

Automatiser les tests

L'enjeu

Comprendre la place de l'automatisation dans le contrôle de la qualité

- Pourquoi cherche-t-on à automatiser les tests ?
- A quel niveau* peut-on automatiser les tests ?
→ Quand ?
- Est-ce que tout type* de test est automatisable ?
- Quels sont les outils de l'automatisation ?
→ Comment ?

* Les termes de **niveau** et **type** de test doivent être compris au sens de la définition de l'ISTQB

I. Pourquoi automatiser les tests?

1. Niveaux et types de tests
2. Les avantages de l'automatisation
3. Risques et inconvénients de l'automatisation

II. L'automatisation selon le types de tests

1. Tests fonctionnels
2. Tests de sécurité applicative
3. Tests de performance
4. Qualimétrie
5. Tests d'utilisabilité



Les types de tests

I. Pourquoi automatiser les tests?

1. Niveaux et types de tests

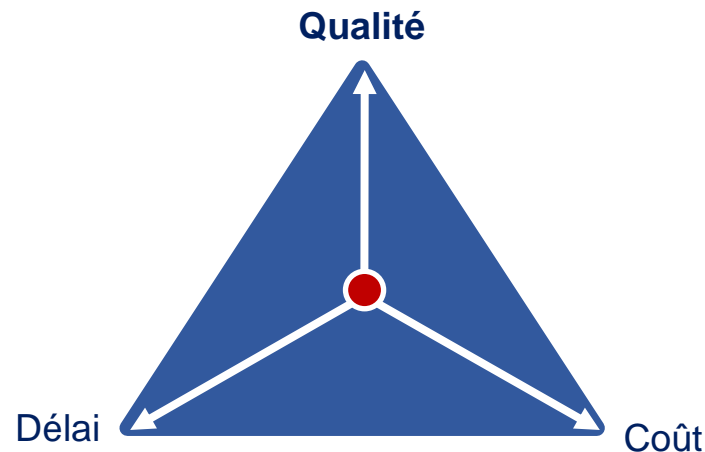
2. Les avantages de l'automatisation
3. Risques et inconvénients de l'automatisation

II. L'automatisation selon le types de tests



Qu'est-ce que l'activité de test

- Les tests permettent de **s'assurer du niveau de qualité** du produit livré.
- Ils apportent une assurance contre les **risques** et les **coûts cachés** résultant de la non-qualité.
 - Exemples de risques : les bugs, les pannes ...
 - Exemple de coût cachés : perte d'exploitation, inactivité des équipes, pénalités de retard de livraison
- Le test est une véritable activité, qui implique du **temps**, des **ressources** et donc de **l'argent**. Il s'agit d'un **investissement**



le triangle **Qualité / Coût / Délais**

Les types de tests : facteurs et critères de la qualité

Le standard ISO/IEC 25010 définit la qualité logicielle selon huit critères

ISO/IEC 25010

Capacité Fonctionnelle

- Complétude fonctionnelle
- Exactitude fonctionnelle
- Adéquation fonctionnelle

Performance / Rendement

- Temps de réponse
- Utilisation des ressources
- Capacité maximale

Compatibilité

- Coexistence
- Interopérabilité

Utilisabilité

- Facilité de prise en main
- Facilité d'apprentissage
- Opérabilité
- Protection contre les erreurs
- Esthétique
- Accessibilité

Fiabilité

- Maturité
- Disponibilité
- Tolérance aux pannes
- Facilité de recouvrement

Sécurité

- Confidentialité
- Intégrité
- Non-répudiation
- Traçabilité
- Authenticité

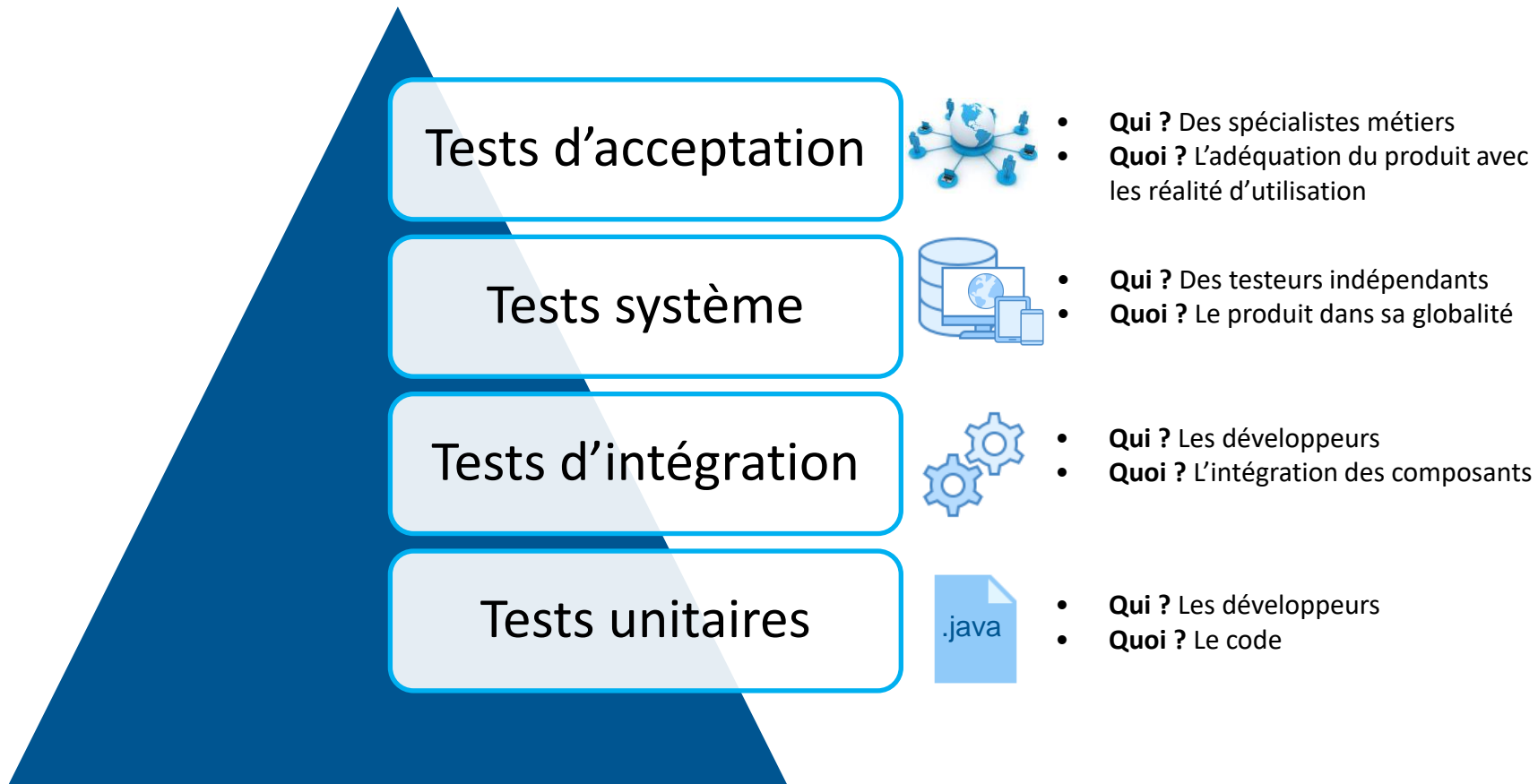
Maintenabilité

- Modularité
- Réutilisabilité
- Analysabilité
- Capacité d'évolution
- Testabilité

Portabilité


- Facilité d'adaptation
- Facilité d'installation
- Facilité de substitution

Les niveaux de test



Les niveaux de tests sont souvent représentés sous la forme d'une pyramide, signifiant que l'effort de test devrait être porté intensivement dès le niveau unitaire, au fondement de la pyramide (vision Agile)

Les types de tests par niveaux



NIVEAUX	QUI ?	QUOI ?
ACCEPTATION	UTILISATEURS EXPLOITATION	Comportement ... Performance / Fiabilité / Sécurité...
SYSTÈME	TEST	Fonctionnalités... Livrables / Installations / paramétrage...
INTEGRATION	DEV	Capacité d'échange information et données Assemblage composants
UNITAIRE	DEV	Exécution des fonctions et méthodes Qualité du code

* Tests fonctionnels

* Tests techniques

Les avantages de l'automatisation

I. Pourquoi automatiser les tests?

1. Niveaux et types de tests
2. **Les avantages de l'automatisation**
3. Risques et inconvénients de l'automatisation

II. L'automatisation selon le types de tests



Objectifs de l'automatisation du contrôle de la qualité

La mise en place d'un chantier d'automatisation des tests n'est pas anodin. C'est une stratégie sur le moyen / long terme qui vise certains objectifs :

- Améliorer la productivité des tests
- Etendre la couverture des tests
- Augmenter la profondeur des tests
- Réduire le coût global du test
- Réduire le temps d'exécution des tests (donc du délais de mise en production)

Basiquement, il s'agit de lutter contre le principe du triangle
Qualité / Coût / Délais



Augmenter la qualité

Réduire les coûts

Réduire les délais

Avantages de l'automatisation du contrôle de la qualité

AMÉLIORER LA QUALITÉ DE L'APPLICATION

➤ Augmentation de la **couverture de test**

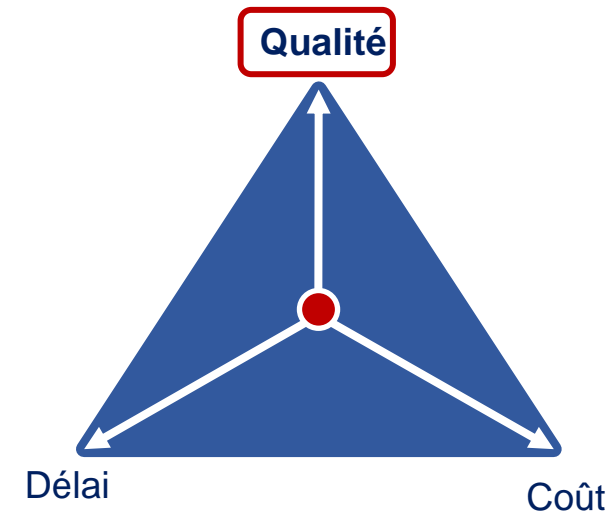
Une partie des tests étant automatisée, l'équipe peut mettre à profit le temps économisé pour exécuter d'autres tests, qui n'auraient sans doute pas été exécutés sans cela.

➤ **Exécution systématique et fréquente** des tests automatisés

Les tests automatisés peuvent être rejoués à loisir, et dans l'urgence (ex. : avant la mise en production d'un patch urgent).

➤ Augmentation de la « **profondeur** » des tests (= nombre et précision des vérifications effectuées)

Les automates permettent d'effectuer des tests pénibles, ou source d'erreurs lorsqu'ils sont effectués par un humain (ex. : vérification de fichiers à champs fixes, vérifications en base de données, etc.).



Avantages de l'automatisation du contrôle de la qualité

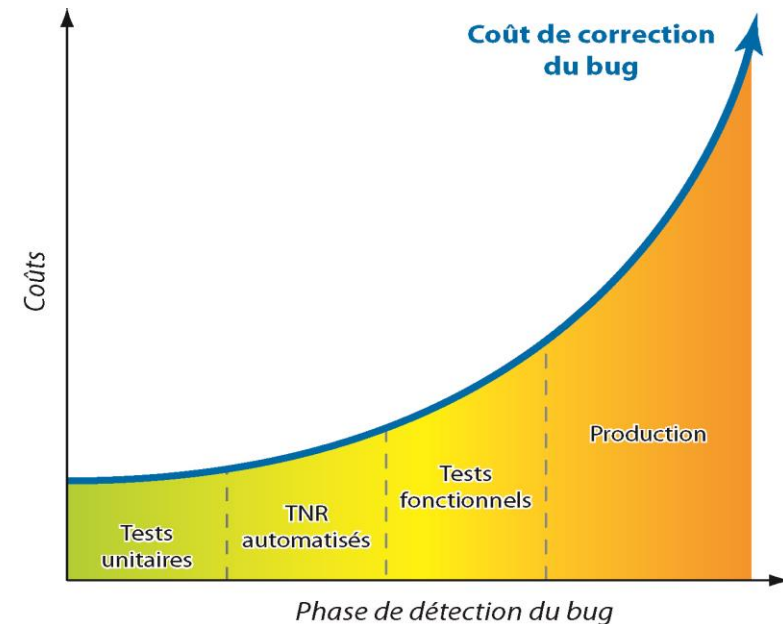
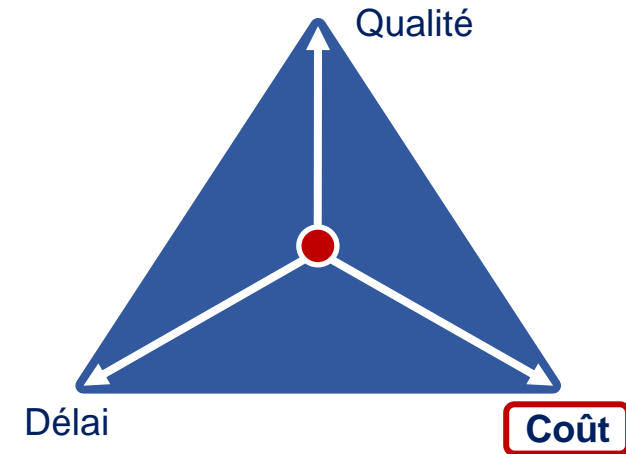
RÉDUIRE LE COÛT

➤ D'exécution des tests

Le temps machine nécessaire à l'exécution des tests automatisés ne coûte « rien » (mais il faut quand même intégrer une petite charge d'analyse).

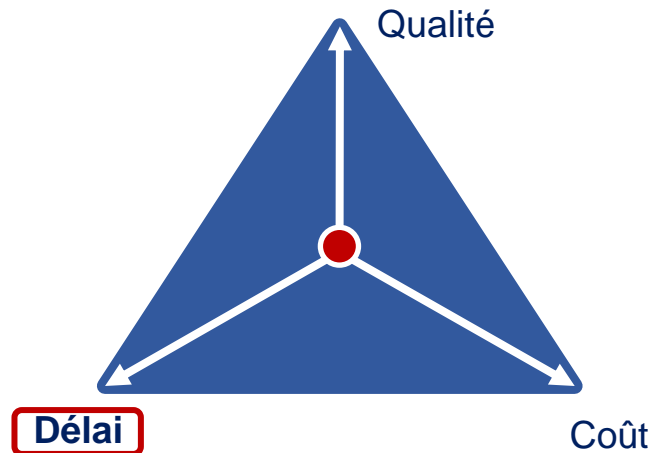
➤ De correction des anomalies

En exécutant les tests automatisés avant la livraison en recette (par exemple, pendant la phase d'intégration), on peut trouver des anomalies plus tôt, et donc, moins coûteuses à corriger.



Avantages de l'automatisation du contrôle de la qualité

RÉDUIRE LES DÉLAIS DE MISE EN PRODUCTION



- L'automatisation permet de diminuer la durée d'exécution des tests et donc :
 - De diminuer la durée du chemin critique du projet.
 - De diminuer les délais de mise en production.

Risques et inconvénients de l'automatisation

I. Pourquoi automatiser les tests?

1. Niveaux et types de tests
2. Les avantages de l'automatisation
3. **Risques et inconvénients de l'automatisation**

II. L'automatisation selon le types de tests



Les risques liés à l'automatisation des tests

Coût additionnels

La mise en place d'un projet d'automatisation coûte cher. Une mauvaise évaluation de l'investissement initial peut être fatal pour un projet d'automatisation de test.

- Besoins en technologies additionnelles (outils commerciaux, technologie propriétaire...)
- L'équipe doit avoir les compétences requises (formation, expertise, montée en compétences...)
- Les tests automatisés sont sujets à des régressions. Prévoir un coût de maintenance sur les scripts de tests automatisés.

Risques projets

Supposé accroître l'efficacité, l'automatisation mal planifiée peut s'avérer contre-productif :

- La complexité des tests peut augmenter. *Quid* de la maintenance du patrimoine et des données de test.
- L'automatisation peut elle-même introduire des erreurs (erreurs d'écriture de scripts, ajout de la dette technique d'un projet de développement, faux-positifs, faux-négatifs)
- Peut détourner des objectifs de tests : l'effort se concentre sur l'automatisation des TNR et non sur la conception de tests de validation pour les nouvelles fonctionnalités.

Limites de l'automatisation du contrôle de la qualité

Enfin, l'automatisation n'est pas une solution magique :

- Tous les tests ne sont pas automatisables.
- Seuls les tests dont le résultat est interprétable par une machine sont automatisables.
- Le test automatisé ne peut que comparer des résultats obtenus à des résultats attendus (fourni par un oracle de test automatisé).
- Le test automatisé ne peut pas remplacer le test exploratoire.

L'automatisation selon le types de tests

I. Pourquoi automatiser les tests?

II. L'automatisation selon le types de tests

1. Tests fonctionnels

- 2. Tests de sécurité applicative
- 3. Tests de performance
- 4. Qualimétrie
- 5. Tests d'utilisabilité



Les facteurs et critères qualité

ISO/IEC 25010

Capacité Fonctionnelle

Complétude fonctionnelle
Exactitude fonctionnelle
Adéquation fonctionnelle

Performance / Rendement

Temps de réponse
Utilisation des ressources
Capacité maximale

Compatibilité

Coexistence
Interopérabilité

Utilisabilité

Facilité de prise en main
Facilité d'apprentissage
Opérabilité
Protection contre les erreurs
Esthétique
Accessibilité

Fiabilité

Maturité
Disponibilité
Tolérance aux pannes
Facilité de recouvrement

Sécurité

Confidentialité
Intégrité
Non-répudiation
Traçabilité
Authenticité

Maintenabilité

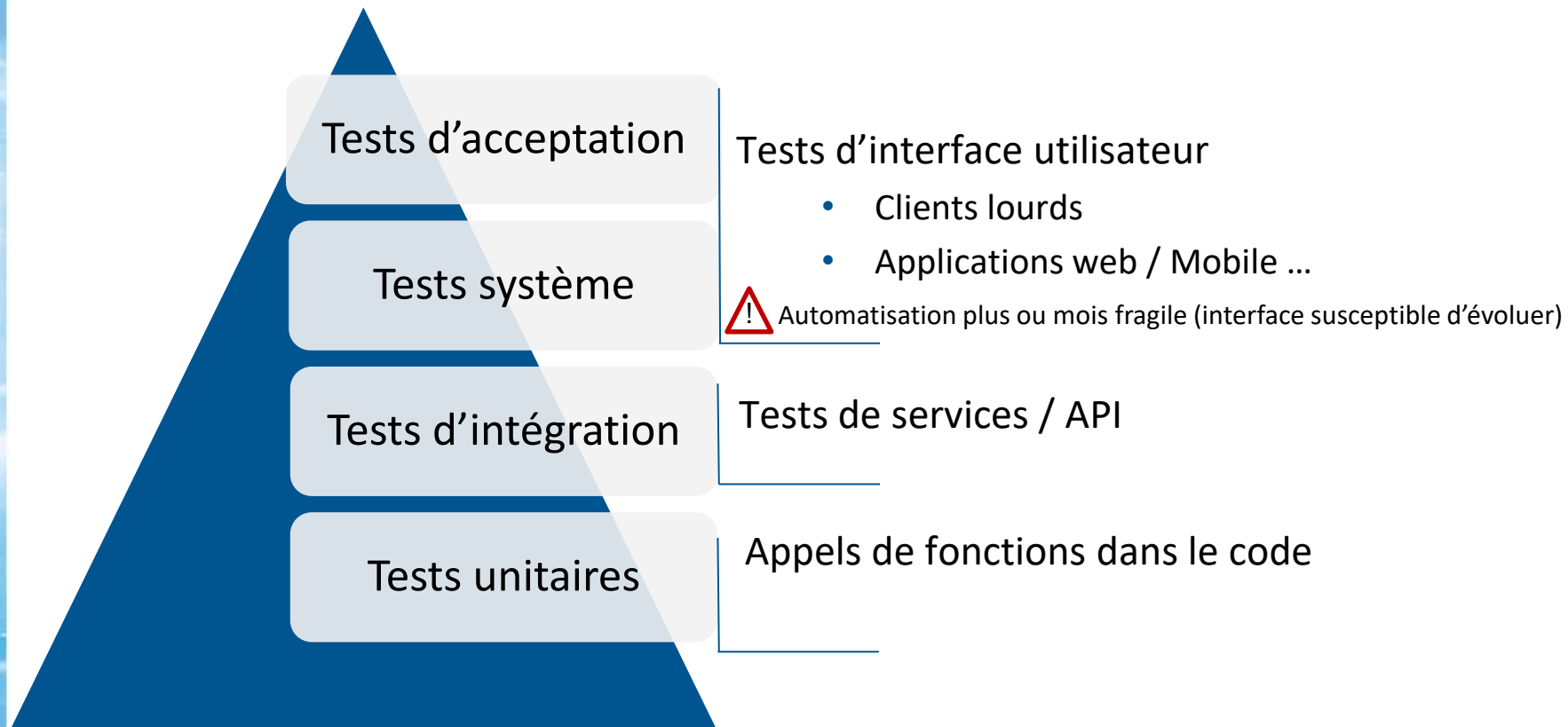
Modularité
Réutilisabilité
Analysabilité
Capacité d'évolution
Testabilité

Portabilité

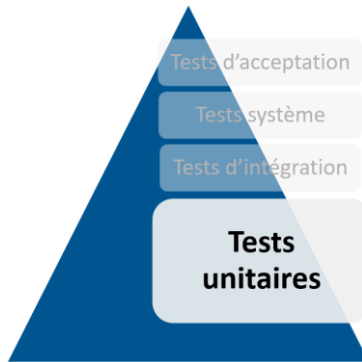
Facilité d'adaptation
Facilité d'installation
Facilité de substitution

Types de tests fonctionnels automatisés

Les tests fonctionnels sont réalisés à tous les niveaux du développement de l'application :



Les tests fonctionnels unitaires



- Conçus par les développeurs, pendant les développements
- Testent des méthodes, des fonctions de code.

Le test unitaire est une fonction (code) qui vérifie qu'une unité de code applicatif ne comporte pas d'erreur de programmation.

Mode opératoire :

→ Soit une méthode addition d'une calculatrice

```
public int addition(int operand1, int operand2) {  
    return operand1+operand2  
}
```

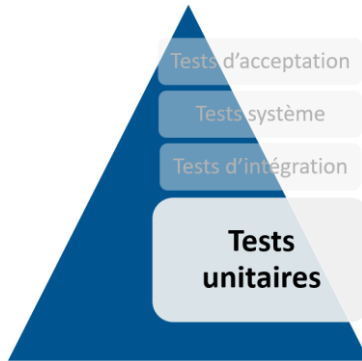
→ Voici un exemple de test unitaire

Jeu de
données
(JDD)

```
@Test  
public void testAddition(){  
    int op1 = 2;  
    int op2 = 2;  
    int somme_attendue = 4;  
    assertTrue(cal.addition(op1,op2) == somme_attendue);  
}
```

Appel de la
méthode et
vérification

Les tests fonctionnels unitaires

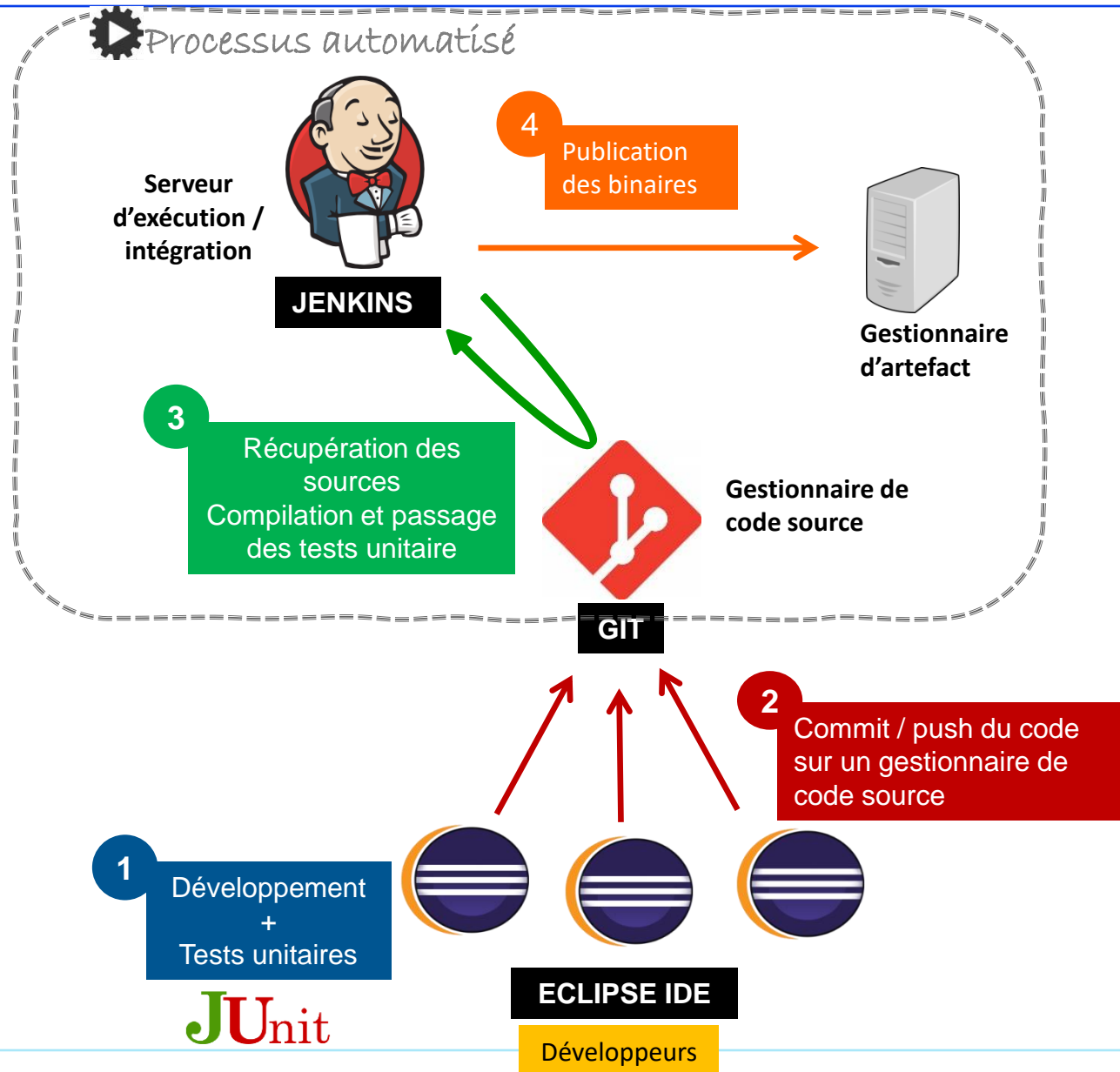


Les tests unitaires sont, par nature, des tests automatisés

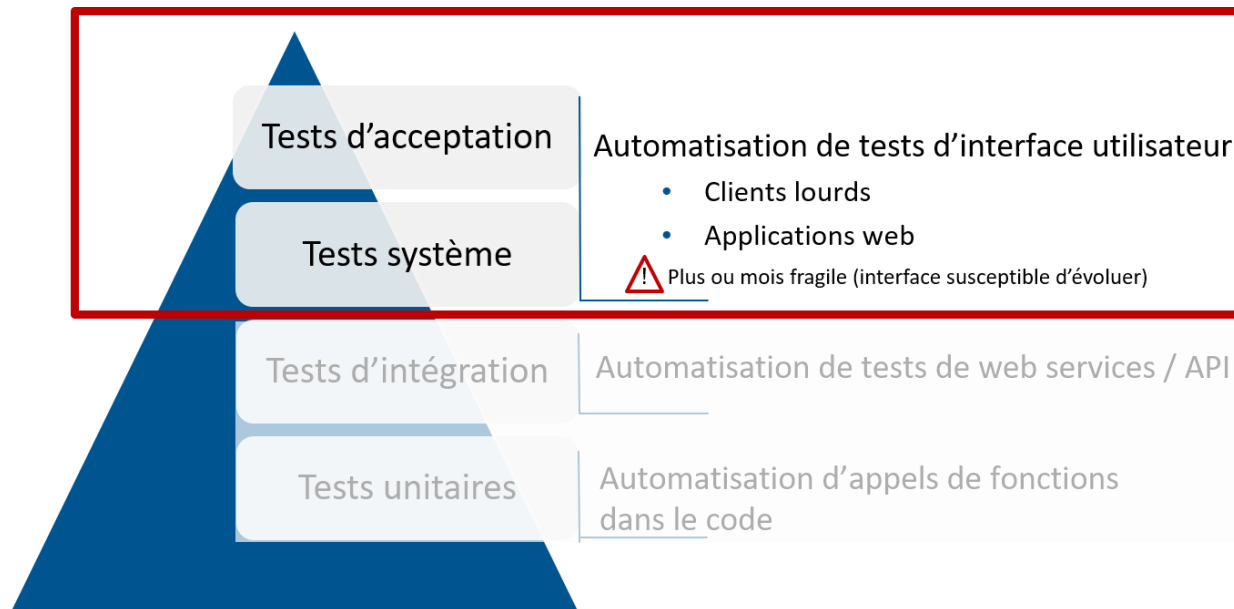
- On ne peut pas jouer un test unitaire « manuellement »....
- Ils sont intégrés au projet de développement
- Ils sont joués automatiquement après compilation du code

Objectif : couverture de test à 100%

Mise en œuvre – Tests fonctionnels unitaires automatisés



Les tests fonctionnels système



Le test fonctionnel de haut niveau vérifie la réaction d'un système à la suite d'une action.

- Tests dynamiques, boîte noire (sans visibilité du code), joués au niveau de l'interface utilisateur.
- Conçus et exécutés manuellement par des testeurs indépendants
- Ne peuvent / doivent pas **tous** être automatisés

Les tests fonctionnels système

Projet d'automatisation des tests fonctionnels système

L'automatisation des tests fonctionnels constitue un projet à part entière, découpé en phases distinctes



Les tests fonctionnels système

Choisir les tests à automatiser

Patrimoine de tests complet



Etape 1 : Identifier les tests de non régression

Patrimoine de tests de non régression



Etape 2 : Identifier les tests éligibles à l'automatisation

Patrimoine de tests éligibles à l'automatisation, par ordre de priorité

Facteurs	Critères de sélection
Risque de régression	Stabilité fonctionnelle
	Complexité technique et qualité du code
	Statistiques sur les anomalies passées
Criticité métier	Importance fonctionnelle
	Fréquence d'utilisation de la fonction
Complexité d'exécution manuelle	Complexité et durée de l'exécution manuelle
	Reproductibilité des tests manuels
Complexité d'automatisation	Technologie de développement de la fonction / du SST
	Interdépendance de la fonction / du SST avec les autres applications du SI
	Complexité et durée d'implémentation du cas de test
	Besoins en maintenance des tests automatisés

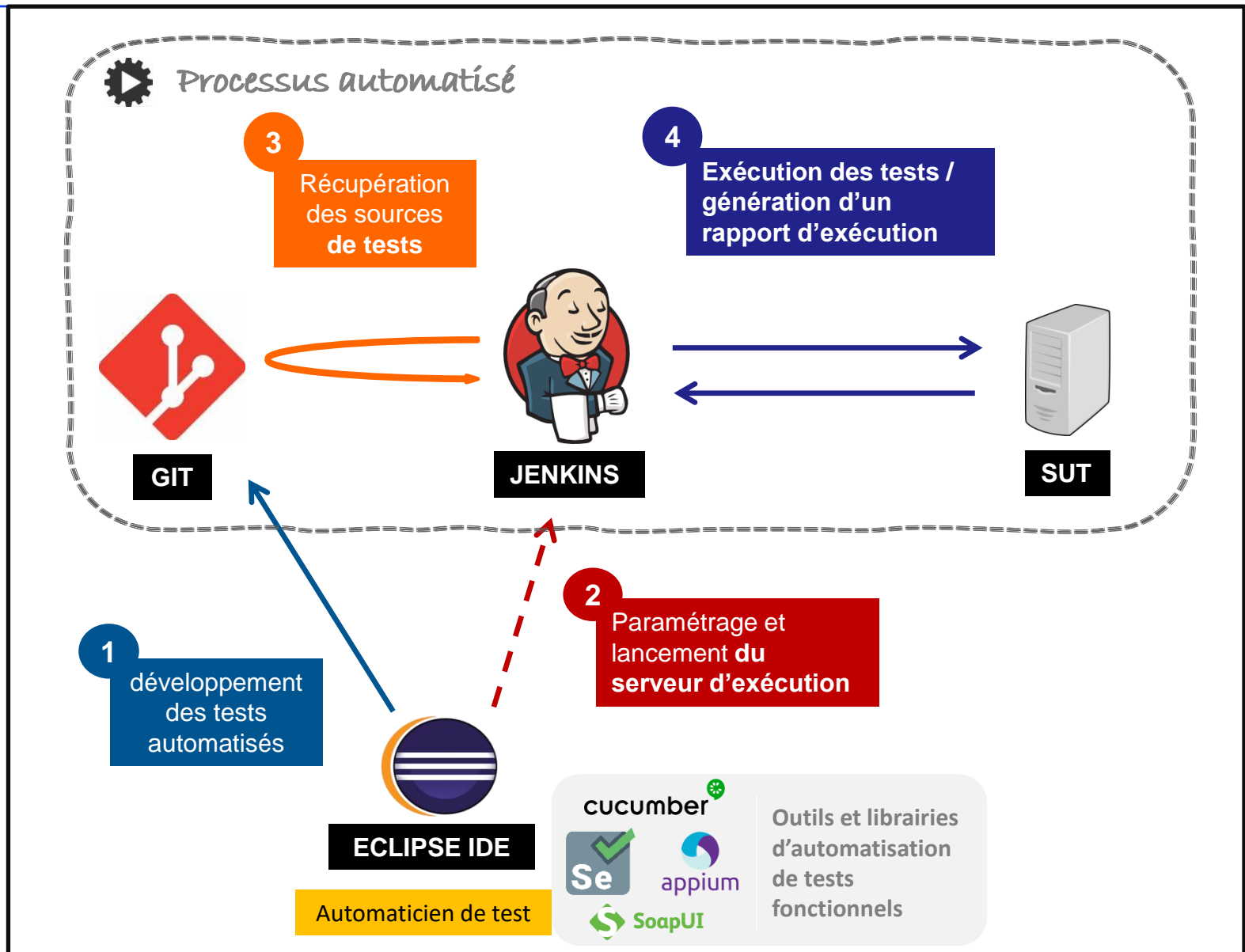
Choisir les outils d'automatisation

Les robots utilisés pour l'automatisation des tests fonctionnels sont nombreux et se distinguent par :

- Leur coût,
- Leur complexité d'utilisation
- Les SUT qu'ils supportent

OUTILS	Mode d'écriture des scripts	Type de licence	SUT supportés
Selenium	Scripting / programmation	Open source	Web
Appium	Scripting / programmation	Open source	Mobile
Katalon	Record / <u>scripting</u>	Commerciale	Web, Mobile, client lourd
UFT	Record / <u>scripting</u>	Commerciale	Web, Mobile, client lourd
Ranorex	Record / <u>scripting</u>	Commerciale	Web, Mobile, client lourd
Sahi	Record / <u>scripting</u>	Open source / Commerciale	Web

Mise en œuvre – Tests fonctionnels système automatisés



I. Pourquoi automatiser les tests?

II. L'automatisation selon le types de tests

1. Tests fonctionnels
2. **Tests de sécurité applicative**
3. Tests de performance
4. Qualimétrie
5. Tests d'utilisabilité

Les facteurs et critères qualité

ISO/IEC 25010

Capacité Fonctionnelle

Complétude fonctionnelle
Exactitude fonctionnelle
Adéquation fonctionnelle

Performance / Rendement

Temps de réponse
Utilisation des ressources
Capacité maximale

Compatibilité

Coexistence
Interopérabilité

Utilisabilité

Facilité de prise en main
Facilité d'apprentissage
Opérabilité
Protection contre les erreurs
Esthétique
Accessibilité

Fiabilité

Maturité
Disponibilité
Tolérance aux pannes
Facilité de recouvrement

Sécurité

Confidentialité
Intégrité
Non-répudiation
Traçabilité
Authenticité

Maintenabilité

Modularité
Réutilisabilité
Analysabilité
Capacité d'évolution
Testabilité

Portabilité

Facilité d'adaptation
Facilité d'installation
Facilité de substitution

Norme ISO/IEC 25010 : Facteurs et critères qualité

La qualité de la sécurité applicative se mesure selon cinq critères

- **Confidentialité** : degré auquel le produit ou système garantit que seules les personnes autorisées ont accès aux données.
- **Intégrité** : degré auquel le produit, système ou composant garantit qu'une donnée n'a pas été modifiée par une entité tierce (accidentellement ou intentionnellement).
- **Non-répudiation** : degré auquel il est possible de prouver que les actions qui ont été effectuées ou les événements qui se sont produits ont bel et bien eu lieu.
- **Traçabilité** : degré auquel il est possible d'identifier l'entité qui est l'auteur des actions effectuées.
- **Authenticité** : degré auquel il est possible de garantir le lien entre identité dans l'application et hors application.

OWASP – Open Web Application Security Project

Pour tester ces critères l'OWASP nous invite à répondre à ces questions :

Collecte d'informations

- *A quel point est-il facile d'obtenir des informations sur le système (serveur, base de données) ou sur l'application (technologie utilisée) qui pourront par la suite être exploitées ?*

Configuration et déploiement

- *La configuration et le mode de déploiement permet-elle d'empêcher certains types d'intrusion ?*

Gestion des identités

- *La définition des rôles et des permissions dans l'application permet-elle de garantir une protection de l'application et de ses données ?*

Authentification

- *Le mécanisme d'authentification est-il contournable? Quelle sécurité garantit-il ?*

Autorisations

- *Est-il possible d'obtenir des autorisations de manière détournée ?*

Gestion des sessions

- *Une fois qu'une session utilisateur est créée, quelles sont les failles qui pourraient rendre possible l'utilisation non-autorisée de cette session par un tiers ou hors du contexte autorisé ?*

OWASP – Open Web Application Security Project

... et encore des questions :

Validation des données

- *Les données sont-elles contrôlées (origine, cohérence, type, valeurs) avant leur utilisation dans le système? Est-il possible d'injecter des données susceptibles de provoquer un dysfonctionnement ou de faire passer du code pour des données ? (cross-site scripting / injection sql / xml bombing)*

Gestion des erreurs

- *L'analyse des erreurs permet-elle d'obtenir des informations susceptibles de permettre une utilisation malicieuse ou détournée du système ?*

Cryptographie

- *Lorsqu'elle est utilisée, à quelle point la cryptographie est-elle fiable ?*

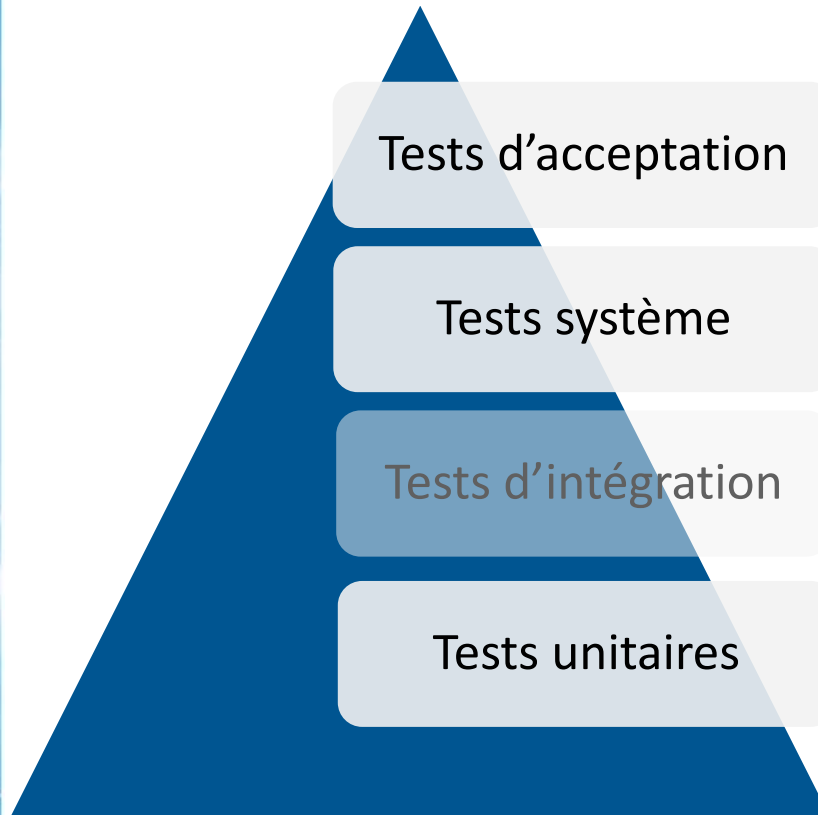
Logique métier

- *La logique métier a-t-elle prévu les vérifications nécessaires pour empêcher les détournements d'utilisation et l'introduction de données incorrectes ou malicieuses dans le système ?*

Client web

- *Côté navigateur, quels sont les vérifications et protections mises en place pour empêcher l'introduction d'éléments malicieux ?*

Types de tests de sécurité



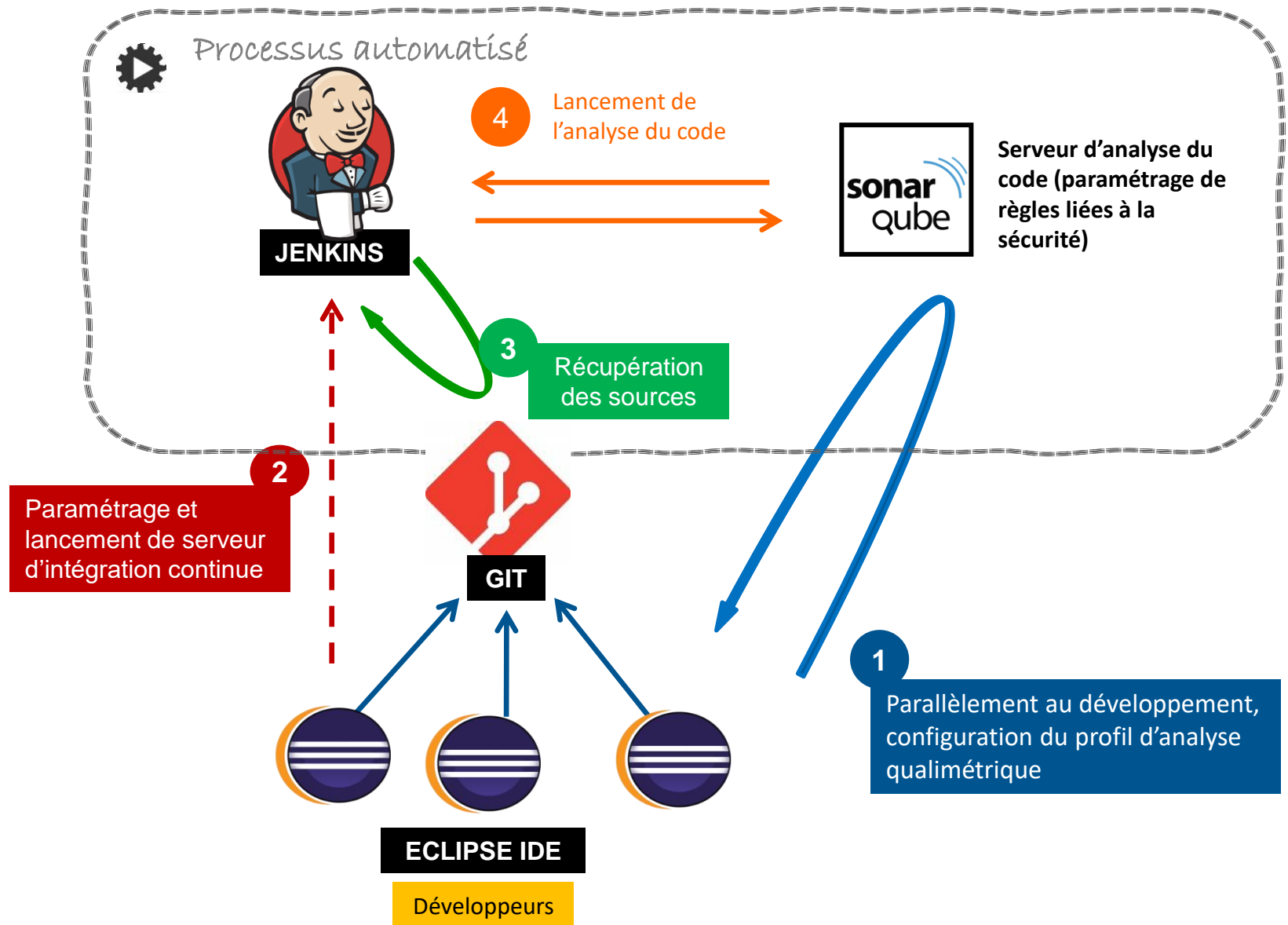
Tests d'intrusion (Pen test)

- Trouver une vulnérabilité à exploiter
- Imaginer une attaque
- Tester l'attaque
- Entrer l'attaque sur une donnée cible
- Exploiter la réponse pour obtenir des informations

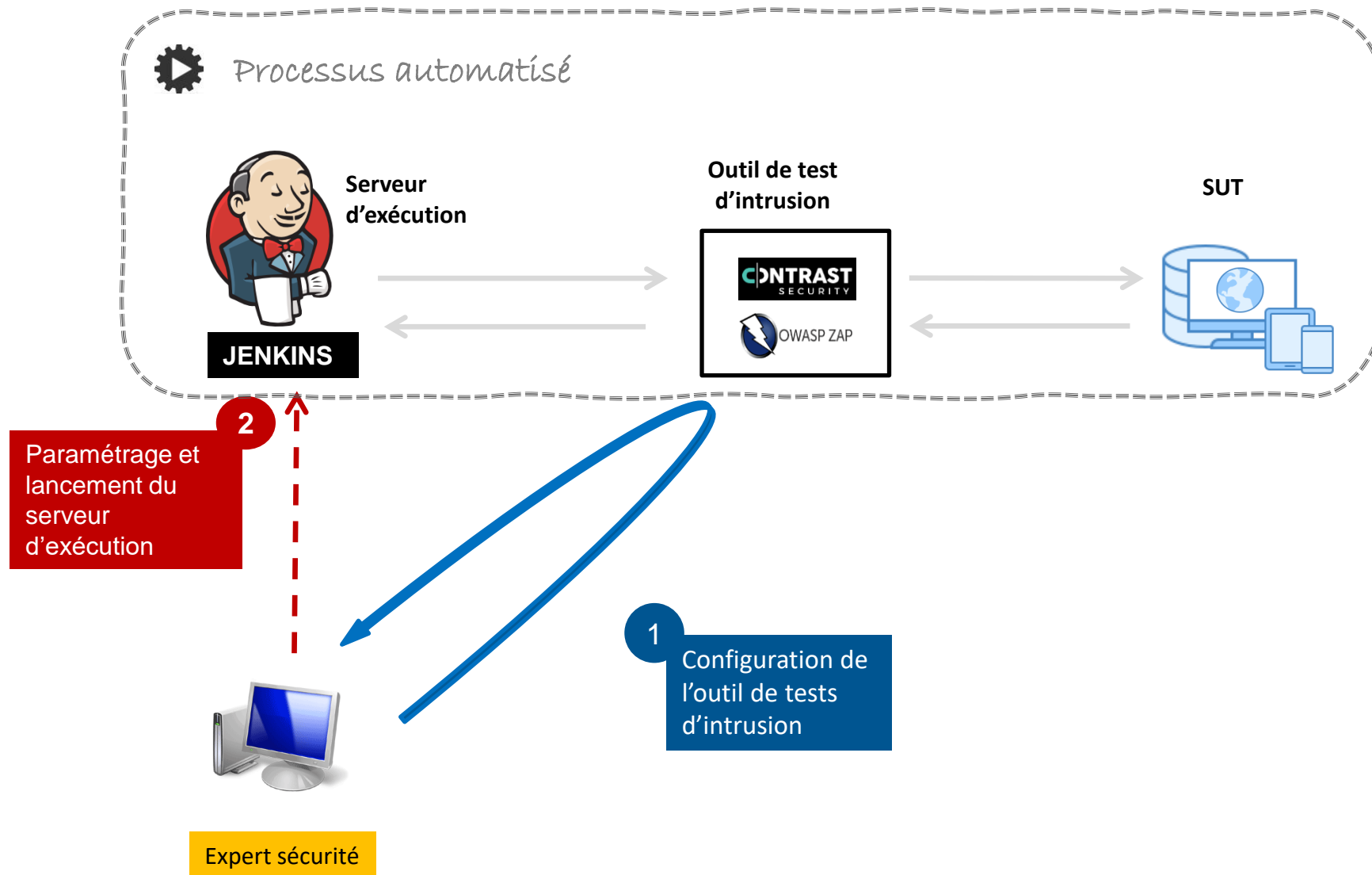
Analyse de code

- Recherche de ces éléments dans le code qui constituent des vulnérabilités

Mise en œuvre – Analyse de sécurité



Mise en œuvre – Tests d'intrusion



La performance

I. Pourquoi automatiser les tests?

II. L'automatisation selon le types de tests

1. Tests fonctionnels
2. Tests de sécurité applicative
- 3. Tests de performance**
4. Qualimétrie
5. Tests d'utilisabilité



Les facteurs et critères qualité

ISO/IEC 25010

Capacité Fonctionnelle

Complétude fonctionnelle
Exactitude fonctionnelle
Adéquation fonctionnelle

Performance / Rendement

Temps de réponse
Utilisation des ressources
Capacité maximale

Compatibilité

Coexistence
Interopérabilité

Utilisabilité

Facilité de prise en main
Facilité d'apprentissage
Opérabilité
Protection contre les erreurs
Esthétique
Accessibilité

Fiabilité

Maturité
Disponibilité
Tolérance aux pannes
Facilité de recouvrement

Sécurité

Confidentialité
Intégrité
Non-répudiation
Traçabilité
Authenticité

Maintenabilité

Modularité
Réutilisabilité
Analysabilité
Capacité d'évolution
Testabilité

Portabilité

Facilité d'adaptation
Facilité d'installation
Facilité de substitution

Types de tests de performance

Le principe

Simuler une activité multi-utilisateur sur une application pour évaluer la performance d'un système

Les différents types de tests de performance



Valider une application pour une charge attendue

Tester les limites de performance d'un système par une politique de charge croissante

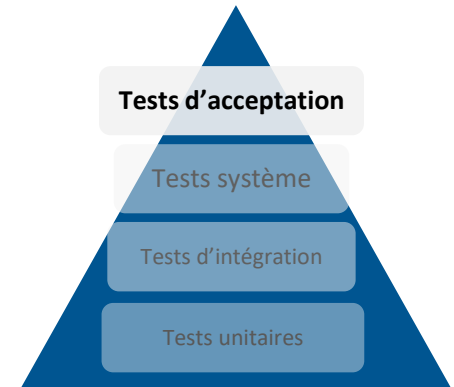
Observer le comportement de l'application soumise à une charge extrême et soudaine

Observer le comportement de l'application sur le long terme (utilisation de la mémoire, des connexions à la BDD, etc...)

Observer les impacts de différentes configurations sur la performance à charge constante

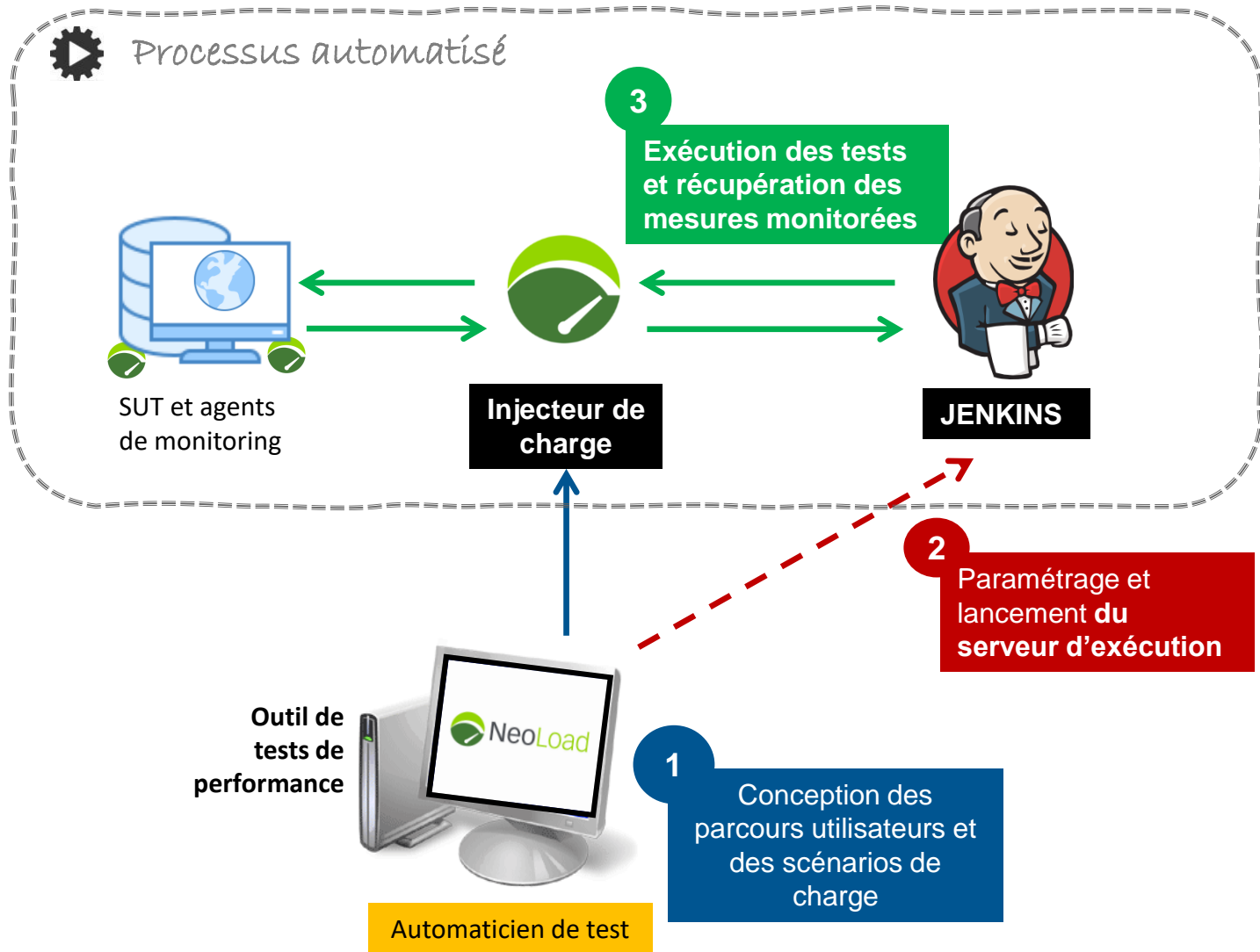
Objectifs techniques et projet

À quoi servent les tests de performance ?



- Identifier un ou plusieurs points faibles causant des pertes de performance dans un système
- Vérifier la conformité d'un système avec des exigences de performance
- Collecter des informations sur la performance d'une application afin de prendre des décisions afin d'améliorer les performances

Mise en œuvre



I. Pourquoi automatiser les tests?

II. L'automatisation selon le types de tests

1. Tests fonctionnels
2. Tests de sécurité applicative
3. Tests de performance
4. **Qualimétrie**
5. Tests d'utilisabilité



Les facteurs et critères qualité

ISO/IEC 25010

Capacité Fonctionnelle

Complétude fonctionnelle
Exactitude fonctionnelle
Adéquation fonctionnelle

Performance / Rendement

Temps de réponse
Utilisation des ressources
Capacité maximale

Compatibilité

Coexistence
Interopérabilité

Utilisabilité

Facilité de prise en main
Facilité d'apprentissage
Opérabilité
Protection contre les erreurs
Esthétique
Accessibilité

Fiabilité

Maturité
Disponibilité
Tolérance aux pannes
Facilité de recouvrement

Sécurité

Confidentialité
Intégrité
Non-répudiation
Traçabilité
Authenticité

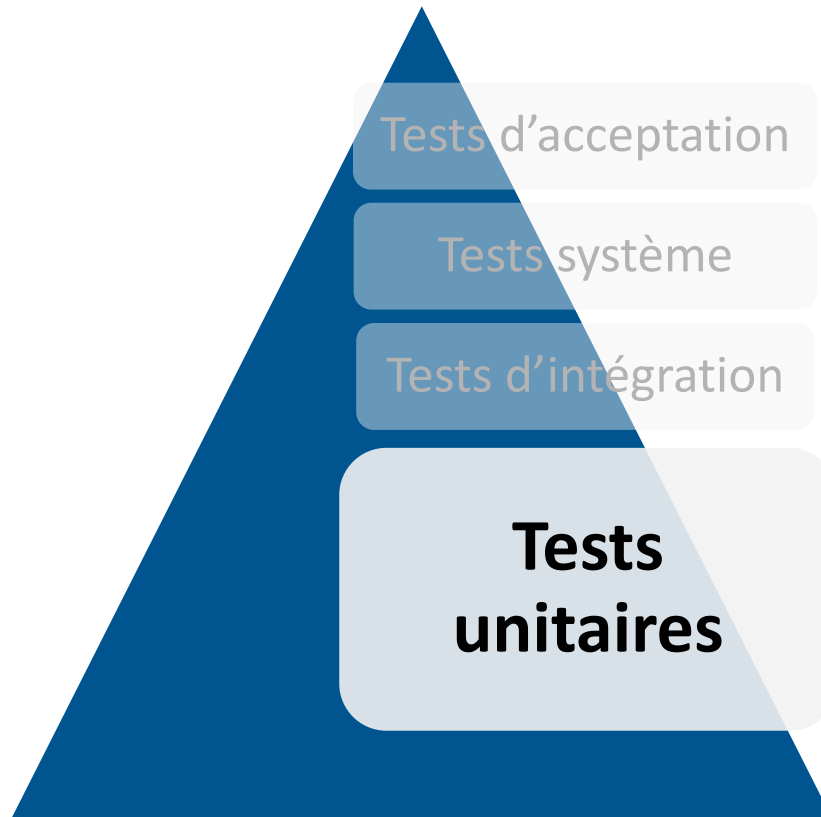
Maintenabilité

Modularité
Réutilisabilité
Analysabilité
Capacité d'évolution
Testabilité

Portabilité

Facilité d'adaptation
Facilité d'installation
Facilité de substitution

Qu'est ce que c'est?



Teste la qualité du code source de l'application

- Revue / inspection de code
- Analyse statique du code source

Qu'est ce que c'est?

Objectifs techniques :

- Trouver des erreurs / points d'attention dans le code
- Trouver des déviations par rapport à un standard
- Observer l'évolution du code entre plusieurs analyses
- Observer les dépendances entre modules
- Repérer les duplications (copier-coller) de code
- Effectuer des mesures de complexité
- Effectuer des mesures de couverture de code par les tests (dynamique)

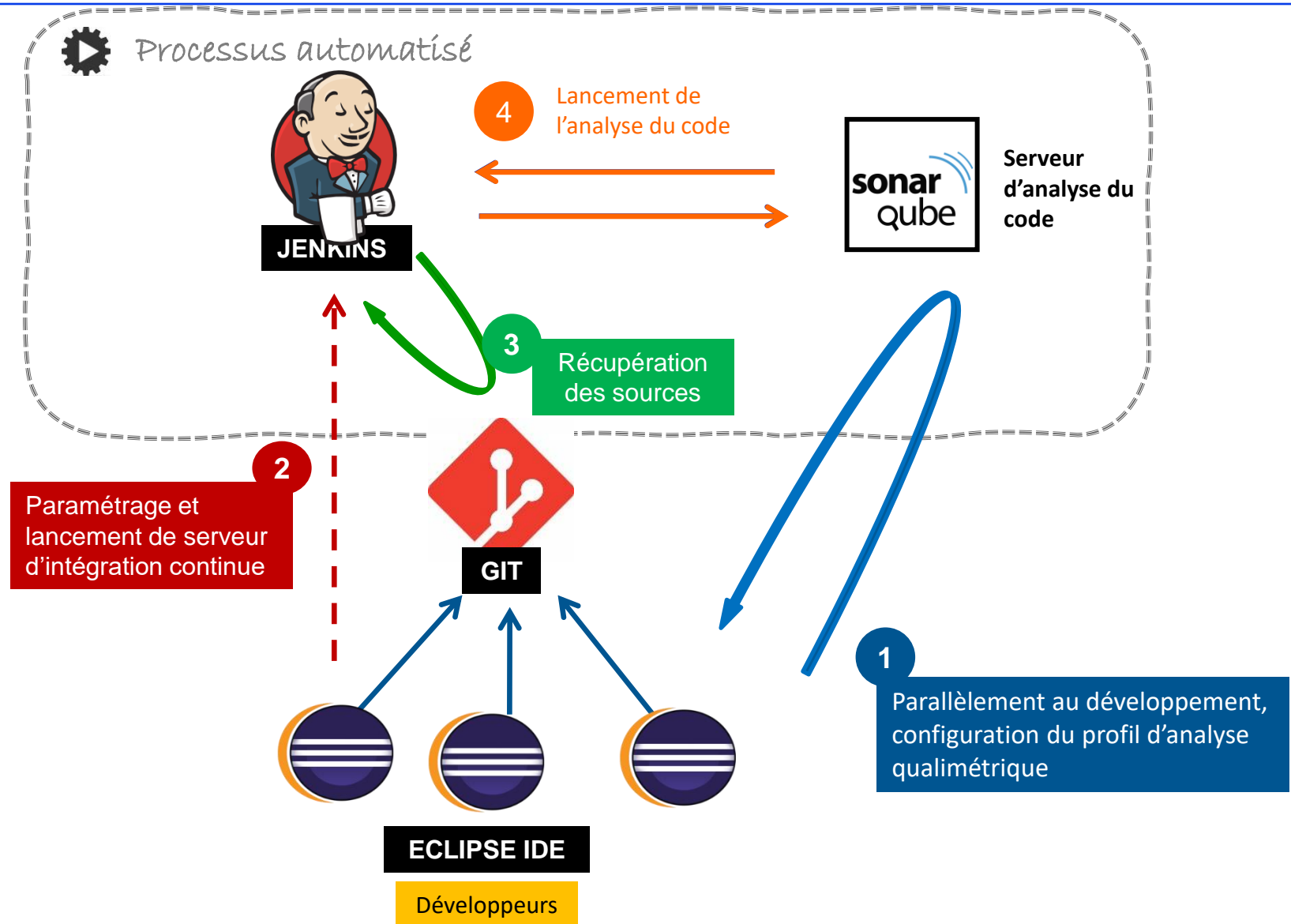


Objectifs projet :

- Mettre en place un suivi de la qualité de la production d'une équipe de développement.
- Bloquer une livraison si la qualité du code n'atteint pas le niveau requis



Exemple de Mise en Œuvre des analyse de code automatisées



1^{ère} étape, éditer les règles

- Une règle décrit une configuration :
 - qui doit être détectable par un analyseur de code
 - qui doit être signalée comme problème ou comme point d'attention lorsqu'elle est repérée
- Lorsque du code ne respecte pas une règle, on parle de **violation**.



Exemple de règle qualimétrique

"<title>" should be present in all pages

Titles are important because they are displayed in search engine results as well as the browser's toolbar.

This rule verifies that the `<head>` tag contains a `<title>` one, and the `<html>` tag a `<head>` one.

Noncompliant Code Example

```
<html>          <!-- Non-Compliant -->

<body>
...
</body>

</html>
```

Compliant Solution

```
<html>          <!-- Compliant -->

<head>
  <title>Some relevant title</title>
</head>

<body>
...
</body>

</html>
```

Notion de règle

Chaque règle dispose :

- d'une **criticité** (info, mineure, majeure, bloquante)
- d'un **type** (erreur impactant la maintenabilité, la sécurité ou encore causant un bug)
- d'un **domaine** (par exemple le langage pour lequel elle s'applique)

"<title>" should be present in all pages

Titles are important because they are displayed in search engine results as well as the browser's toolbar.

This rule verifies that the `<head>` tag contains a `<title>` one, and the `<html>` tag a `<head>` one.

Noncompliant Code Example

```
<html>          <!-- Non-Compliant -->

<body>
...
</body>

</html>
```

Compliant Solution

```
<html>          <!-- Compliant -->

<head>
  <title>Some relevant title</title>
</head>

<body>
...
</body>

</html>
```

Modifiable au
niveau du profil
de règle

- **MAJEURE**
- **BUG**
- **EXPERIENCE-UTILISATEUR**
- **HTML**

Définis à la création de la
règle – non modifiables

2^{ème} étape, créer un profil de règles

- Un ensemble de règles permet de constituer un profil de règles (ou profil qualité).
- La création d'un profil de règles propre à un projet permet de filtrer les règles a ne pas prendre en compte dans les analyses.
- Par défaut, SonarQube propose des profils pour chaque langage de programmation : les "Sonarway".

Pour exemple, le Sonarway java intègre 292 règles sur les 440 disponibles



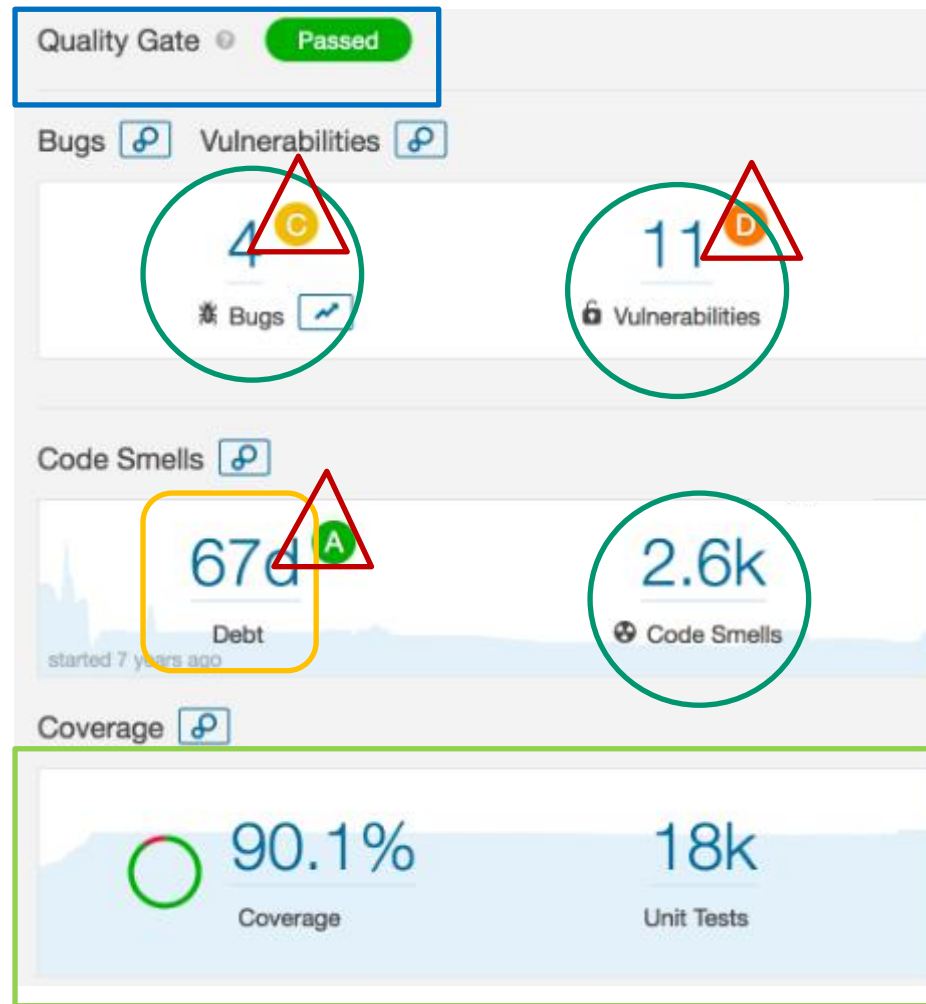
3^{ème} étape, configurer une barrière qualité

- Définir des objectifs :
 - 0 violations bloquantes
 - Pas de dégradation de la qualité (depuis la dernière analyse)
 - Pas de dégradation de la maintenabilité
 - Pas de violation de sécurité introduite depuis la dernière version
- En cas de non-respect des critères définis dans la barrière, on peut :
 - Envoyer des avertissements
 - Refuser la contribution d'un développeur
 - Bloquer la livraison d'un logiciel



Lire une analyse Sonar

Dernière étape, lancer le tir qualimétrique



Statut de l'analyse



Nombre de violations de chaque type



Dette technique



Note globale



Section couverture du code par les tests unitaire...

L'intérêt de la revue de code manuelle

La revue de code est une activité **chronophage** et **complexe**.
Pourquoi continuer les revue de code manuelles alors qu'il existe des outils d'analyse automatisé ?

La revue de code possède de nombreux avantages auxquels l'analyse automatisée ne saurait prétendre :

- Renforce l'appropriation collective du code :
 - Contribue à la connaissance collective du projet de développement
 - Réduit le risque d'une spécialisation trop forte où seul l'auteur du code est capable de le modifier.
- Favorise la montée en compétence :
 - Former les profils juniors en partageant les standards en place
 - Partage autour des nouvelles approches / technos (veille)
- Améliore la qualité des échanges entre les développeurs.

Tests d'utilisabilité

I. Pourquoi automatiser les tests?

II. L'automatisation selon le types de tests

1. Tests fonctionnels
2. Tests de sécurité applicative
3. Tests de performance
4. Qualimétrie
5. **Tests d'utilisabilité**



Les facteurs et critères qualité

ISO/IEC 25010

Capacité Fonctionnelle

Complétude fonctionnelle
Exactitude fonctionnelle
Adéquation fonctionnelle

Performance / Rendement

Temps de réponse
Utilisation des ressources
Capacité maximale

Compatibilité

Coexistence
Interopérabilité

Utilisabilité

Facilité de prise en main
Facilité d'apprentissage
Opérabilité
Esthétique
Accessibilité

Fiabilité

Maturité
Disponibilité
Tolérance aux pannes
Facilité de recouvrement

Sécurité

Confidentialité
Intégrité
Non-répudiation
Traçabilité
Authenticité

Maintenabilité

Modularité
Réutilisabilité
Analysabilité
Capacité d'évolution
Testabilité

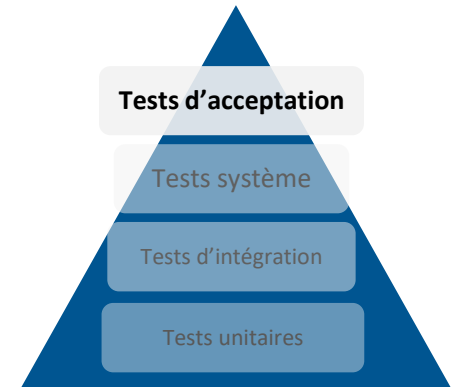
Portabilité

Facilité d'adaptation
Facilité d'installation
Facilité de substitution

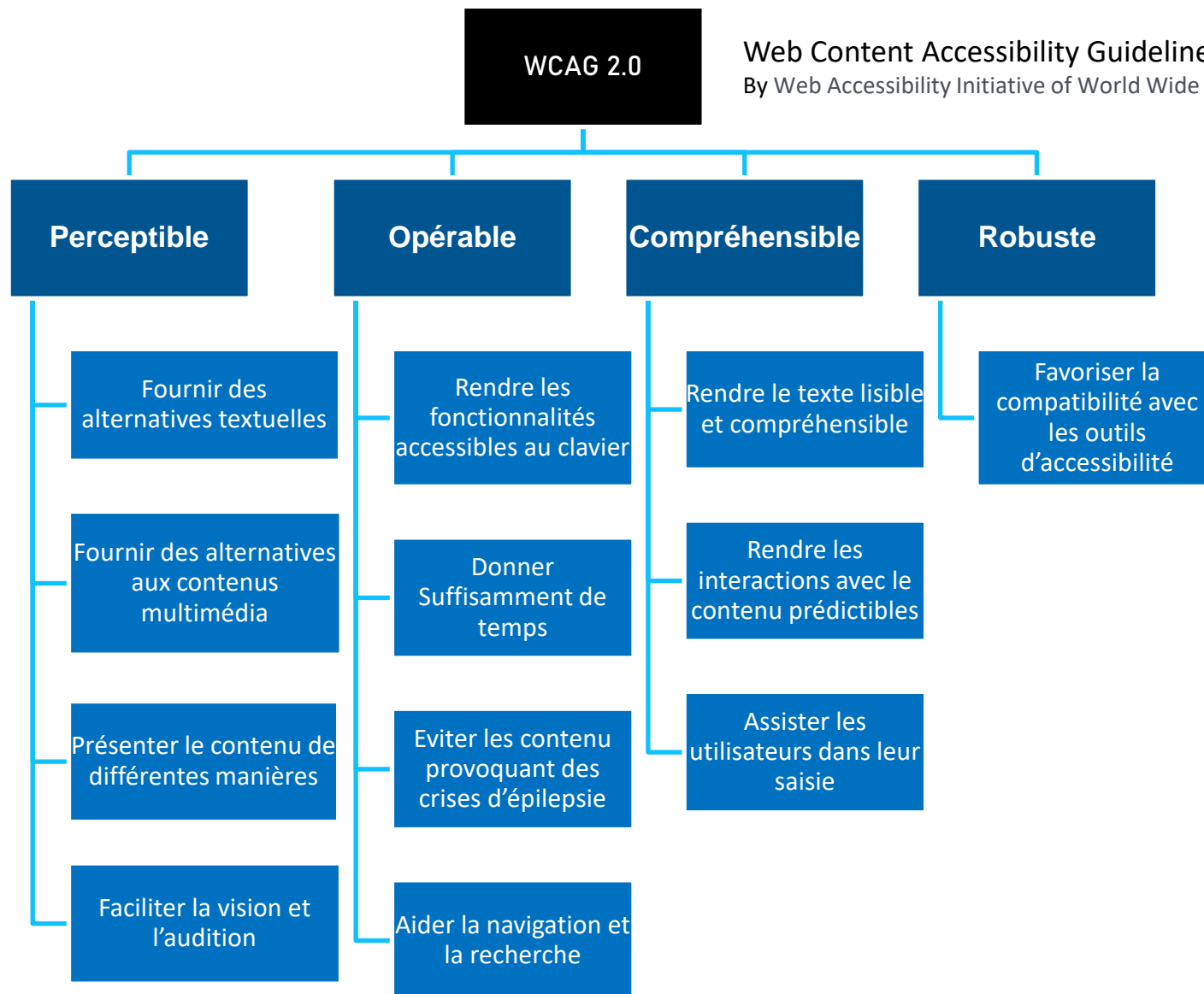
Tests d'utilisabilité

Aussi appelés tests utilisateurs

- L'application est testée directement par les utilisateurs finaux (ou des spécialistes de l'utilisabilité : UX).
- Ce type de test permet d'évaluer :
 - L'ergonomie ;
 - La réponse émotionnelle : l'utilisateur a-t-il apprécié le logiciel, est-il confiant, stressé...
- Les tests utilisateurs n'ont pas vocation à couvrir l'ensemble des fonctionnalités et règles de gestion de l'application.
- Ces tests ne sont évidemment pas éligibles à l'automatisation !



Des règles pour l'accessibilité des contenus web



Tests d'utilisabilité

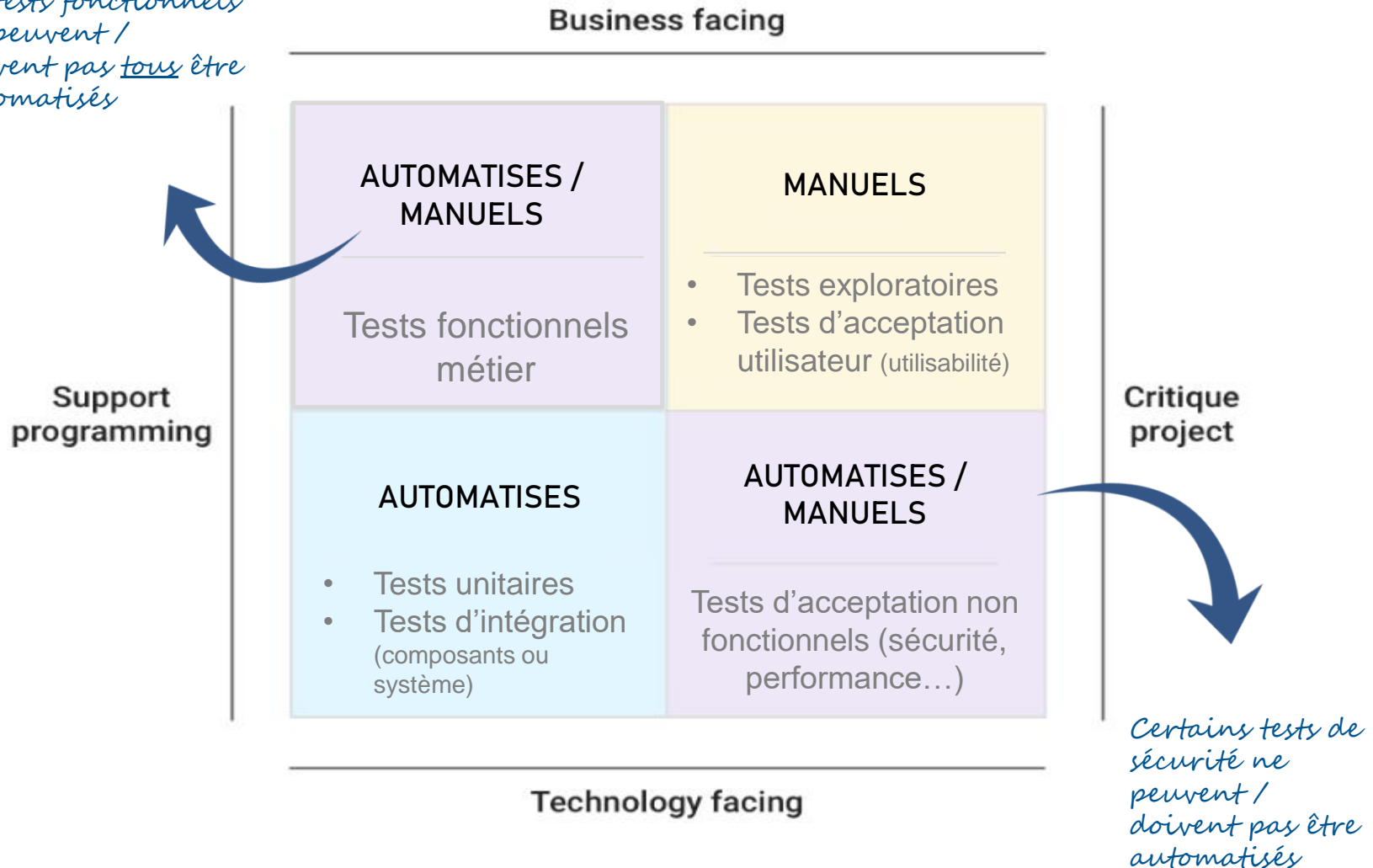
- I. Pourquoi automatiser les tests?
- II. L'automatisation selon le types de tests
 - 1. Tests fonctionnels
 - 2. Tests de sécurité applicative
 - 3. Tests de performance
 - 4. Qualimétrie
 - 5. Tests d'utilisabilité

III. Conclusion




L'automatisation selon le types de tests

*Au niveau système,
les tests fonctionnels
ne peuvent /
doivent pas tous être
automatisés*



L'automatisation selon les niveaux



NIVEAUX	QUI ?	QUOI ?	AUTOMATISATION DU PATRIMOINE
ACCEPTATION	UTILISATEURS EXPLOITATION	Comportement ... Performance / Fiabilité / Sécurité...	15%
SYSTÈME	TEST	Fonctionnalités... Livrables / Installations / paramétrage...	30%
INTEGRATION	DEV	Capacité d'échange information et données Assemblage composants	60%
UNITAIRE	DEV	Qualité du code Exécution des fonctions et méthodes	90%