



Introduction au test logiciel

Fabrice Ambert, Fabrice Bouquet
prenom.nom@femto-st.fr

Automatisation des tests logiciels

- Résumé:
 - apprêhender les méthodes et outils pour automatiser les tests logiciels (tests unitaires, fonctionnels, d'intégration, de charge) afin de gagner en productivité et en sûreté pour le développement logiciel
- Durée:
 - 4 jours
- Pré-requis:
 - connaissances de base en test logiciel et capacité à effectuer des développements logiciels objet

Bienvenue



Faisons connaissance

- Votre formateur
- Pour chaque participant
 - Vos activités actuelles ?
 - Votre expérience ?
 - Sur des projets de développement de logiciels et systèmes d'information
 - Sur les activités de test (Rôles, Outils utilisés)
 - Vos attentes pour cette formation ?

Contenu de la formation

Programme standard

Introduction, rappels sur le processus du test

Automatisation de la gestion des tests

Automatisation des tests unitaires

Automatisation des tests d'intégration

Automatisation du test fonctionnel

Automatisation des tests système

Synthèse





Déroulement de la formation (3 jours)

Alternance de Cours et de Pratique

Cours	Thèmes	Durée	Exercices	Outils	Durée
1. Introduction, rappels sur le processus du test logiciel	Types, Niveaux, Méthodes	J1, 2h			
2. Automatisation des tests unitaires	Couverture de Code	J1, 1.5h	Intégrés au cours Cahier d'exercices	Junit, mockito, Jacoco	J1, 3h
3. Automatisation des tests d'intégration	Stratégie, Intégration continue	J1, 1h	Cahier d'exercices	Maven, Jenkins, SOnarQube	J2, 3h
4. Test fonctionnel et son automatisation	Partitions, Limites, Combinatoire	J2, 1.5h	Intégrés au cours	FitNesse, Concordion Selenium, HTMLUnit	J2, 3h J3, 2h
5. Automatisation de la gestion des tests	Conception, exécution, suivi	J3, 2h	Cahier d'exercices	Squash TM	J3, 3.5h

1. Introduction, rappels sur le processus du test logiciel

Rôle du test dans le processus de développement.

Les tests : unitaires, fonctionnels, etc.

Les différentes méthodes de test.

Processus de test et stratégie de test.

Outils et méthodes intervenant à différentes étapes

Rôle du test dans le processus de développement



Les bugs sont nombreux et couteux pour la société possédant le logiciel, ses fournisseurs, ses clients, les utilisateurs finaux...

The screenshot shows a Google search results page with the following details:

- Search Query:** panne informatique 2016
- Number of Results:** Environ 6 810 000 résultats (0,37 secondes)
- First Result:**
 - Title:** Fin de la panne mondiale qui a paralysé Delta Airlines - Le Figaro
 - Source:** www.lefigaro.fr > ECONOMIE
 - Summary:** 8 août 2016 - «Delta Airlines fait face à une panne informatique qui affecte les vols programmés ce matin. Les vols en Le 09/08/2016 à 11:05. Alerter ...»
- Second Result:**
 - Title:** Une panne majeure sur le réseau informatique a perturbé le CHU de ...
 - Source:** www.ladepeche.fr > Grand Sud > Haute-Garonne > Toulouse > Santé
 - Summary:** 1 sept. 2016 - Publié le 01/09/2016 à 16:42. Une panne majeure sur le réseau informatique a perturbé le CHU de Toulouse ... La panne affecte tout le CHU.

Les failles de sécurité sont également nombreuses

Problèmes de sécurité (Figaro 21/09/16)

En avril 2015, la chaîne TV Monde avait été contrainte de suspendre la diffusion de ses programmes après l'attaque informatique.

Geoffrey VAN DER HASSELT / ANALYST AGENCY

Les attaques informatiques sont en forte hausse

(Le Figaro
21 septembre 2016)

Le secteur médical est particulièrement touché par des failles de sécurité, en France et dans le monde.

LUCIE RONFAUT @lucieronfaut

INTERNET Sur les services en ligne, au sein des gouvernements ou même dans les hôpitaux. Les attaques informatiques sont en hausse de trois à quatre endroits de notre vie quotidienne. En France, 554 millions de jeux de données personnelles ont été dérobés au premier semestre 2016 par Gemalto, entreprise spécialisée dans la sécurité en ligne, qui publie mardi son étude Breach Level Index. Cela représente une augmentation de 15 % en six mois, par rapport aux deux derniers derniers semestres 2015. En tout, plus de 554 millions de jeux de données ont été compromis sur les six premiers mois de 2016, contre 424 millions en 2015. La fraude à l'identité est la première raison de ces attaques, avec 82 % d'incidents. Viennent ensuite le piratage de comptes bancaires, avec 155 failles exploitées.

Le secteur médical a été particulièrement touché par le phénomène, avec plus de 263 incidents rapportés pour le seul début 2016, en hausse de 25 % sur les six derniers mois. Ces données sont particulièrement sensibles et doivent être protégées. « Un vol impliquant 100 millions de noms d'utilisateurs sera sans doute gravant que celui qui concerne la violation de comptes comprenant des données de sécurité sociale ou d'autres informations personnelles peuvent être utilisées à surveiller leurs réseaux, comme les opera-

teurs télécoms, les banques ou l'aéronautique française», doit pousser les autres à adopter des règles plus strictes en matière de sécurité et à notifier à l'autorité compétente.

D'après Gemalto, l'Amérique du Nord, et plus particulièrement les Etats-Unis, reste la région la plus touchée par les attaques informatiques. Plus de 79 % des incidents concernent les services américains. L'Europe, quant à elle, vient ensuite. L'Europe concerne plus de 57 % du nombre de jeux de données dérobés depuis le début de l'année. Viennent ensuite le secteur technologique (applications, sites Web, etc.), qui représente 16 % des informations volées.

Cet équilibre pourrait néanmoins rapidement bouleverser au fil et à mesure de la numérisation des données de santé, un processus qui s'accélère grâce au développement des objets connectés et de l'intelligence artificielle au service du médical.

Des nouveaux réflexes

En juillet, l'Agence nationale de la sécurité des systèmes d'information (Anssi) a déclaré le secteur de l'opérateur de télécommunications d'importance vitale devant renforcer leur système informatique pour se protéger des attaques. Sont également concernés les gestionnaires d'eau, d'alimentation, le ministère de l'Intérieur, la justice. D'autres secteurs sont déjà habitués à surveiller leurs réseaux, comme les opera-

teurs télécoms, les banques ou l'aéronautique française. Le nouveau cadre juridique français doit pousser les autres à adopter des règles plus strictes en matière de sécurité et à notifier à l'autorité compétente.

D'après Gemalto, l'Amérique du Nord, et plus particulièrement les Etats-Unis, reste la région la plus touchée par les attaques informatiques. Plus de 79 % des incidents concernent les services américains. L'Europe, quant à elle, vient ensuite. L'Europe concerne plus de 57 % du nombre de jeux de données dérobés depuis le début de l'année. Viennent ensuite le secteur technologique (applications, sites Web, etc.), qui représente 16 % des informations volées.

Cet équilibre pourrait néanmoins rapidement bouleverser au fil et à mesure de la numérisation des données de santé, un processus qui s'accélère grâce au développement des objets connectés et de l'intelligence artificielle au service du médical.

LES ATTAQUES INFORMATIQUES EN 2016

SOURCE : BREACH LEVEL INDEX, GEMALTO

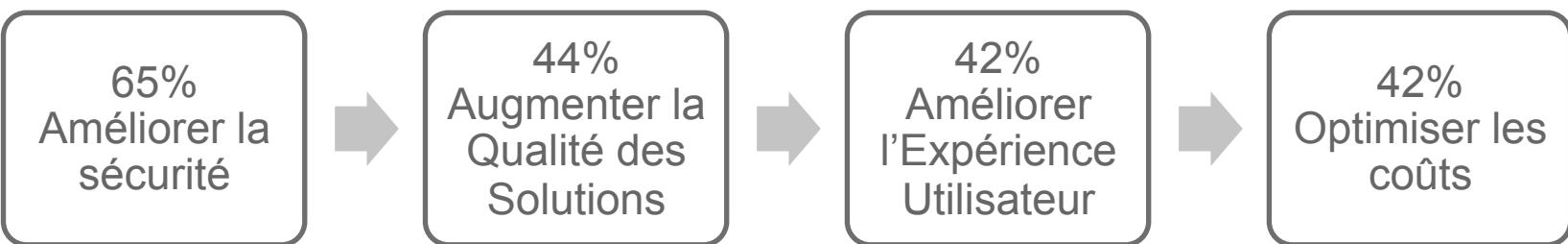
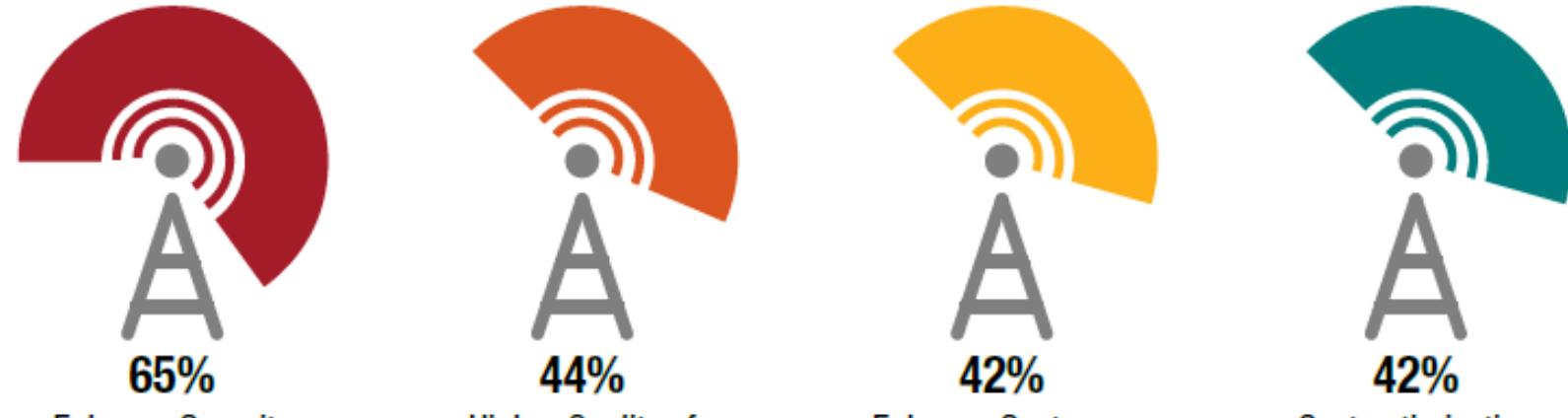
Indicateur	Valeur
vols de données	974
millions de jeux de données dérobés	554
d'augmentation entre le deuxième semestre 2015 et le premier semestre 2016	15 %



Les DSI ont des objectifs directement liés au test

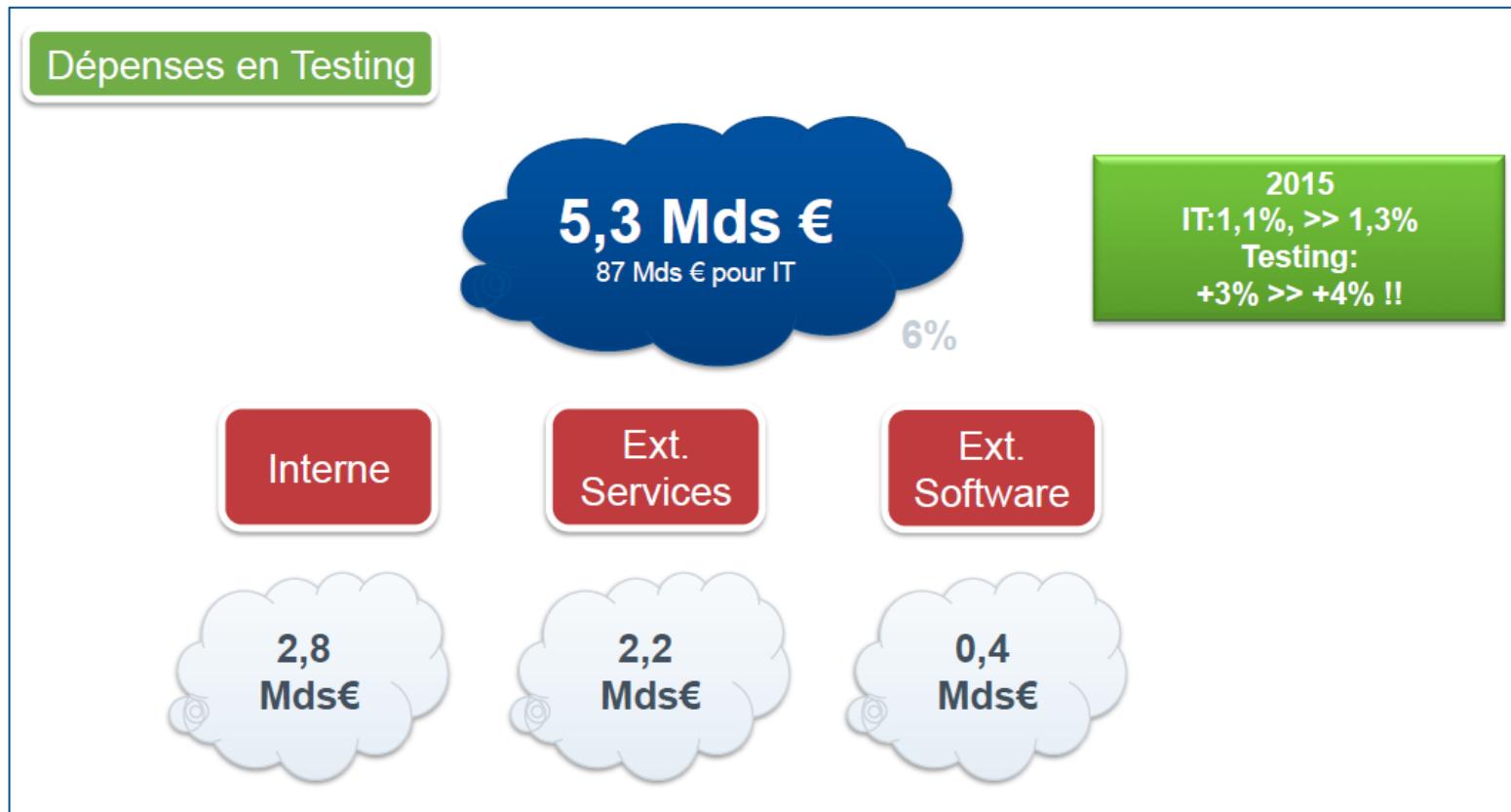


Source: « World Quality Report 2016-2017, Sogeti »



Le Test continue a être un investissement nécessaire et important

Une place très importante dans l'IT



Source: PAC 2016, Arnold Aumasson



Définitions du test

- « *Le test est l'exécution ou l'évaluation d'un système ou d'un composant par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus.* »
IEEE (Standard Glossary of Software Engineering Terminology)
- « *Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts.* »
G. Myers (The Art of Software testing)
- « *Testing can reveal the presence of errors but never their absence.* » Edsgar W. Dijkstra (Notes on Structured Programming)



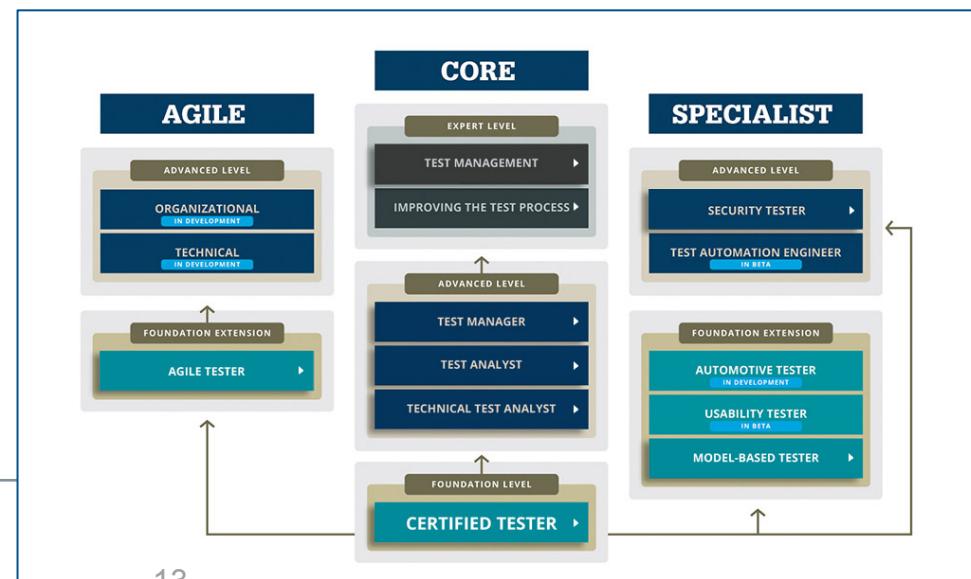
La réalité du test

- Le test constitue aujourd’hui le vecteur principal de l’amélioration de la qualité du logiciel
- Actuellement, le test dynamique est la méthode la plus diffusée
- Il peut représenter jusqu’à 60 % de l’effort complet de développement d’un logiciel
- Coût moyen de l’activité de test :
 - 1/3 durant le développement du logiciel
 - 2/3 durant la maintenance du logiciel

Industrialiser et professionnaliser le test logiciel avec des compétences spécifiques

Un schéma mondial de Formations et Certifications en Test Logiciel

- ISTQB: International Software Testing Qualifications Board
 - www.istqb.org
 - Un parcours de formations reconnu dans le monde entier
 - Près de **500 000 certificats délivrés dans le monde**



Industrialiser et professionnaliser le test logiciel avec des compétences spécifiques

Une implantation en France très importante



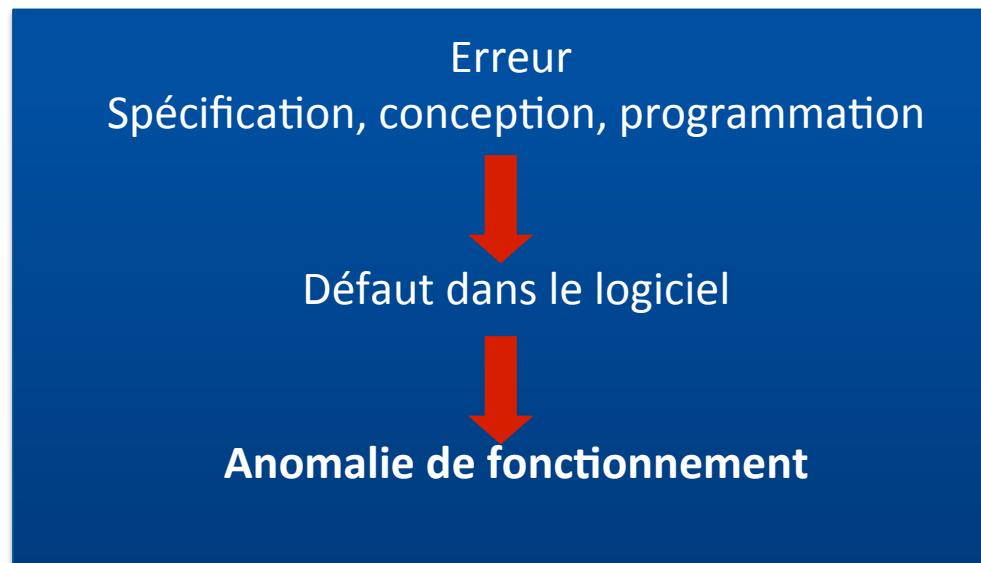
- CFTL: Comité Français des Tests Logiciels
 - www.cftl.fr
 - Déploiement en France
 - Des certifications ISTQB
 - De certifications connexes au test
 - Près de **10000 certificats délivrés en France**





Motivations du test

- Complexité croissante des architectures et des comportements
- Coût d'un bug (Ariane 5, carte à puces allemande bloquée, prime de la PAC...)



- Coût des bugs informatiques : ≈ 60 milliards \$ / an
- 22 milliards économisés si les procédures de test de logiciels étaient améliorées.

(source : NIST - National Institute of Standards and Technology)

Les tests : unitaires, fonctionnels, etc.

- V & V
 - Validation : Est-ce que le logiciel offre les services attendues ?
 - Vérification : Est-ce que les artefacts utilisés sont corrects ?
- Méthodes de V & V
 - Test statique : Revue de code, de spécifications, de documents de design
 - Test dynamique : Exécuter le code pour s'assurer d'un fonctionnement correct
 - Vérification symbolique : Run-time checking, Execution symbolique,
...
 - Vérification formelle : Preuve ou model-checking d'un modèle formel

Les tests : unitaires, fonctionnels, etc

La pratique du test

- Le test appartient à l'activité de Validation du logiciel :
est-ce-que le logiciel fait les choses bien et les bonnes choses ?
- Activité historiquement peu populaire en entreprise
- Difficultés d'ordre psychologique ou « culturel » :
 - L'activité de programmation est un processus constructif : on cherche à établir des résultats corrects
 - Le test est un processus destructif : un bon test est un test qui trouve une erreur



Les différentes méthodes de test

Test statique

Traite le code du logiciel sans l'exécuter sur des données réelles.

Test dynamique

Repose sur l'exécution effective du logiciel pour un sous ensemble bien choisi du domaine de ses entrées possibles.



Test statique

Définitions

Compilateur

- Outil logiciel qui traduit un programme exprimé dans un langage de haut niveau dans son équivalent en langage machine [IEEE 610]

Complexité

- Degré par lequel un composant ou système a une conception et/ou une structure interne difficile à comprendre, maintenir et vérifier



Test statique

L'analyse statique trouve des défauts (et non des défaillances)

- Le code n'est pas exécuté mais des outils peuvent être utilisés par des profils développeur
- La compilation peut être considérée comme du test statique outillé
- Des défauts difficiles à trouver avec des tests dynamiques peuvent être détectés
 - Dépendance dans des modèles
 - Violation de règles de sécurité et programmation
 - Maintenabilité difficile
 - Code mort
 - Variable jamais utilisée
 - Boucles infinies



Test statique

Exemples :

- *Lectures croisées / inspections*
Vérification collégiale d'un document (programme ou spécification du logiciel)
- *Analyse d'anomalies*
Corriger de manière statique les erreurs (typage impropre, code mort, ...)

Avantages :

- Méthodes efficaces et peu coûteuses
- 60% à 95% des erreurs sont détectées lors de contrôles statiques

Inconvénients :

- Ne permet pas de valider le comportement d'un programme au cours de son exécution

→ Les méthodes de test statiques sont nécessaires, mais pas suffisantes



Test statique

Exemple de rapport Sonar

Sonar - HR4U - Windows Internet Explorer
http://localhost:9000/dashboard/index/1?did=1 Sonar - HR4U

Configuration Log in Search

Dashboards Projects > HR4U

Dashboard

- Hotspots
- Reviews
- Time Machine
- Components
- Violations Drilldown
- Clouds
- Design
- Libraries

sonar

Version 1.0 - Segunda, 22 de Abril de 2013, 16:35h Time changes...

Lines of code 163.272 207.717 lines 54.766 statements 592 files	Classes 1.037 7.300 methods 3.432 accessors	Violations 12.549 Rules compliance 69,6%
Comments 7,2% 12.593 lines +2.007 blank 7,3% docu. API 5.208 undocu. API 0 commented LOCs	Duplications 21,6% 44.854 lines 175.487 blocks 171 files	Blocker 2.888 Critical 839 Major 3.878 Minor 4.944 Info 0
Complexity 3,1 /method 21,8 /class 38,3 /file Total: 22.644	Unit tests coverage 0,0% 0,0% line coverage	

Events All

22/04/2013	Version	1.0
13/12/2012	Profile	Sonar way version 1
13/12/2012	Profile	Sonar way version 1



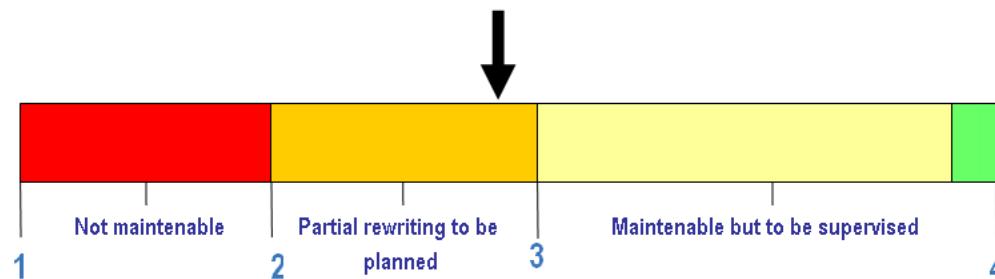
Test statique

Exemple d'analyse de code outillée

Summary	Maintenability	Transferability	Evolutivity	Robustness	Performance	Security
---------	----------------	-----------------	-------------	------------	-------------	----------

Technical Quality Index 2.89

Global CAST index: 2.89



Determines the cost and how the application is easily/difficultly maintainable.

Below a grade of 3, the number of exchanges between the developers and tests is multiplied by 3.



Test statique

Exemple d'analyse de code outillée

Summary	Maintainability	Transferability	Evolutivity	Robustness	Performance	Security
Metric grade 2.31						
Name Dead code (static)						
Description Respect of code coverage practices						
Child Metric Weight ▾	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade		
6	No	Java: Avoid unreferenced Interfaces	Moderate Risk	3.94		
4	No	JavaScript: Avoid unreferenced JavaScript Functions	Very High Risk	1		
4	No	Java: Avoid unreferenced Classes	Very High Risk	1.89		
4	No	Java: Avoid unreferenced Methods	High Risk	2.41		
4	No	Java: Avoid unreferenced Fields	High Risk	2.7		
4	No	PL/SQL: Avoid unreferenced Functions / Procedures	Very High Risk	1.31		
1	No	PL/SQL: Avoid unreferenced Tables	High Risk	2.38		
1	No	JSP: Avoid unreferenced JSPs	Very High Risk	1.22		
1	No	PL/SQL: Avoid unreferenced views	Very High Risk	1		

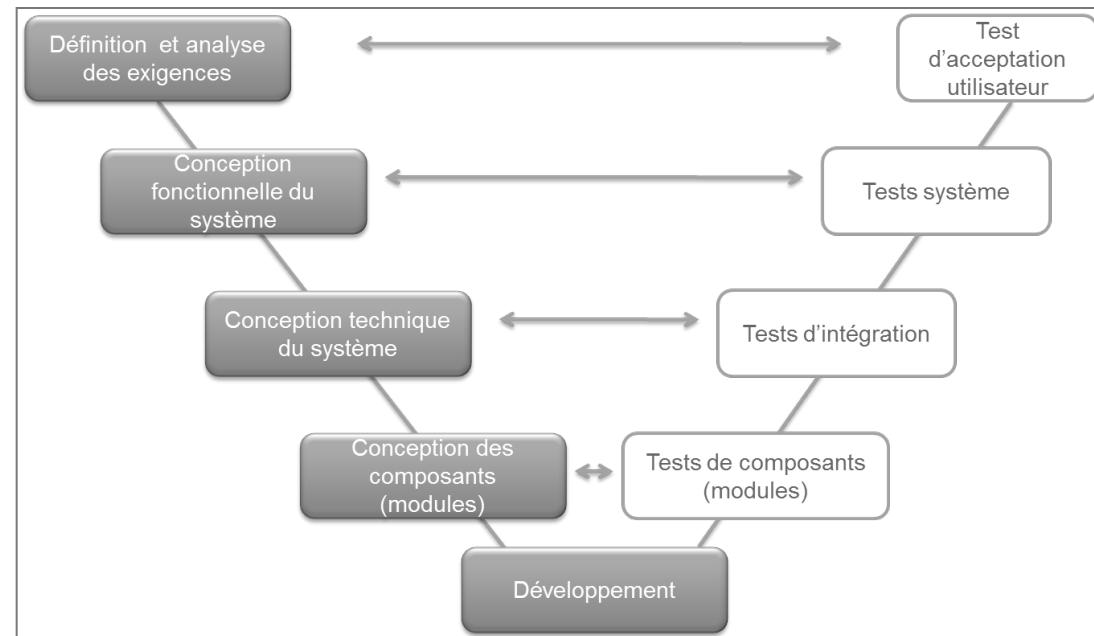
The dead code makes it harder to understand how the application really works
and makes its maintaining more expansive

Test dynamique - niveaux



Repose sur l'exécution du programme à tester

- 4 niveaux complémentaires
 - Test de composants (unitaire)
 - Test d'intégration des composants
 - Test du système global
 - Test d'acceptation (recette)





Test dynamique - techniques

Deux techniques :

- **Test structurel**
Jeu de test sélectionné en s'appuyant sur une analyse du code source du logiciel (*test boîte blanche / boîte de verre*)
- **Test fonctionnel**
Jeu de test sélectionné en s'appuyant sur les spécifications (*test boîte noire*)

En résumé, les méthodes de test dynamique consistent à :

- Exécuter le programme sur un ensemble fini de données d'entrées
- Contrôler la correction des valeurs de sortie en fonction de ce qui est attendu



Test structurel (white box)

- Les données de test sont produites à partir d'une analyse du code source

Critères de test :

- tous les chemins,
- toutes les instructions,
- etc...

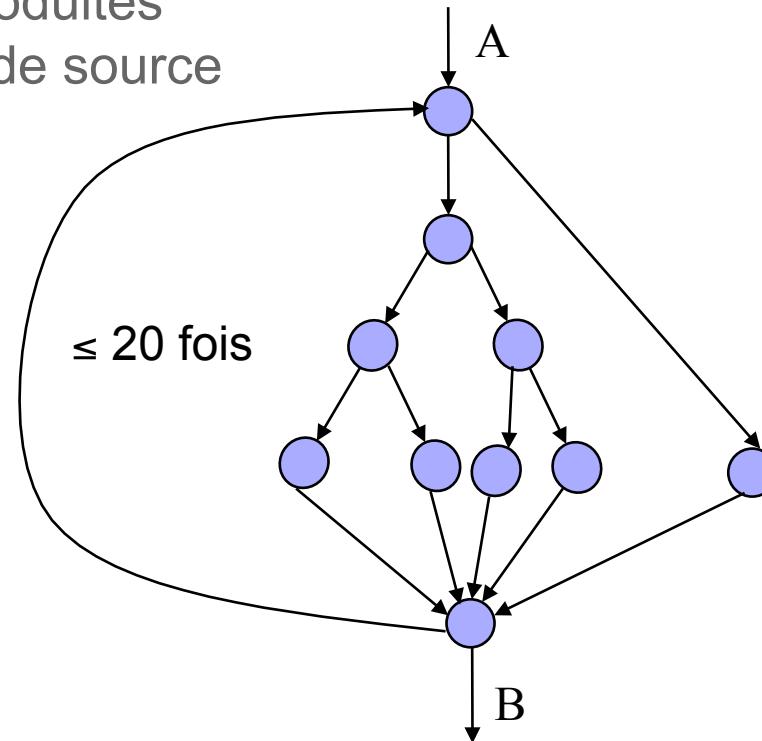
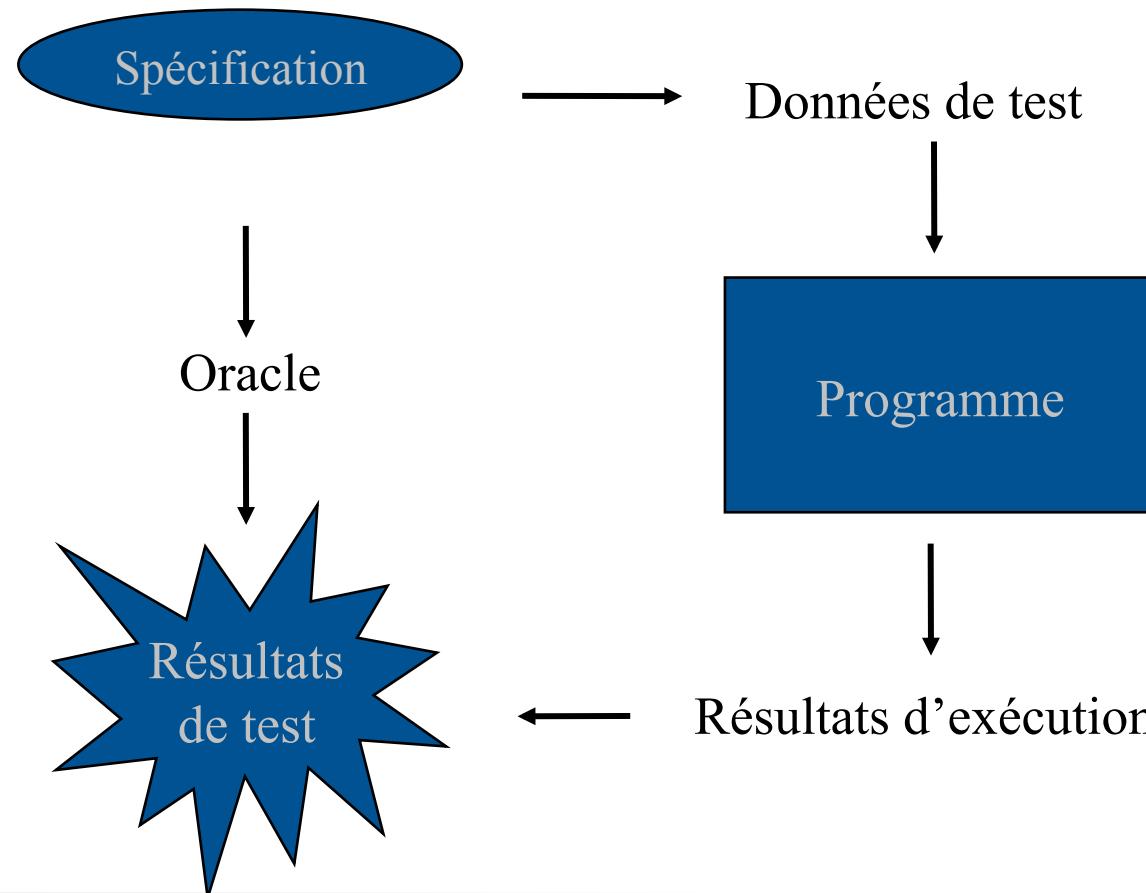


Fig. 1 : Flot de contrôle d'un petit programme



Test fonctionnel (black-box)

- Test de conformité par rapport à la spécification



Complémentarité (1) test fonctionnel / structurel



- Les 2 approches sont utilisées de façon complémentaire
- Exemple : soit le programme suivant, censé calculer la somme de 2 entiers :

```
function sum (x,y : integer) : integer;
  if (x = 600) and (y = 500)
    then
      sum := x-y
    else
      sum := x+y;
  end
```
- Une approche fonctionnelle détectera difficilement le défaut, alors qu'une approche par analyse de code pourra produire la donnée de test :
 $x = 600, y = 500.$

Complémentarité (2) test fonctionnel / structurel



- En examinant ce qui a été réalisé, on ne prend pas forcément en compte ce qui aurait du être fait :
 - Les approches structurelles détectent plus facilement les erreurs commises dans le programme
 - Les approches fonctionnelles détectent plus facilement les erreurs d'omission et de spécification
- Une difficulté du test structurel consiste dans la définition de l'oracle de test.

Test de logiciels – auto-évaluation

L'exemple du triangle



- Soit la spécification suivante :

Un programme prend en entrée trois entiers. Ces trois entiers sont interprétés comme représentant les longueurs des cotés d'un triangle. Le programme rend un résultat précisant s'il s'agit d'un triangle scalène, isocèle ou équilatéral (ou une erreur si les données ne correspondent pas aux longueurs d'un triangle).

G. Myers (The Art of Software testing - 1979)

- Donnez un ensemble de cas de test que vous pensez adéquat pour tester pour ce programme...

Test de logiciels – auto-évaluation

L'exemple du triangle



- Avez-vous un cas de test pour :

Test de logiciels – auto-évaluation

L'exemple du triangle



- Chacun de ces cas correspond à un défaut constaté dans des implantations de cet exemple du triangle
- La moyenne des résultats obtenus par un ensemble de développeurs expérimentés est de 7.8 sur 14.

=> Le test est une activité complexe, a fortiori sur de grandes applications

Test dynamique – 4 activités

Sélection d'un jeu de tests : choisir un sous-ensemble des entrées possibles du logiciel

Soumission du jeu de tests

Dépouillement des résultats : consiste à décider du succès ou de l'échec du jeu de test (*verdict*):

- Fail, Pass, Inconclusive

Évaluation de la qualité et de la pertinence des tests effectués (déterminant pour la décision d'arrêt de la phase de test)

Types de tests (1) (renseignent la nature du test mené)



Tests fonctionnels

Valide les résultats rendus par les services

Tests non-fonctionnels

Valide la manière dont les services sont rendus

Tests nominaux / de bon fonctionnement

Vérifie que le résultat calculé est le résultat attendu, en entrant des données valides au programme (*test-to-pass*)

Tests de robustesse

Vérifie que le programme réagit correctement à une utilisation non conforme, en entrant des données invalides (*test-to-fail*)

Types de tests (2) (renseignent la nature du test mené)



Test de performance

- Load testing (test avec montée en charge)
- Stress testing (soumis à des demandes de ressources anormales)

Test de non-régression

Vérifie que les corrections ou évolutions dans le code n'ont pas créé d'anomalies nouvelles

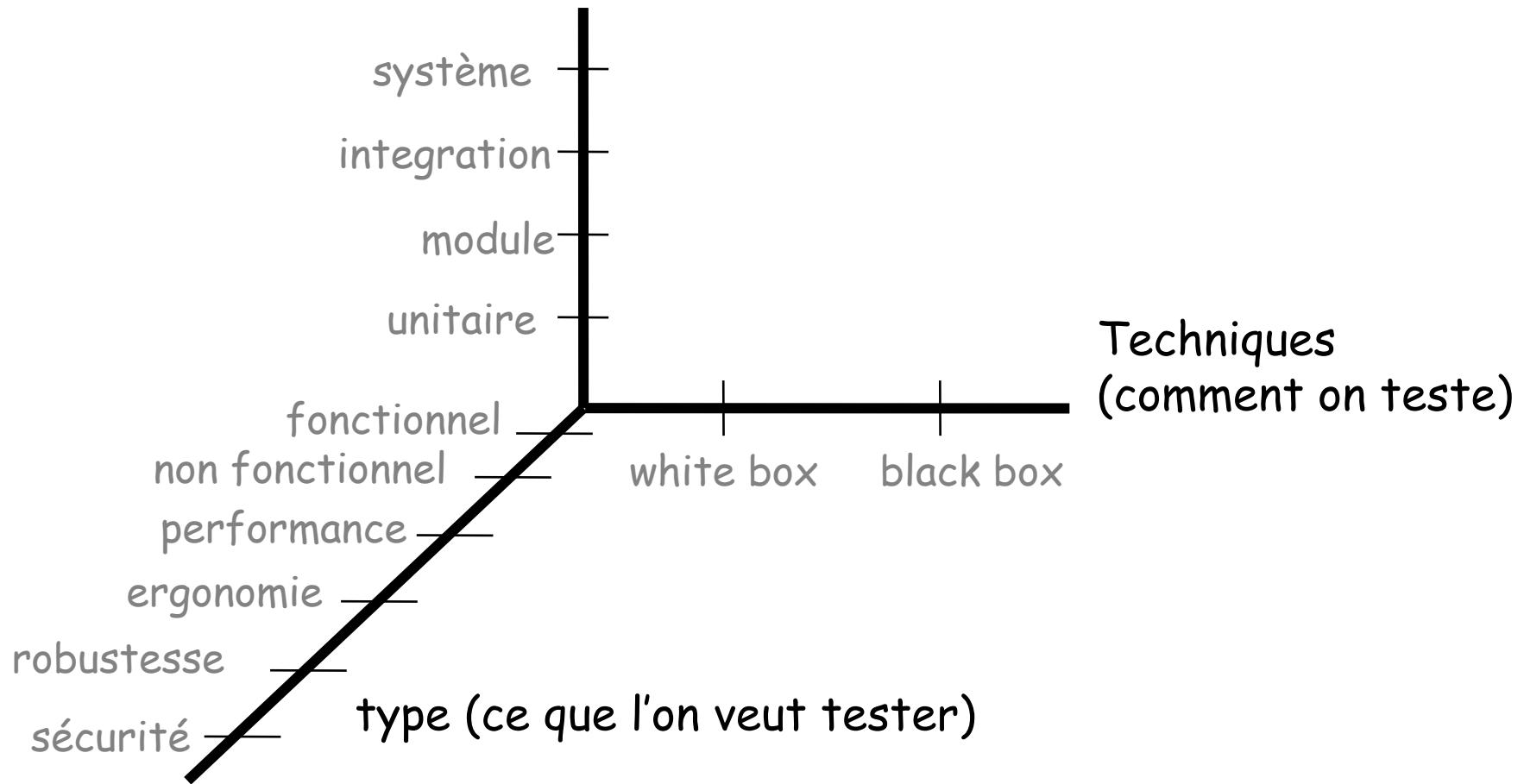
Test de confirmation

Valide la correction d'un défaut

Catégories de tests

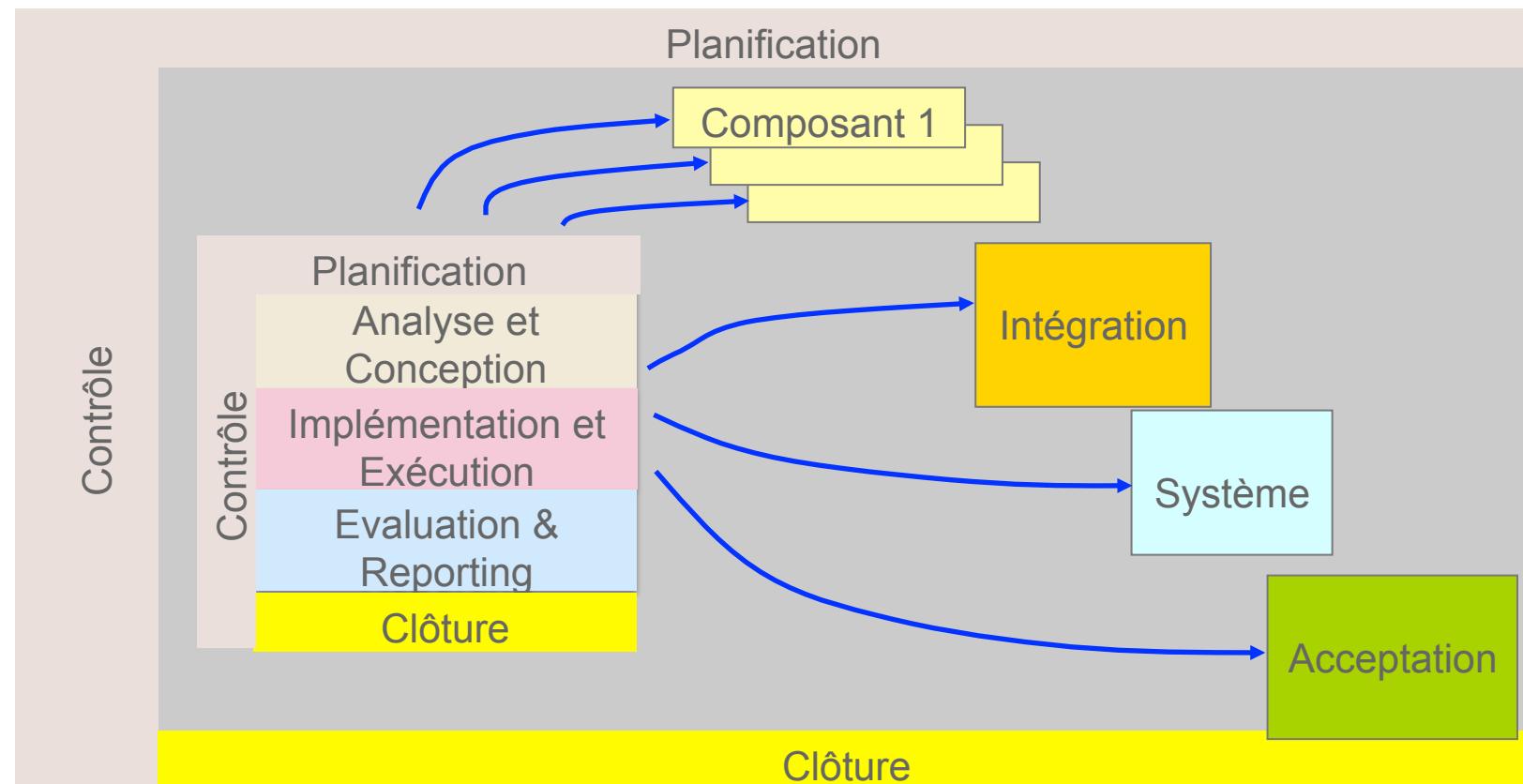


Niveau de détail (situation dans le cycle de vie)

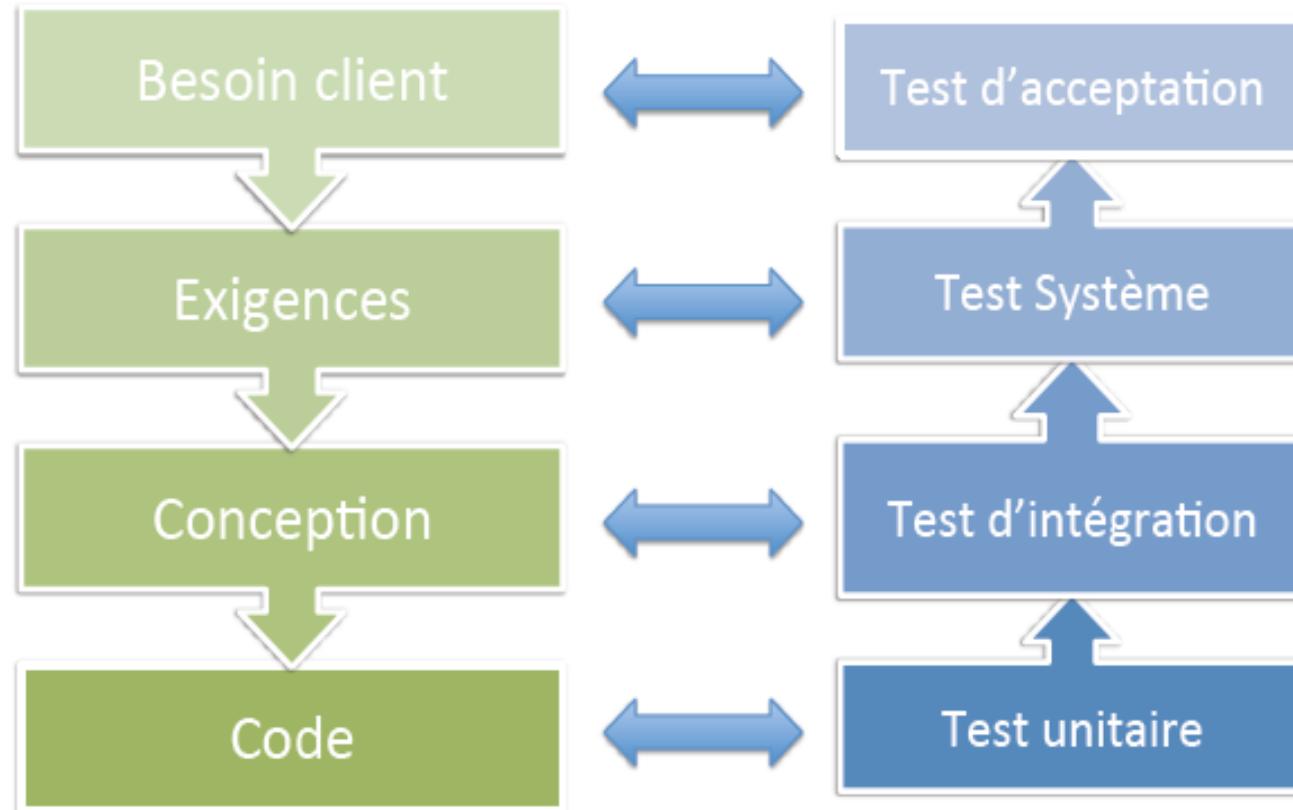


Processus de test et stratégie de test

Le Test est un ensemble d'activités à dérouler à différents niveaux



Développement et niveaux de tests



Niveaux de tests (renseignent l'objet du test)



Tests (structurels) unitaires

Test de procédures, de modules, de composants
(coût : 50% du coût total du développement initial correspondant)

Tests d'intégration

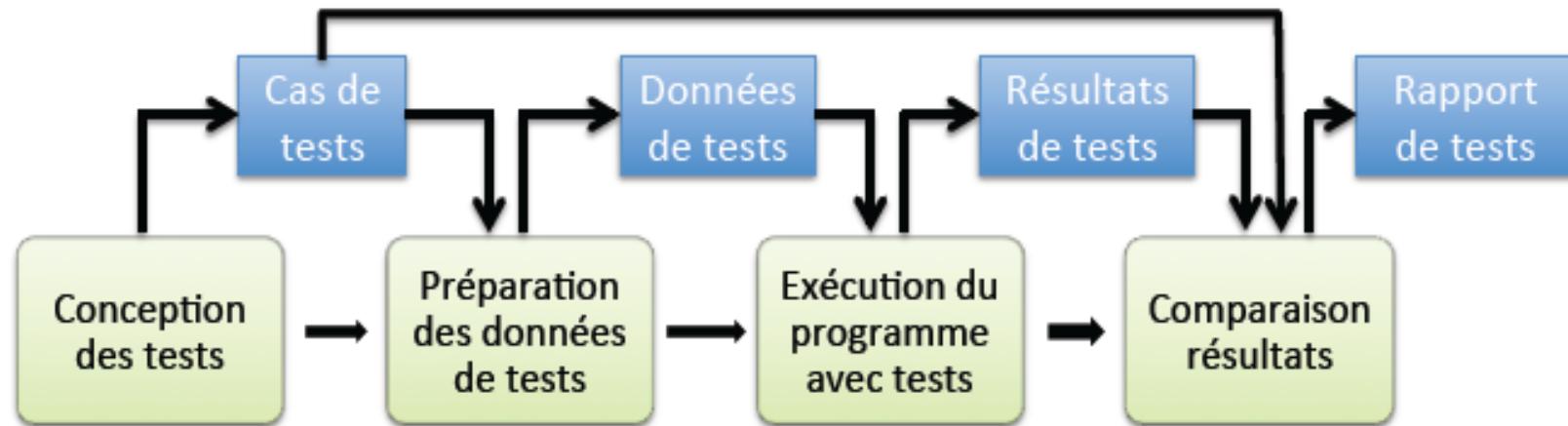
Test de bon comportement lors de la composition des procédures,
modules ou composants
(coût d'un bug dans cette phase : 10 fois celui d'un bug unitaire)

Tests système / de recette

Validation de l'adéquation aux spécifications
(coût d'un bug dans cette phase : 100 fois celui d'un bug unitaire)

Processus de test et stratégie de test

Test et cycle de vie du processus de test





Difficultés du test

- Le test exhaustif est en général impossible à réaliser
 - En test structurel, le parcours du graphe de flot de contrôle conduit à une forte explosion combinatoire

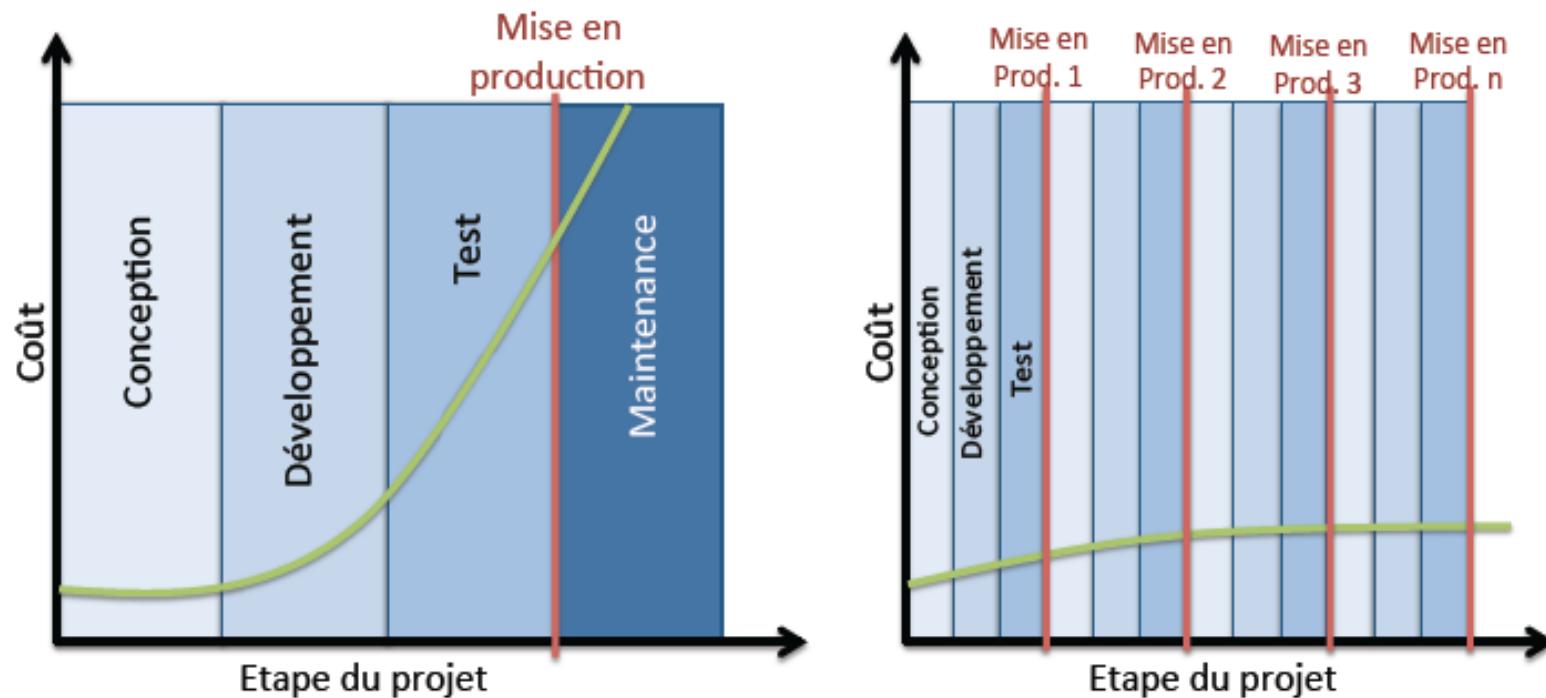
Exemple : le nombre de chemin logique dans le graphe de la figure 1 est supérieur à $10^{14} \approx 5^{20} + 5^{19} + \dots + 5^1$
 - En test fonctionnel, l'ensemble des données d'entrée est en général infini ou très grande taille

Exemple : un logiciel avec 5 entrées analogiques sur 8 bits admet 2^{40} valeurs différentes en entrée
- => le test est une méthode de validation partielle de logiciels
=> la qualité du test dépend de la pertinence du choix des données de test

Coût d'un bug



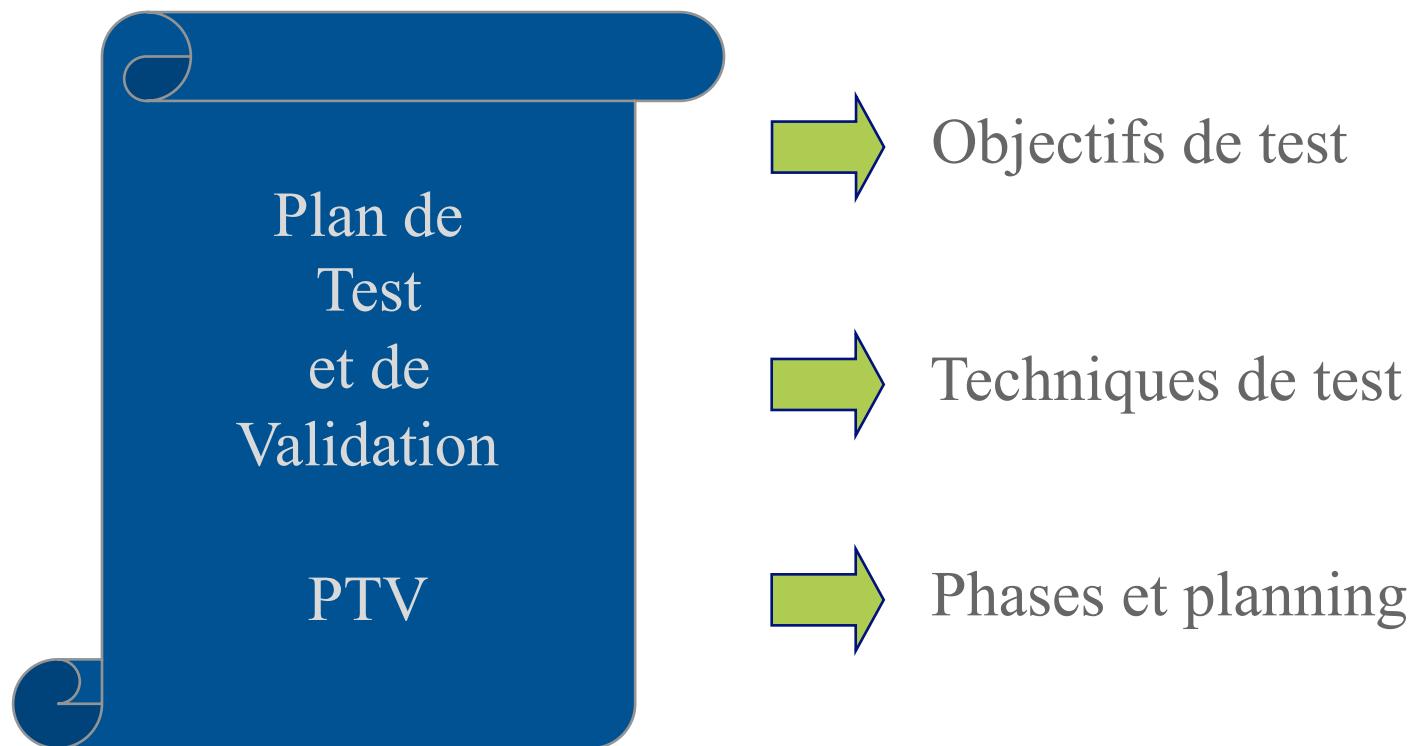
Le but est de les détecter le plus tôt possible (Stratégie)





Stratégie de test

En début de projet, définition d'un Plan de Test et de Validation (PTV)





Acteurs du test

- Deux situations :
 - Je teste un programme que j'ai écrit
 - Je teste un programme que quelqu'un d'autre a écrit
- Trois questions :
 - Comment choisir la technique de test ?
=> **boite blanche** ou **boite noire** ?
 - Comment obtenir le résultat attendu ?
=> problème de **l'oracle** du test
 - Comment savoir quand arrêter la phase de test ?
=> problème de **l'arrêt**

Outils et Méthodes intervenants à différentes étapes

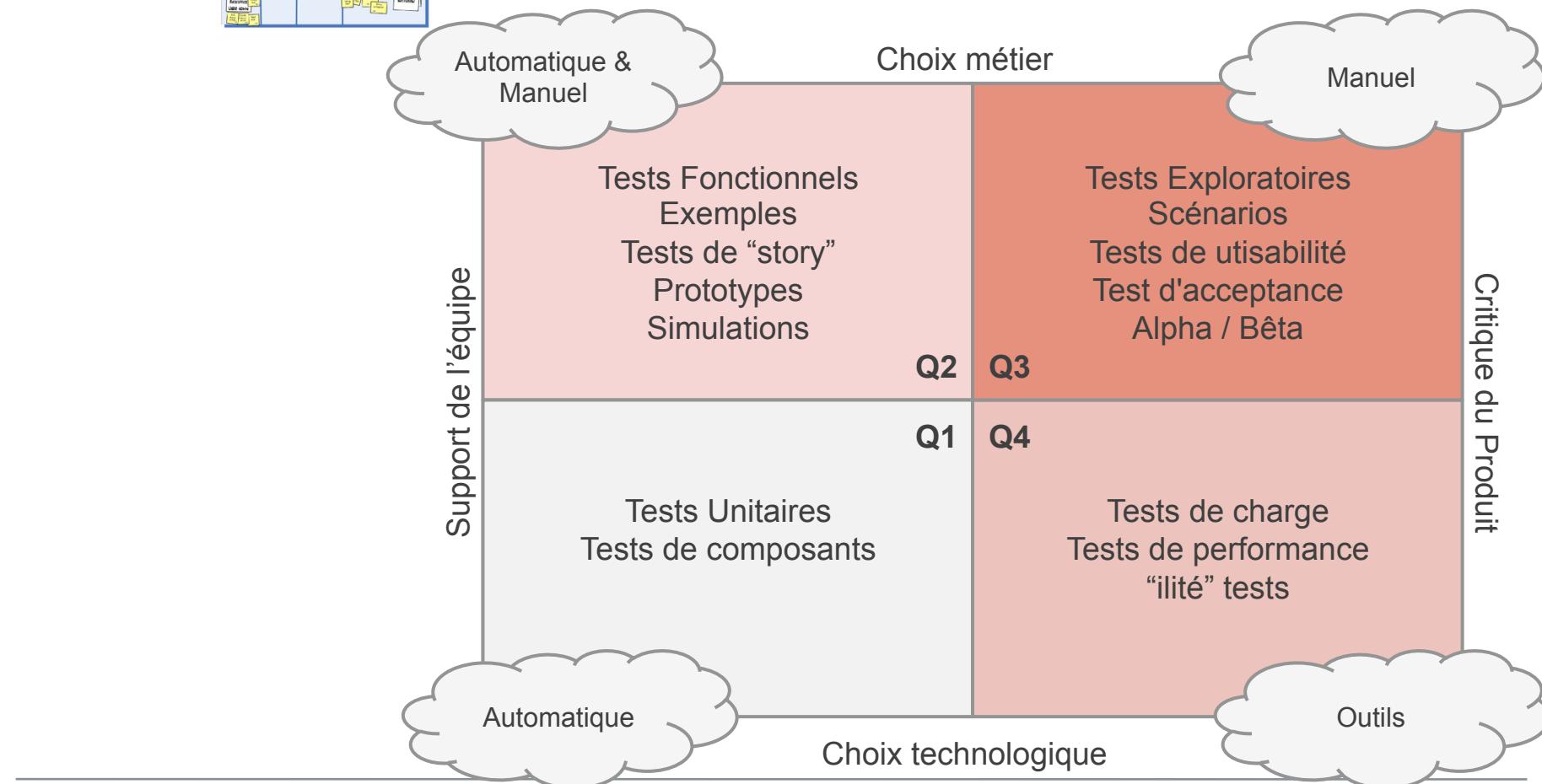


mingle®

JIRA Agile

ScrumDo

hp Agile Manager



Outils et Méthodes intervenants à différentes étapes



Exemple Mingle pour les projets Agile

Click n Dev PB 9. Storyboard - Mingle - Firefox for-e-Buro

Bécher Édition Affichage Historique Marque-pages Quits 2

http://10.127.111.51/projects/didc_n_dev_cards/grid?aggregate_property=complexité&aggregate_type=sum&color_by=complexité&filters[]=(type%3D%22UserStory%22)&filters[]=(status%3D%22In Progress%22)

Les plus visités Google

coopnet - récapitulatif Click n Dev PB 9. Storyboard - Mi... □

Rank cards

Filter Advanced Filter

Show cards where:

Type is Story

Planning - is Sprint (Sprint en cours) and they are tagged with Add a filter

Add tags

Color legend

Complexité

- 1
- 2
- 3
- 5
- 6
- 13

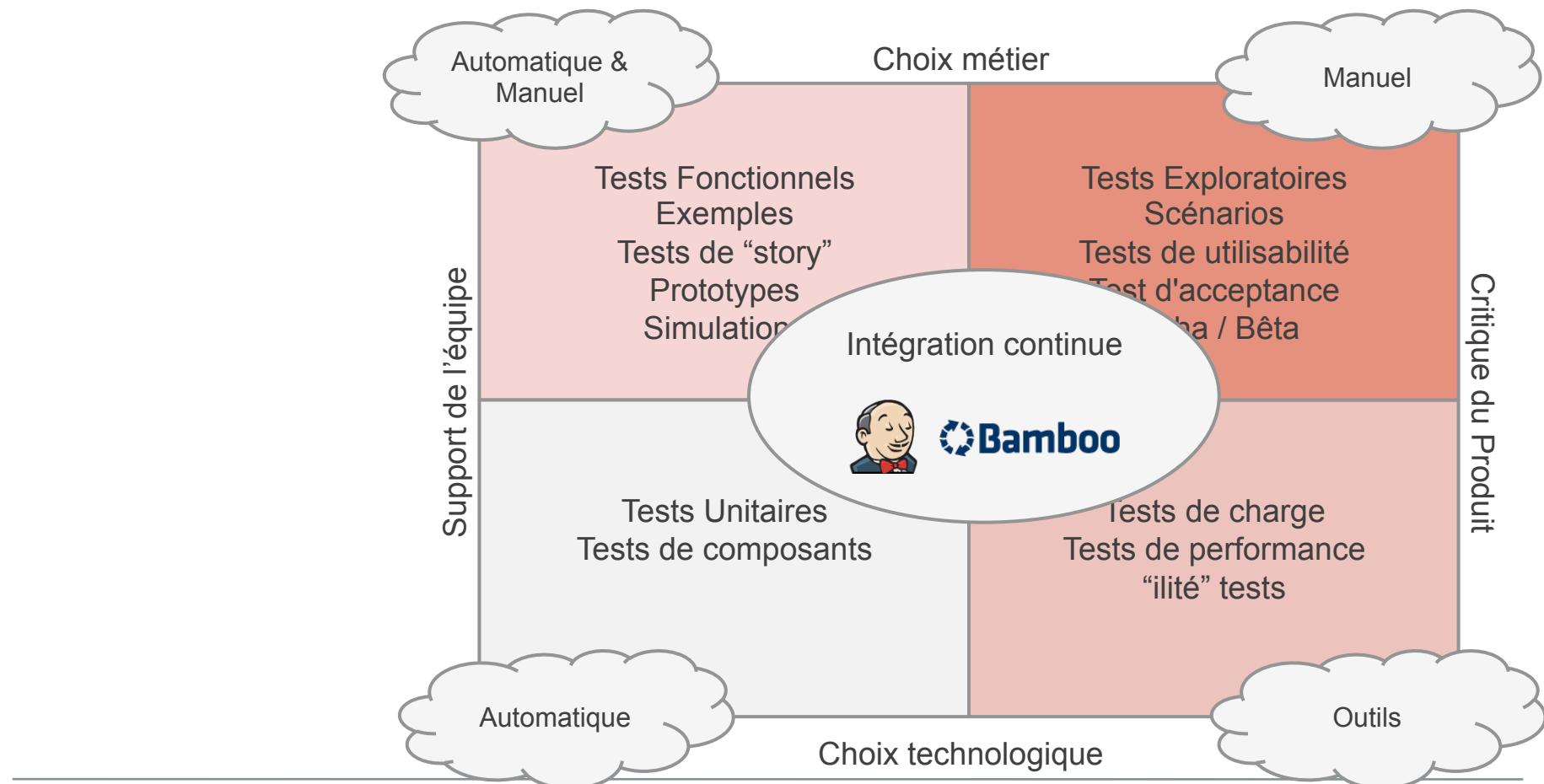
Import / Export

	(not set) (2)	A faire (17)	En cours (55)	A tester (10)	A valider par PO (0)	Terminé (0)
Dev CnD	<p>[OCAWA] #799 Interface avec Ocawa</p> <p>#1131 [EvalOrbis] Ergonomie - instruments</p>	<p>[GC] #562 Consultation des historiques.</p> <p>[GC] #1029 Ordre les utilisateurs d'une</p> <p>#1090 [ANOS] Correction des sons.</p> <p>[GC] #1020 Génération des options sur conférence.</p> <p>[Ergo] #1096 Retourne page d'accueil V2</p> <p>[Ergo] #1111 Page socle V2 - Notifications.</p> <p>[Ergo] #1112 Page socle V2 - Notifications.</p> <p>[Ergo] #1113 Page socle V2 - Notifications.</p> <p>[Ergo] #1114 Page socle V2 - Notifications.</p> <p>[Ergo] #1115 Page socle V2 - Notifications.</p>	<p>[GDA] #1019 Génération des actions sur notifications.</p> <p>[GC] #1075 Ajout d'une enveloppe GC à un envoi.</p> <p>#1116 001R00022</p>			
Dev WSOC	<p>[WSOC] #1078 Ajouter la possibilité de filtre les</p> <p>[WSOC] #1091 Modifier l'entité de financement</p> <p>[WSOC] #1144 L'opération de réactivation</p>	<p>#1035 [WSOC] Etudier les possibilités de</p>	<p>#1037 [WSOC] Génération des index.</p> <p>#1092 [WSOC] Génération des annulations.</p> <p>#1094 [WSOC] Recréation</p>		<p>[WSOC] #1073 Répartir les enveloppes par ville</p>	
Terminé						

Outils et Méthodes intervenants à différentes étapes



Build et Distribution: Utile pour tout le quadrant de test



Outils et Méthodes intervenants à différentes étapes



Intégration continue : Tableau de contrôle

The Jenkins interface displays the following information:

- Build Queue:** Shows several builds in progress, including "Building plugin-compat-tester #5380" and "Building plugin-sladiator #2".
- Project Status:** A grid showing the last success, last failure, and last duration for various projects like "infra_plugins_svn_to_dj", "infra_syntac", etc.
- Checkstyle Warnings - New Warnings:** A detailed view of new warnings, showing a summary table and a distribution table for source folders.

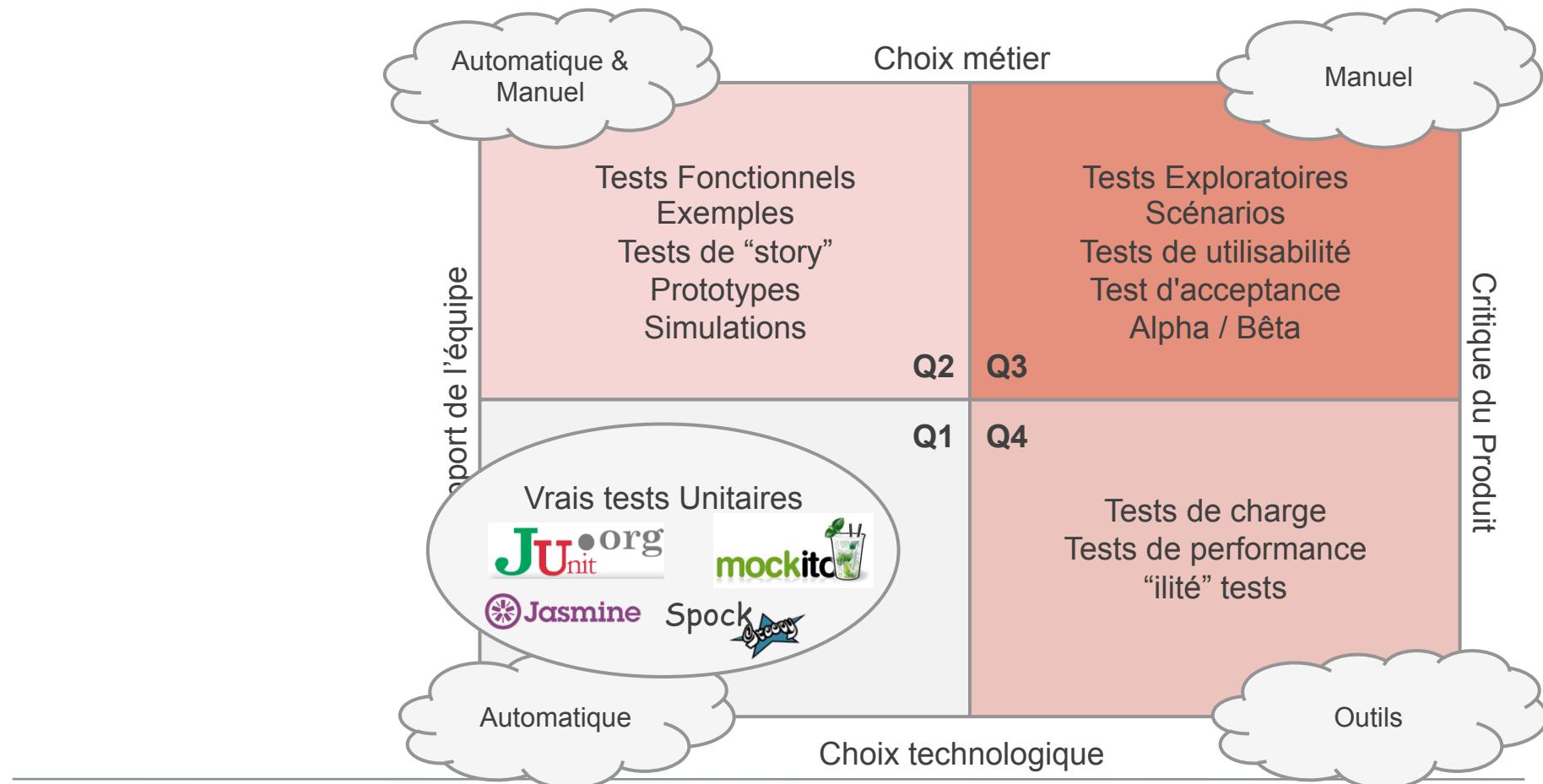
Permet d'avoir en temps réel le statut du projet (tests unitaires, tests statiques, couverture...)



Outils et Méthodes intervenants à différentes étapes



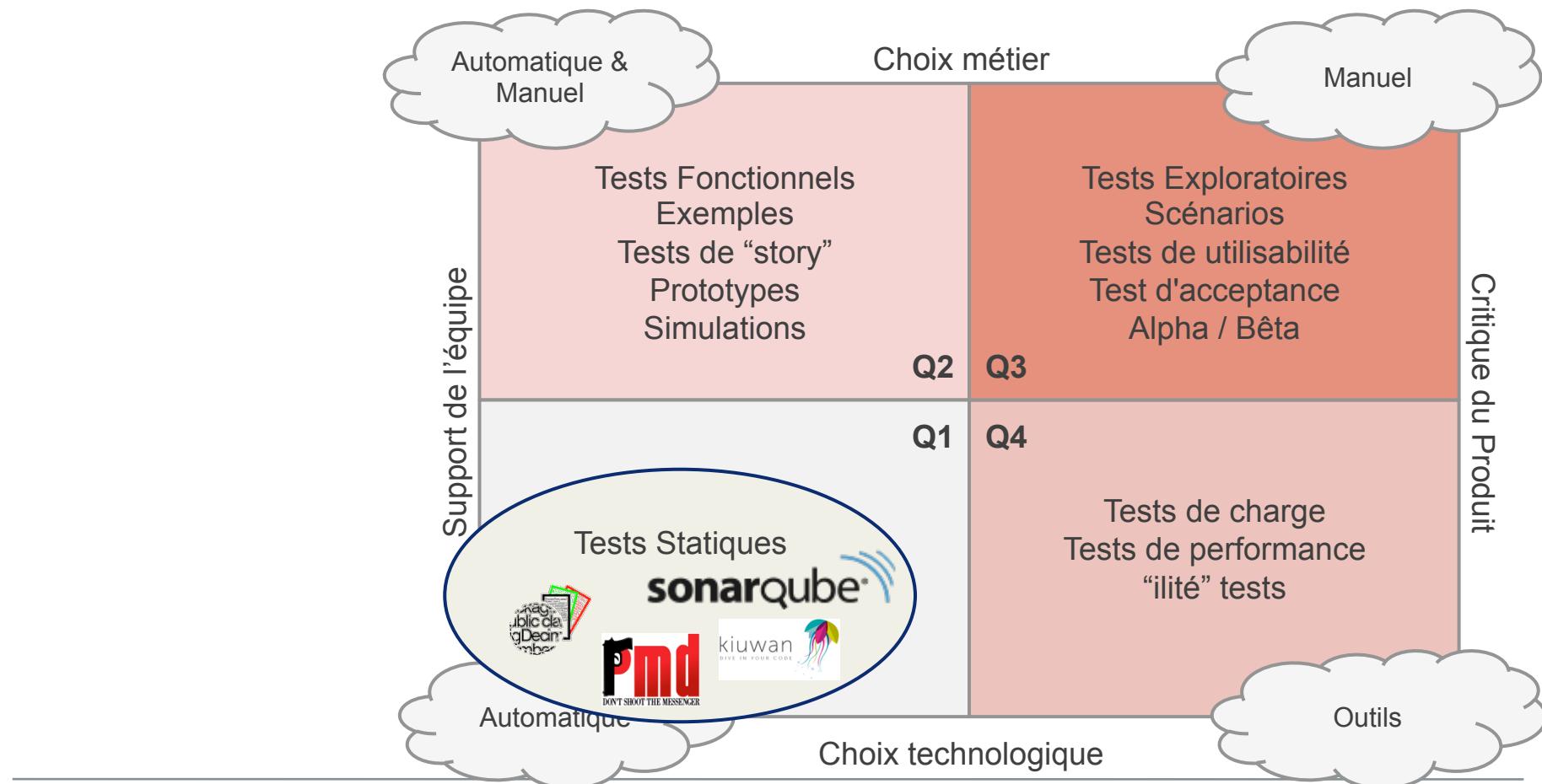
Tests unitaires



Outils et Méthodes intervenants à différentes étapes



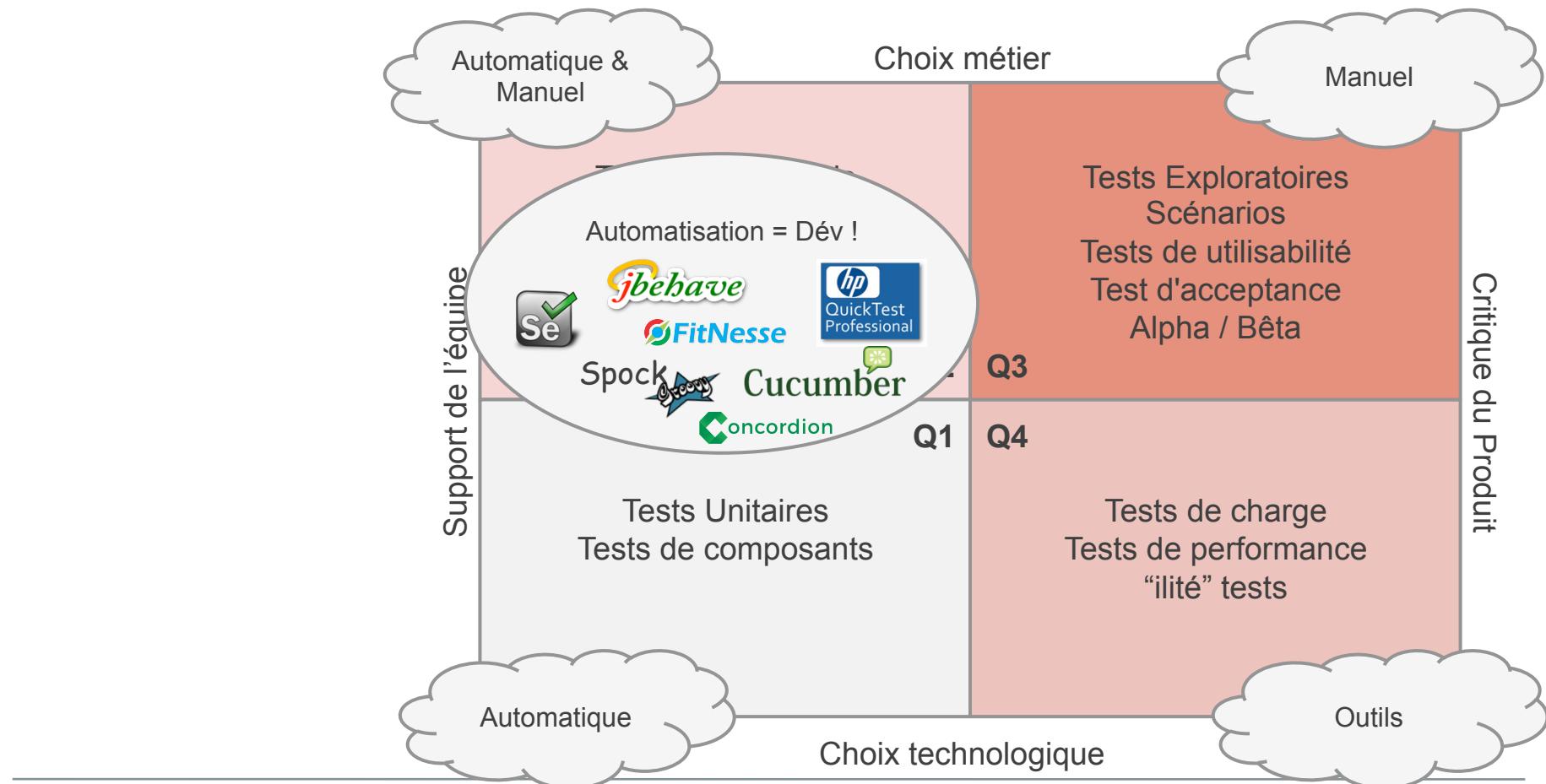
Tests statiques !



Outils et Méthodes intervenants à différentes étapes



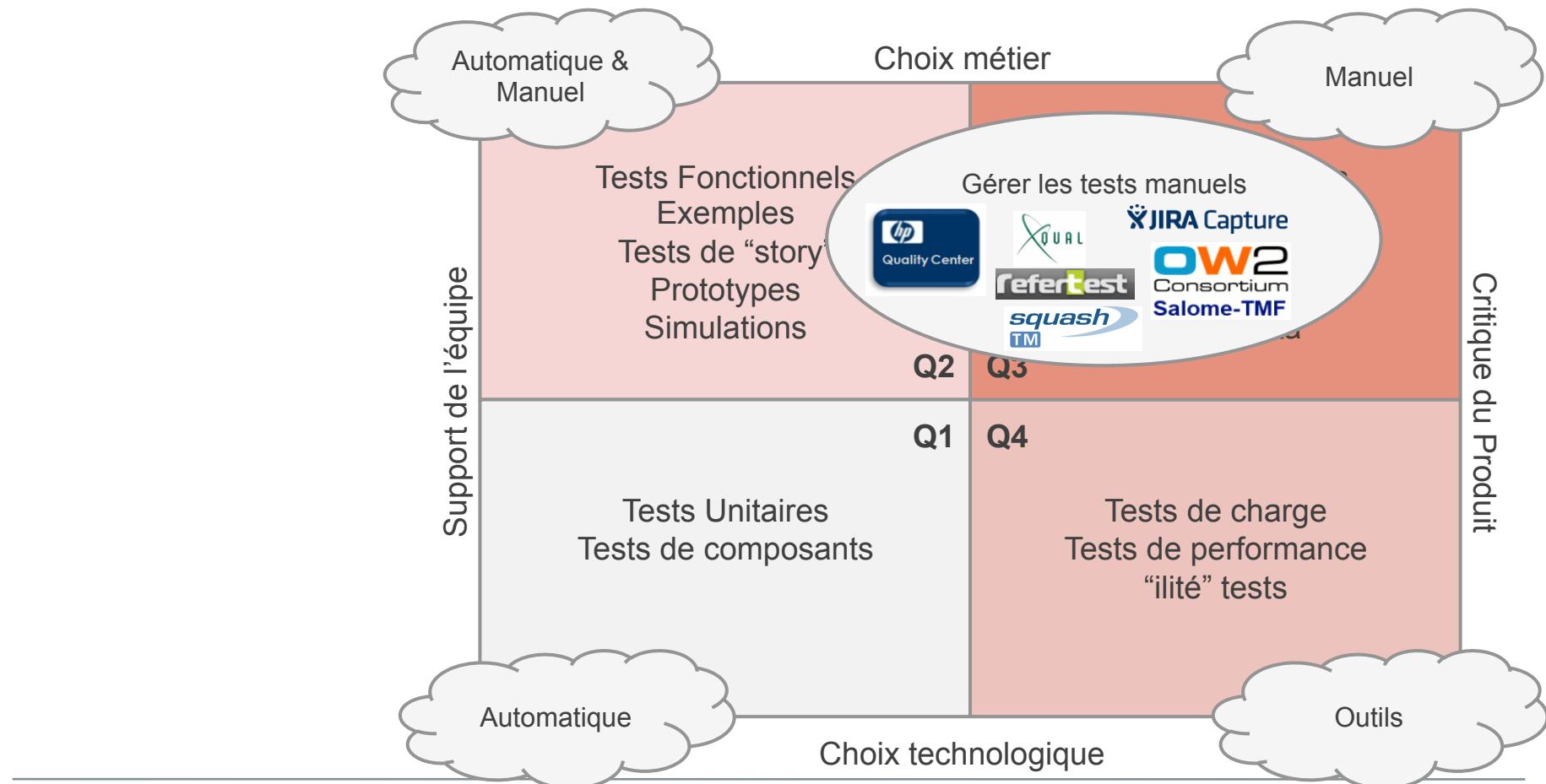
Automatisation des tests



Outils et Méthodes intervenants à différentes étapes



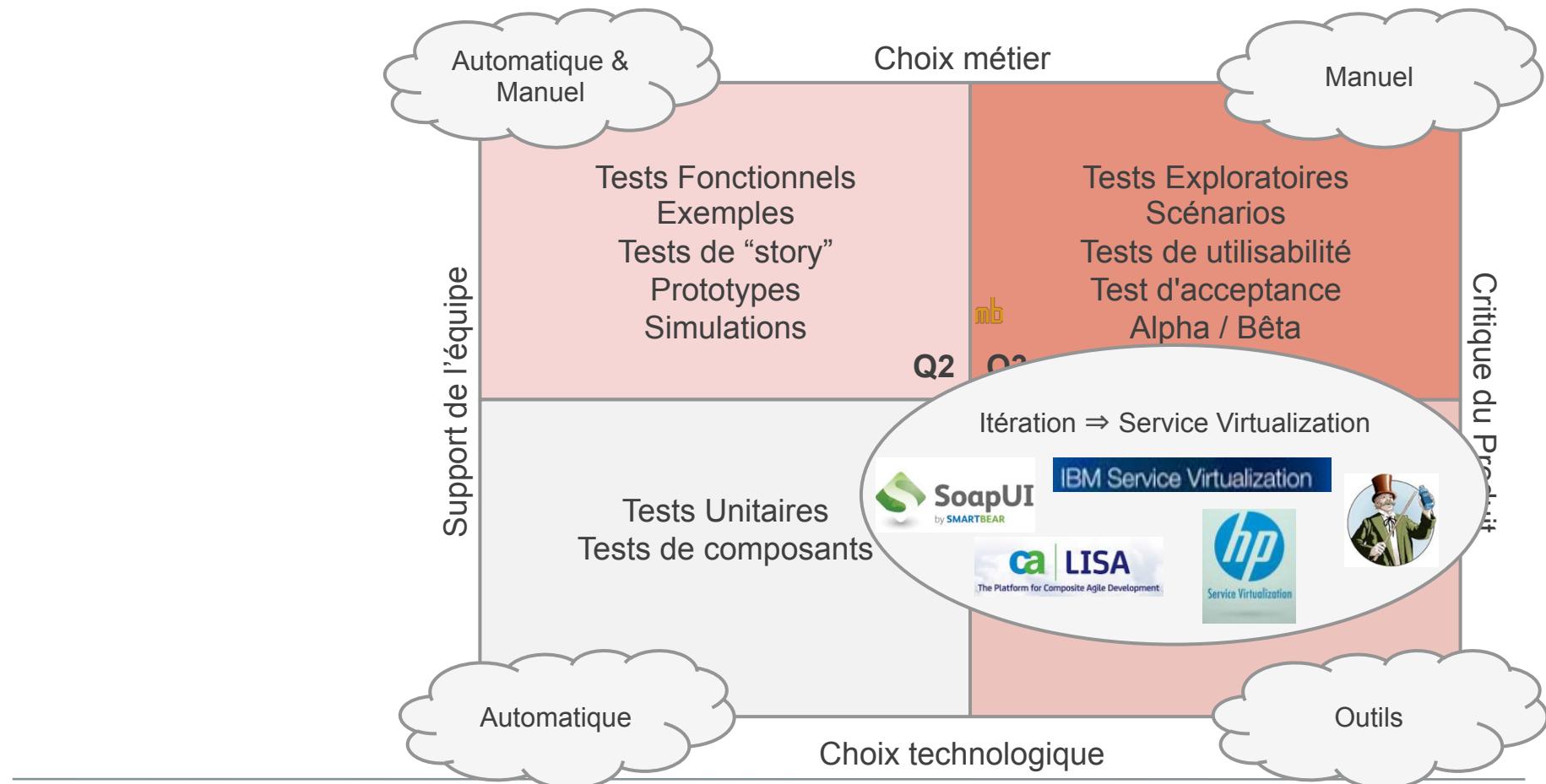
Gestion des tests y compris exploratoires !



Outils et Méthodes intervenants à différentes étapes



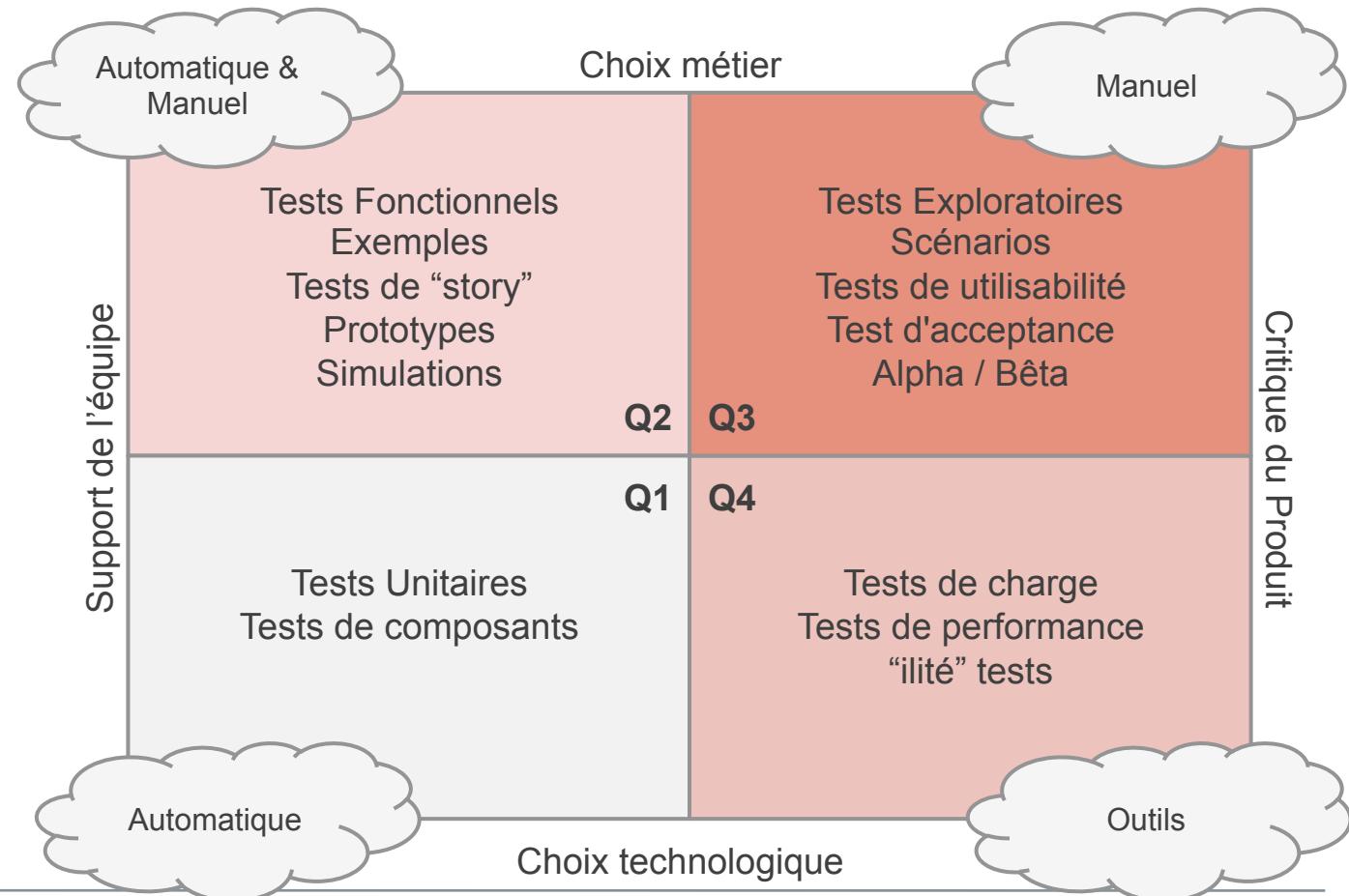
Virtualisation et Cloud : un nouveau défit !



Outils et Méthodes intervenants à différentes étapes



Couverture : un métrique important



Outils et Méthodes intervenants à différentes étapes



BDD avec JBehave

1. Write story

Plain
text

Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status should be OFF
When stock is traded at 16.0
Then the alert status should be ON

2. Map steps to Java

POJO

```
public class TraderSteps {  
    private TradingService service; // Injected  
    private Stock stock; // Created  
  
    @Given("a stock and a threshold of $threshold")  
    public void aStock(double threshold) {  
        stock = service.newStock("STK", threshold);  
    }  
    @When("the stock is traded at price $price")  
    public void theStockIsTraded(double price) {  
        stock.tradeAt(price);  
    }  
    @Then("the alert status is $status")  
    public void theAlertStatusIs(String status) {  
        assertThat(stock.getStatus().name(), equalTo(status));  
    }  
}
```

Outils et Méthodes intervenants à différentes étapes



Tester avec FitNesse

- FitNesse utilise un wiki pour définir les cas de tests : on écrit des tables de décision. FitNesse est bien adapté à des testeurs fonctionnels
- Le développeur doit écrire le “fixture” code pour appeler les fonctionnalités concernées

eg.triviaGameExample.fitnesseFixtures.AddRemovePlayerFixture		
playerName	addPlayer?	countPlayers?
AI	true	1
Bertha	true	2

```
import fit.ColumnFixture;

public class AddRemovePlayerFixture {
    private String playerName;
    private Game theGame;

    public void setPlayerName(String playerName) {
        this.playerName = playerName;
    }

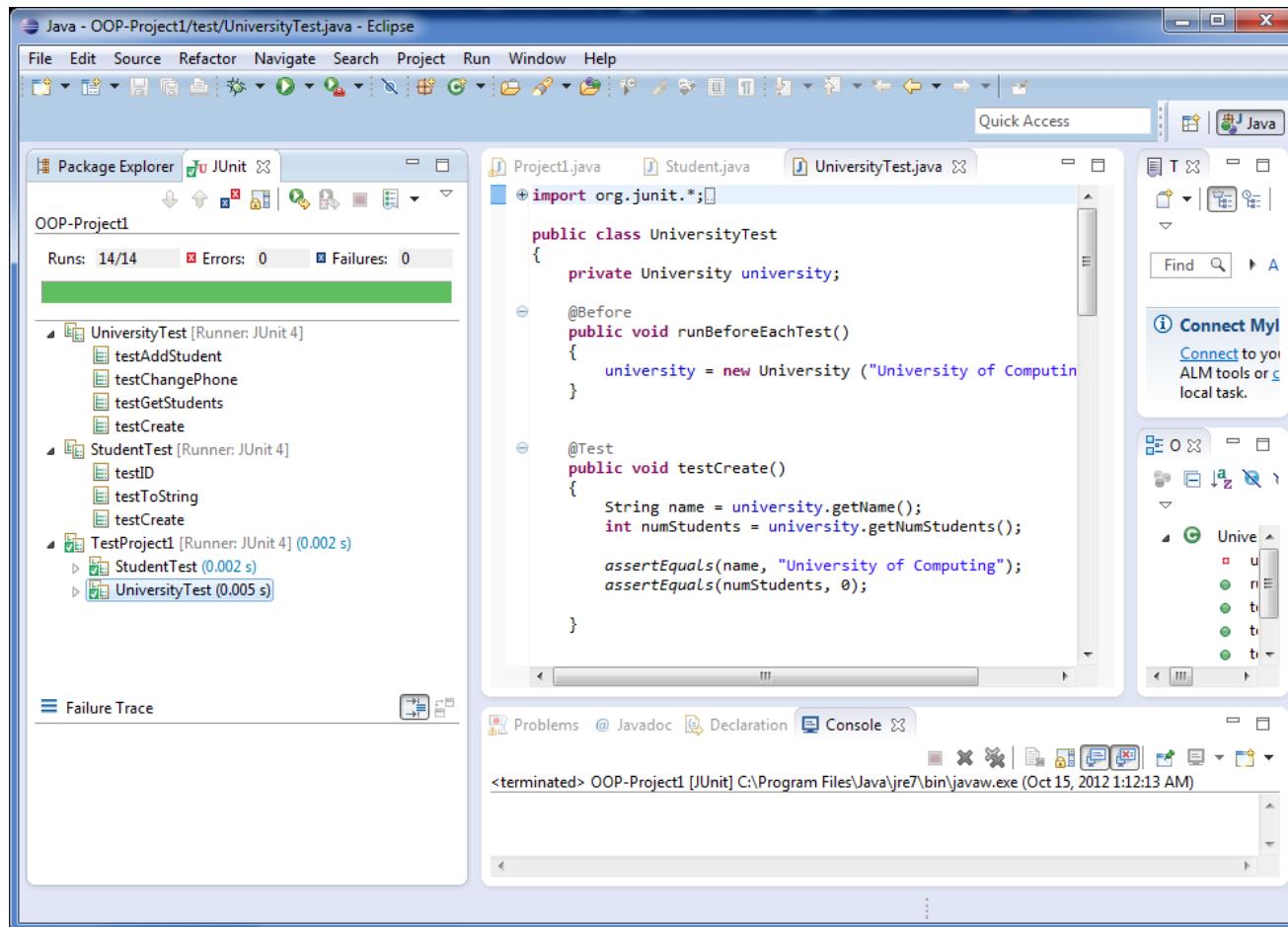
    public boolean addPlayer() {
        theGame = StaticGame.theGame;
        Player thePlayer = theGame.addPlayer(playerName);
        return theGame.playerIsPlaying(thePlayer);
    }

    public int countPlayers() {
        return theGame.getNumberOfPlayers();
    }
}
```

Outils et Méthodes intervenants à différentes étapes



Exemple: tests unitaires avec Junit depuis Eclipse



Outils et Méthodes intervenants à différentes étapes



Exemple: qualité du code mesurée avec Sonar

