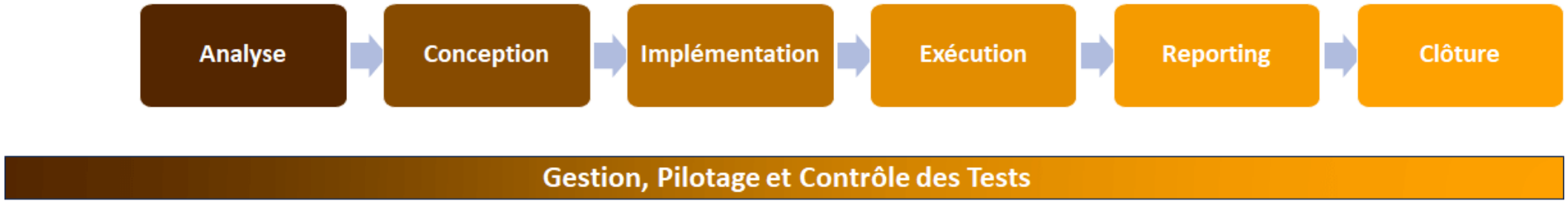
7. L’illusion d’absence d’erreurs

- **Principe :** Un logiciel sans défaut visible ne garantit pas l'absence d'erreurs ; il peut souffrir de problèmes liés aux spécifications ou à des attentes non couvertes.

- **Analogie :** Tester négatif ne signifie pas être immunisé contre le virus ou qu’il n’y a aucun risque.

Le processus de test

- Garantir la qualité logicielle en suivant une suite d'activités structurées :



Activité transverse consistant à suivre l'avancée des tests.
Adapter la planification selon la nécessité.

Analyse des tests

- **Objectif** : Identifier précisément ce qui doit être testé.
  - Définir les **conditions de test.**
  - Analyser les exigences et spécifications.
  - *Exemple : Dans une messagerie, déterminer les types de fichiers autorisés pour les pièces jointes, la taille maximale, etc.*

Livrables

Des conditions de test
Des critères d'acception clairs

Conception des tests

- **Objectif** : Définir comment tester chaque condition de test.
  - Rédaction des **cas de test.**
  - *Exemple : Décider des formats de fichiers à tester et de la manière de les ajouter.*

Livrables

Des cas de test

Implémentation des tests

- **Objectif** : Préparer l'environnement pour l'exécution des tests.
  - Préparer les **environnements et les données de test.**
  - Initialiser les campagnes de test, avec les tests marqués comme non exécutés.
  - Vérifier que tous les prérequis pour l'exécution sont en place.

Livrables

Données de test

Exécution des tests

- **Objectif** : Réaliser les tests et documenter les résultats
  - Exécuter chaque test de la campagne.
  - Analyser les cas d'échec et documenter les **défauts et défaillances détectés**.

Livrables

Rapports d'exécution
Rapports des anomalies

Évaluation des critères de sortie et Reporting

- **Objectif** : Communiquer les résultats et évaluer les critères de sortie.
  - Mesurer l'avancement des tests par rapport aux critères de sortie.
  - Rédiger un **bilan de tests** pour résumer les résultats.

Livrables

Bilan de tests

Clôture des tests

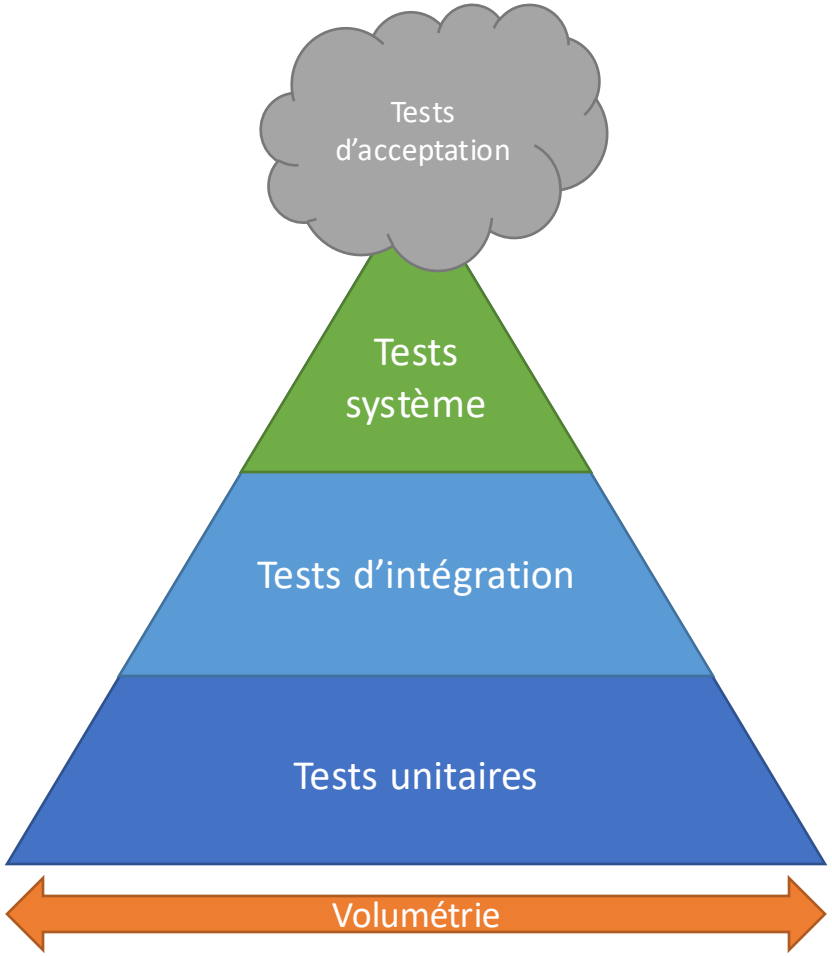
- **Objectif** : Finaliser et améliorer le processus de test.
  - Faire une rétrospective pour identifier ce qui a fonctionné et ce qui peut être amélioré.
  - Documenter les **actions d'amélioration continue.**

Livrables

Plan d'amélioration continue

Les niveaux de test

Unitaire	Intégration	Système	Acceptation
Proche du code	Assemblages de composants	Proche de la production	Validation du produit
Découpage : une fonction / classe	Découpage : classes, base de données etc.	Découpage : le produit dans son ensemble	Intégral



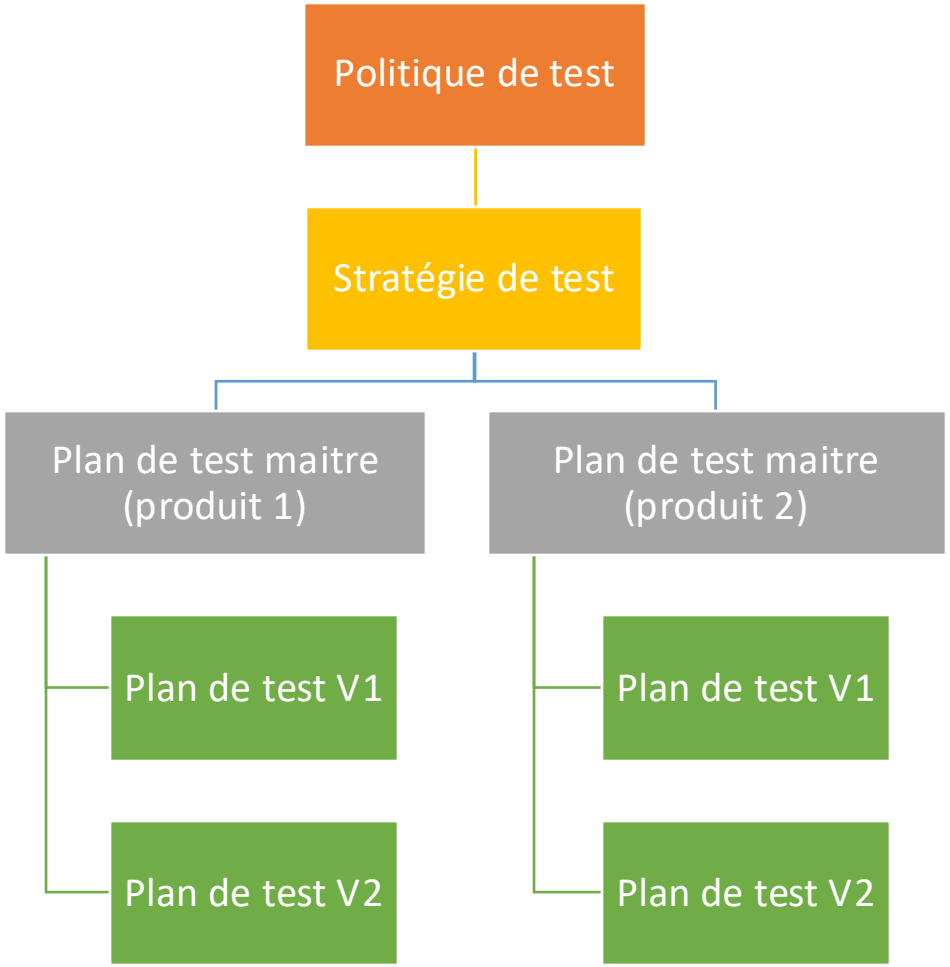
Le niveau unitaire

Le niveau intégration

Le niveau système

Le niveau acceptation

Les documents dans le métier du test



Niveau entreprise :
Pourquoi testons-nous ?
Où allons-nous ?

Niveau entreprise :
Méthodologie de test dans l'entreprise
Quel trajet prenons-nous ?

Niveau équipe :
Comment tester le produit ?
Comment faisons-nous ?

Niveau équipe :
Comment tester la V1 du produit ?

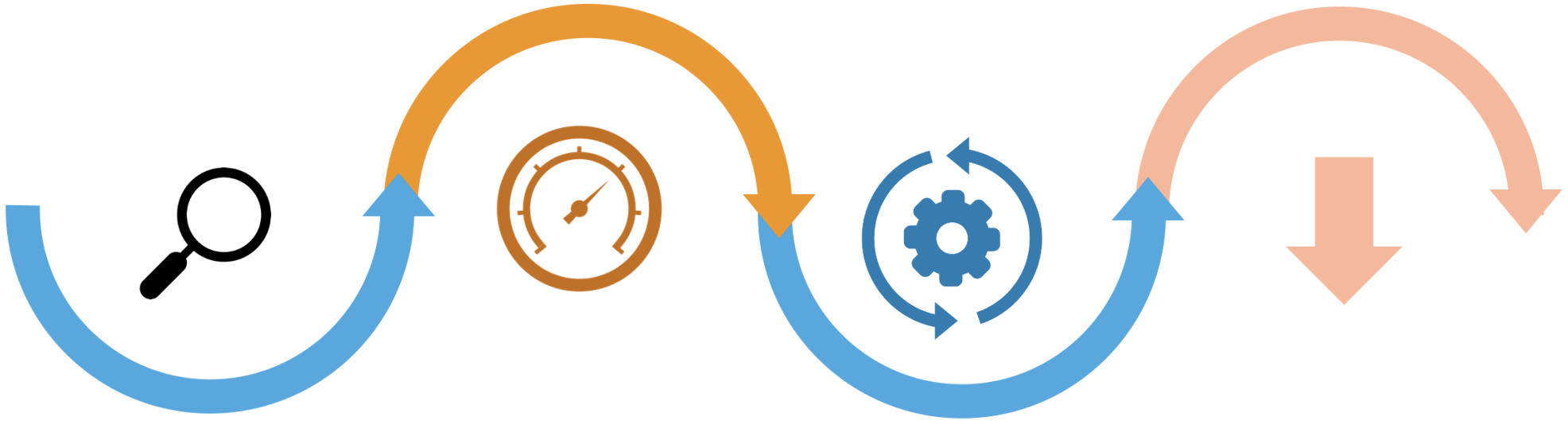
Niveau équipe :
Comment tester la V2 du produit ?

La gestion du risque

- Détermination du niveau de risque selon l'IEC 61508 :

	Anecdotique	*Peu impactant*	*Impactant*	*Catastrophique*
Invraisemblable	*I*	*I*	*II*	*III*
Peu probable	*I*	*II*	*III*	*III*
Probable	*II*	*II*	*III*	*IV*
Certain	*II*	*III*	*IV*	*IV*

La gestion du risque



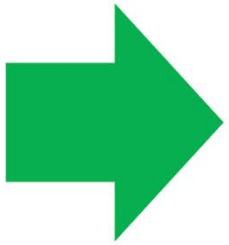
Identifier les risques

Evaluer les risques

Gérer les risques

Réduire les risques

La gestion du risque

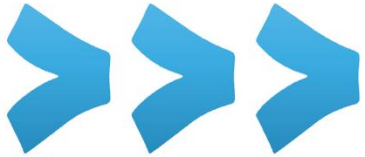


Ne rien faire

Prendre des précautions

Prévenir

Protéger



Transférer

Abandonner

Le plan de test (maître)

- Tout comme on dessine les plans d'un édifice avant de le réaliser, on élabore un plan avant de tester un logiciel.
  - Déterminer le périmètre des tests : Ce que l'on teste (et qu'on ne teste pas).
  - Approches de test : Comment les tests sont abordés.
  - Critères de sortie : Ce que l'on doit atteindre pour arrêter de tester.
  - Livrables de tests (artefacts) : Documents, fichiers, etc. qui seront produits par les tests.
  - Tâches à réaliser : Liste d'actions à faire lors des tests.
  - Besoins en ressources : Humaines et matérielles (compétences, devices, environnements, etc.).
  - Acteurs et responsabilités : Sous forme de RACI par exemple.
  - Planification : Planning temporel des actions.
  - Risques et contingences : Risques produits et risques projets.

Les approches de test

Analytique

Réactive

Basée sur des modèles

Méthodique

Conforme à un processus

Dirigée (ou Consultative)

Anti-régression

Risque produit ou risque projet ?

Retard de livraison d'un prestataire développant une dépendance

Manque de soutien de la hiérarchie

Les développeurs de l'équipe sont tous juniors

L'architecte de la solution est absent pendant 3 mois

Le prestataire développant une dépendance n'a fait aucun test

Absence de besoin clair remonté par les utilisateurs finaux

L'équipe est répartie dans des pays différents

Plannings et budgets irréalistes

Risque produit ou risque projet ?

Retard de livraison d'un prestataire développant une dépendance

Manque de soutien de la hiérarchie

Les développeurs de l'équipe sont tous juniors

L'architecte de la solution est absent pendant 3 mois

Absence de besoin clair remonté par les utilisateurs finaux

Le prestataire développant une dépendance n'a fait aucun test

L'équipe est répartie dans des pays différents

Plannings et budgets irréalistes

Risque produit ou risque projet ?

Retard de livraison d'un prestataire développant une dépendance

Manque de soutien de la hiérarchie

Les développeurs de l'équipe sont tous juniors

L'architecte de la solution est absent pendant 3 mois

Absence de besoin clair remonté par les utilisateurs finaux

Le prestataire développant une dépendance n'a fait aucun test

L'équipe est répartie dans des pays différents

Plannings et budgets irréalistes

› Exemple : Calculer des frais kolométriques

https://www.impots.gouv.fr/simulateur-bareme-kilometrique

Moyen de transport

Puissance du véhicule

Distance parcourue

Exemple de test boîte blanche

La composition du test sera influencée par la structure du code

Exemple de test boîte blanche

```
public double getBaremeVoiture(int nbCv, int nbKm) {
   double ret  0;
   switch(nbCv) {
        case 0 :
        case 1:
        case 2:
        case 3 :
            if (nbKm <= 5000) ret = 0.529;
            else if (nbKm >5000 && nbKm <=20000) ret = 0.316;
            else ret = 0.370;
            break;
        case 4 :
        ...
   }
    return ret;
}
```

La composition du test sera influencée par la structure du code

Exemple de test boîte blanche

```
public double getBaremeVoiture(int nbCv, int nbKm) {
   double ret  0;
   switch(nbCv) {
        case 0 :
        case 1:
        case 2:
        case 3 :
            if (nbKm <= 5000) ret = 0.529;
            else if (nbKm >5000 && nbKm <=20000) ret = 0.316;
            else ret = 0.370;
            break;
        case 4 :
        ...
   }
    return ret;
}
```

```
@Test
public void monTestBaremeVoiture0Cv() {
   assertEquals(0,529, getBaremeVoiture(0, 4999);
   assertEquals(0,370, getBaremeVoiture(0, 210000);
}
```

La composition du test sera influencée par la structure du code

Exemple de test boîte blanche

```
public double getBaremeVoiture(int nbCv, int nbKm) {
   double ret  0;
   switch(nbCv) {
        case 0 :
        case 1:
        case 2:
        case 3 :
            if (nbKm <= 5000) ret = 0.529;
            else if (nbKm >5000 && nbKm <=20000) ret = 0.316;
            else ret = 0.370;
            break;
        case 4 :
        ...
   }
    return ret;
}
```

La composition du test sera influencée par la structure du code

```
@Test
public void monTestBaremeVoiture0Cv() {
   assertEquals(0,529, getBaremeVoiture(0, 4999);
   assertEquals(0,370, getBaremeVoiture(0, 210000);
}
```

```
@Test
public void monTestBaremeVoiture5Cv() {
   assertEquals(0,636, getBaremeVoiture(5, 4999);
   assertEquals(0,427, getBaremeVoiture(5, 21000);
}
```

Exemple de test boîte blanche

```
public double getBaremeVoiture(int nbCv, int nbKm) {
   double ret  0;
   switch(nbCv) {
        case 0 :
        case 1:
        case 2:
        case 3 :
            if (nbKm <= 5000) ret = 0.529;
            else if (nbKm >5000 && nbKm <=20000) ret = 0.316;
            else ret = 0.370;
            break;
        case 4 :
        ...
   }
    return ret;
}
```

La composition du test sera influencée par la structure du code

```
@Test
public void monTestBaremeVoiture0Cv() {
   assertEquals(0,529, getBaremeVoiture(0, 4999);
   assertEquals(0,370, getBaremeVoiture(0, 210000);
}
```

```
@Test
public void monTestBaremeVoiture5Cv() {
   assertEquals(0,636, getBaremeVoiture(5, 4999);
   assertEquals(0,427, getBaremeVoiture(5, 21000);
}
```

```
@Test
public void monTestBaremeVoiture7Cv() {
   assertEquals(0,697, getBaremeVoiture(7, 4999);
   assertEquals(0,470, getBaremeVoiture(7, 21000);
}
```

Exemple de test boîte noire

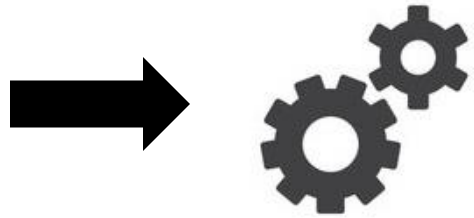
La composition du test sera influencée par le résultat attendu

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Exemple de test boîte noire

La composition du test sera influencée par le résultat attendu

Type de véhicule
Puissance fiscale
Nombre kilomètres



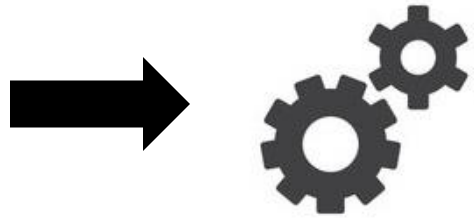
Tarif

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Exemple de test boîte noire

La composition du test sera influencée par le résultat attendu

Type de véhicule
Puissance fiscale
Nombre kilomètres



Tarif

Voiture
5 CV
6800 km

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Exemple de test boîte noire

La composition du test sera influencée par le résultat attendu

Type de véhicule
Puissance fiscale
Nombre kilomètres

Tarif

Voiture
5 CV
6800 km

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Exemple de test boîte noire

La composition du test sera influencée par le résultat attendu

Type de véhicule
Puissance fiscale
Nombre kilomètres

Tarif

Voiture
5 CV
6800 km

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Exemple de test boîte noire

La composition du test sera influencée par le résultat attendu

Type de véhicule
Puissance fiscale
Nombre kilomètres

Tarif

Voiture
5 CV
6800 km

3294 €

6800 x 0,308 = 3294 €

Puissance administrative (en CV)	Distance (d) jusqu'à 5 000 km	Distance (d) de 5 001 km à 20 000 km	Distance (d) au-delà de 20 000 km
3 CV et moins	d x 0,529	(d x 0,316) + 1 065	d x 0,370
4 CV	d x 0,606	(d x 0,340) + 1 330	d x 0,407
5 CV	d x 0,636	(d x 0,357) + 1 395	d x 0,427
6 CV	d x 0,665	(d x 0,374) + 1 457	d x 0,447
7 CV et plus	d x 0,697	(d x 0,394) + 1 515	d x 0,470

Les référentiels de test

- Stockage et gestion des tests manuels
- Permet de suivre et d'effectuer une revue des tests.
- Capitalise dans le temps les tests et leurs exécutions.
- Permet gérer les exécutions des tests.
- Propose le reporting et la génération de tableaux de bord (pilotage).

XRAY

REFER TEST
MANAGE YOUR TEST

ALM Octane

TestLink

squash TM

Les bug tracker

- Stockage et gestion des anomalies.
- Permet de suivre et d'effectuer une revue des anomalies.
- Donne une visibilité des anomalies à l'équipe et à l'extérieur.
- Permet de lier les défauts et les tests.

Jira

mantis BUG TRACKER

Les tests unitaires

- Propose des outils pour tester le code :
  - Assertions
  - Annotations
  - Mécanismes (avant / après) test et ensemble de tests
  - Reporting


Les tests d'API

- Appelle des API ou WebService.
- Fourni des données en paramètre.
- Récupère des données en réponse.
- Compare les données en réponse avec un attendu.

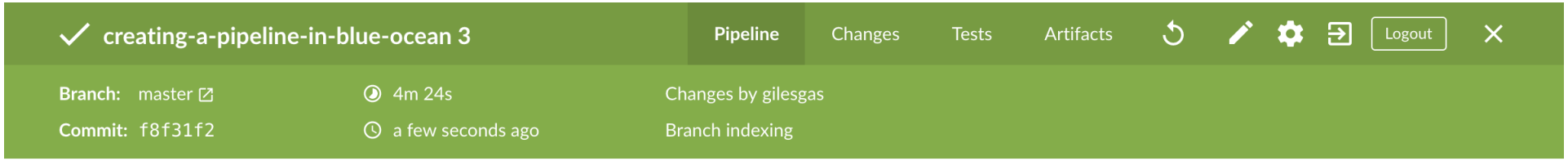
L’automatisation des tests IHM

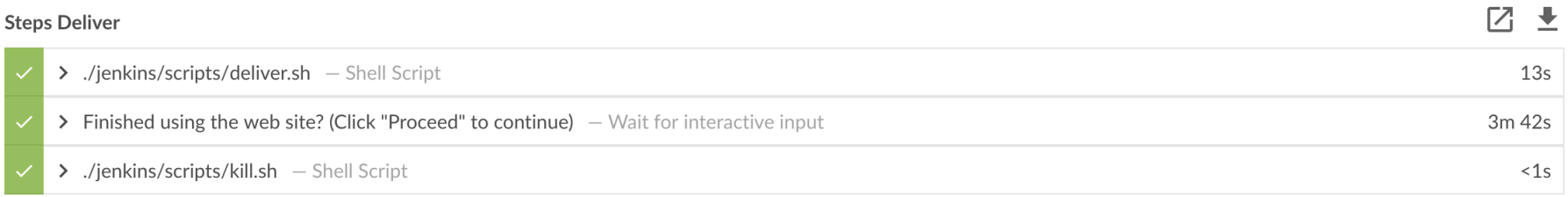
- IHM : Interface Homme Machine.
- Simule l’utilisation de l’application comme un humain.
- La majorité des IHM passent par le web d’où l’existence d’une multitude d’outils.
- Tester par l’IHM est couteux car il est nécessaire
  d’intégrer tous les composants (= niveau système).

Les tests de performance

- Simule l'utilisation simultanée de plusieurs utilisateurs.
- Sollicite le système à une forte charge.
- Mesure les temps de réponse.
- Surveille l'utilisation des ressources.

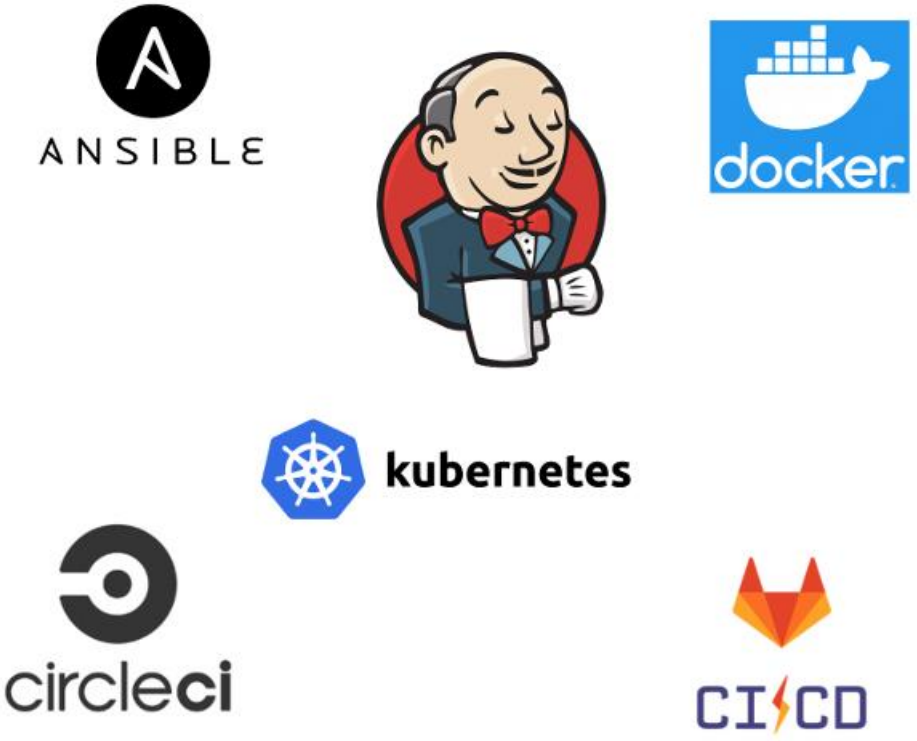
Exemple d’un pipeline





CI/CD

- Au-delà du code développé et des tests, le logiciel vit dans le temps.
- L'intégration continue aide à gérer le cycle de vie du logiciel, et automatiser des tâches.
- Exécution automatique des tests mais aussi son installation dans les environnements.



Pour en apprendre d'avantage

Le métier du test

La taverne du testeur

Comité Français des Tests Logiciels


QUESTIONS