



# Sensibilisation aux Tests de Projets Informatique - Managed Testing -

UM2 - Master 1      Année 2018 - 2019

Le 11 avril 2019

Thierry SINOT | Directeur Conseil Expert  
[thierry.sinot@cgi.com](mailto:thierry.sinot@cgi.com)

Unité d'enseignement (UE) : Conduite de  
projet  
Code UE : HMIN204



La force de l'engagement<sup>MD</sup>

# Introduction

## TOAST

- Thème

Manager les tests  
dans un Projet  
informatique

Sensibiliser les ingénieurs à l'importance des  
Tests Effectuer un rappel sur l'état de l'art et  
autres bonnes pratiques

Fixer les orientations pour les années à venir

- Objectif

- Animation

Présentations avec  
Exemples, Echanges Q/R

- Séquences &

- Timing

Intro Présentation CGI  
Rappel sur les cycles projet  
Typologie des tests – concepts et définitions  
Pourquoi des TU / Impact à ne pas en faire  
Les processus de TU : Préconisations :  
Les différentes stratégies de TU  
DevOps  
Les outils – présentation

# Repères

## Chiffres clés 2017

Fondée en 1976



Plus de 40 années  
de croissance rentable continue



7,5 Md€  
de CA en 2017\*



70 000  
professionnels



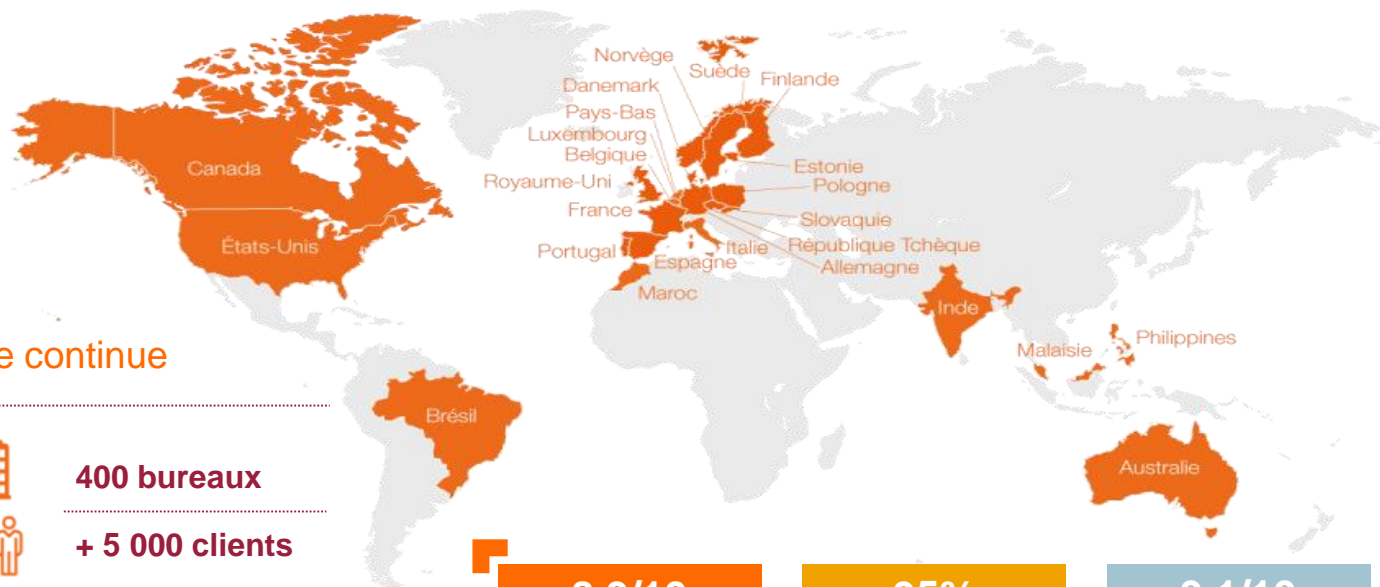
400 bureaux



+ 5 000 clients



Des capacités mondiales de prestation  
de services grâce à des centres situés  
sur 4 continents



8,9/10

NOTE  
DE SATISFACTION  
DES CLIENTS

95%

DES PROJETS  
RESPECTENT  
LES BUDGETS  
ET LE PLANNING

9,1/10

INDICE DE FIDÉLITÉ  
DES CLIENTS

\*Le Chiffre d'affaire en année fiscale CGI, c'est-à-dire  
d'Octobre 2016 à Septembre 2017

Certifiées ISO 9001, les **Assises de Gestion** garantissent l'excellence de  
nos processus opérationnels avec nos clients, membres et actionnaires.

# About CGI

## Key figures 2017



**Over 40 years**  
of long-term profitable growth

**Founded in 1976**



**€7,5 billion**  
revenue in 2017



**72 500**  
professionals



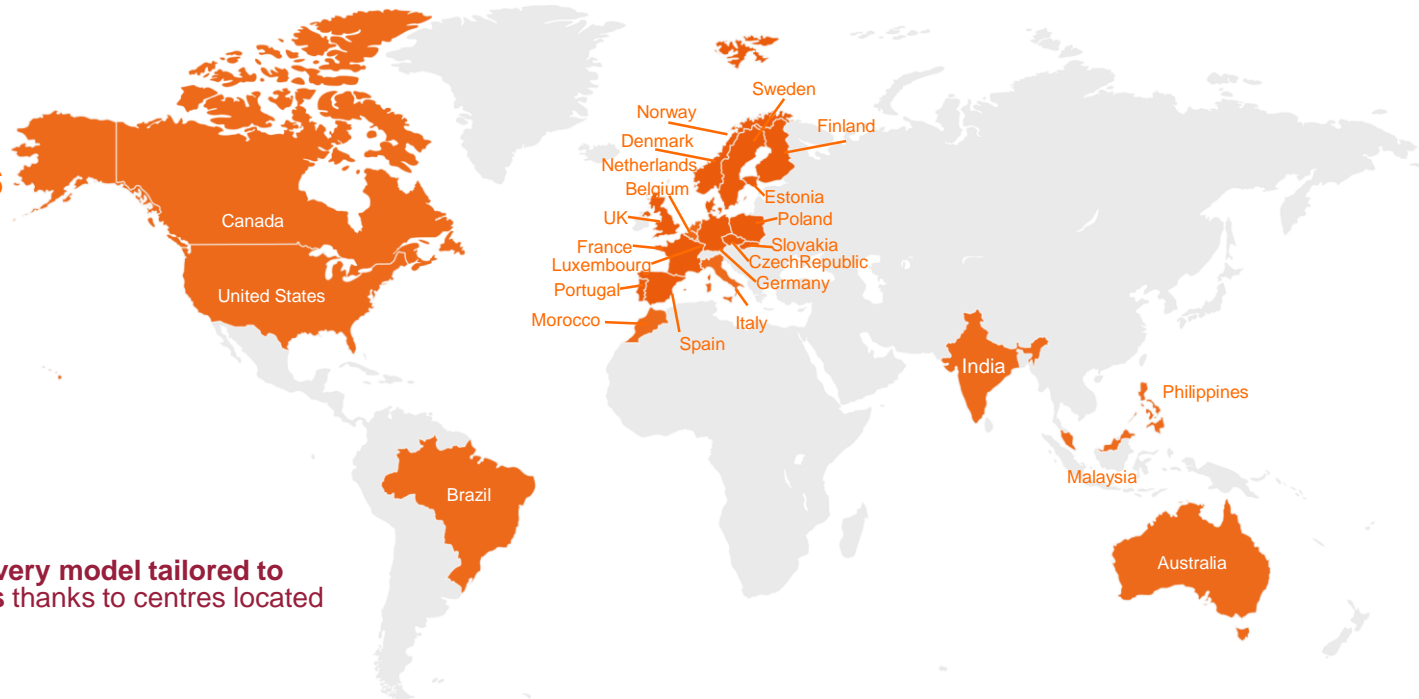
**400 offices**



**+ 5 000 clients**



**A best-fit global delivery model tailored to your business needs** thanks to centres located across 5 continents



# A leader in consulting and digital solutions in France, Luxembourg and Morocco



€1,1 billion  
revenue in 2017



+ 11,000  
professionals



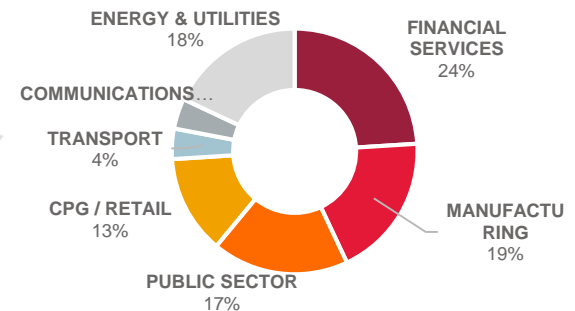
25 offices

- Offices
- Offices & on-shore production centres
- Near-shore production centres



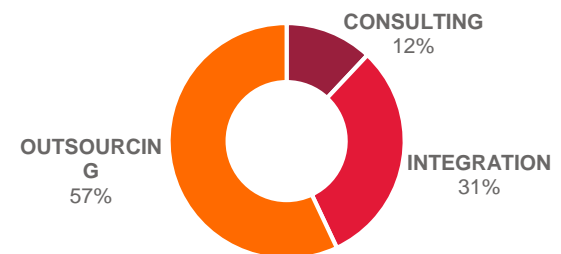
## Distribution of turnover

Per industry

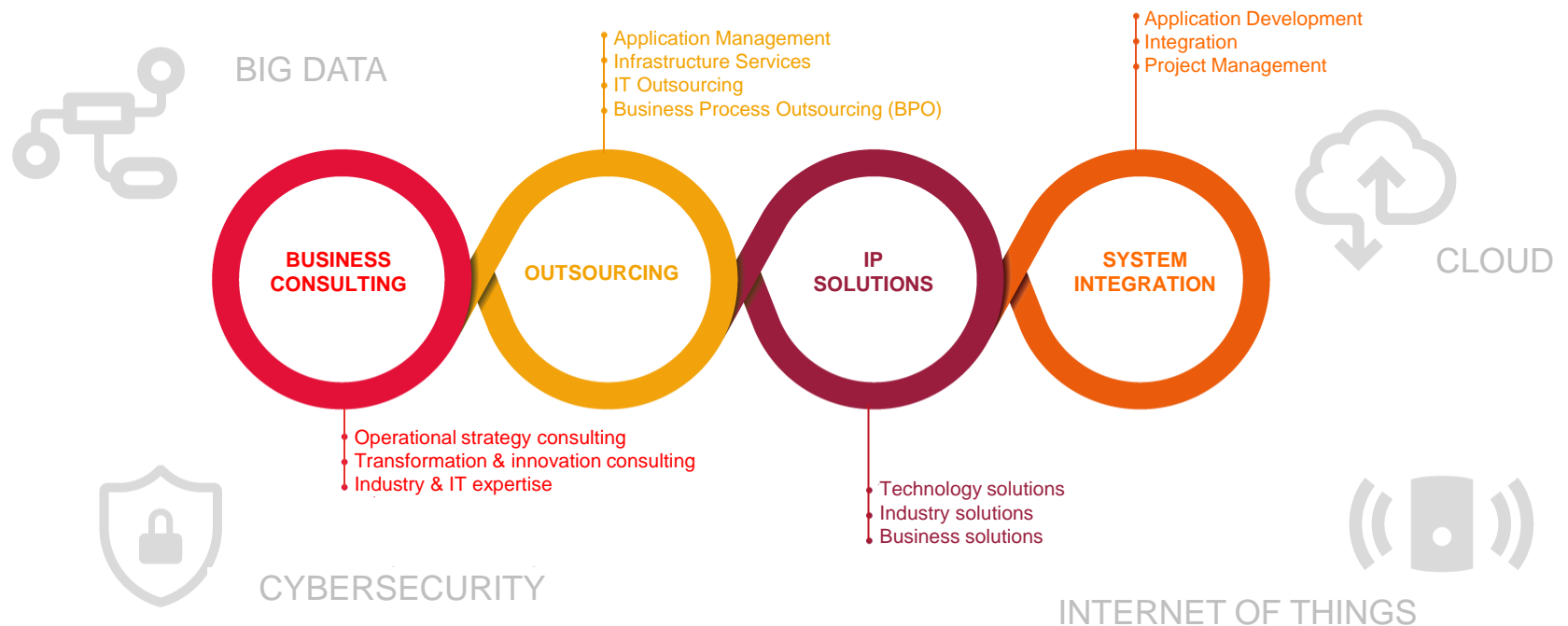


## Distribution of turnover

Per business line



# An end-to-end approach of your digital transformation enabled by our industry-focused organization



# Le Centre d'excellence de Montpellier

## Offre Portails, CMS, e-commerce et mobilité



Portails

Sites transactionnels pour favoriser le partage et l'accès à l'information entre le SI et l'utilisateur Web  
Partenaires clés :



CMS

Gestion et publication de contenu, simplification de la contribution et de la collaboration  
Partenaires clés :



e-Commerce

Portail de services à valeur ajoutée pour le client et pour l'entreprise (chiffre d'affaires)  
Partenaires clés :



Mobilité

Encourager la multi-canalité entre les différents devices et plateformes  
Partenaires clés :



Studio Graphique

SEO, ergonomie, création graphique, intégration, web mastering, web analytique  
Partenaires clés :



Clients principaux



L'ORÉAL



# Introduction

## Tour de table

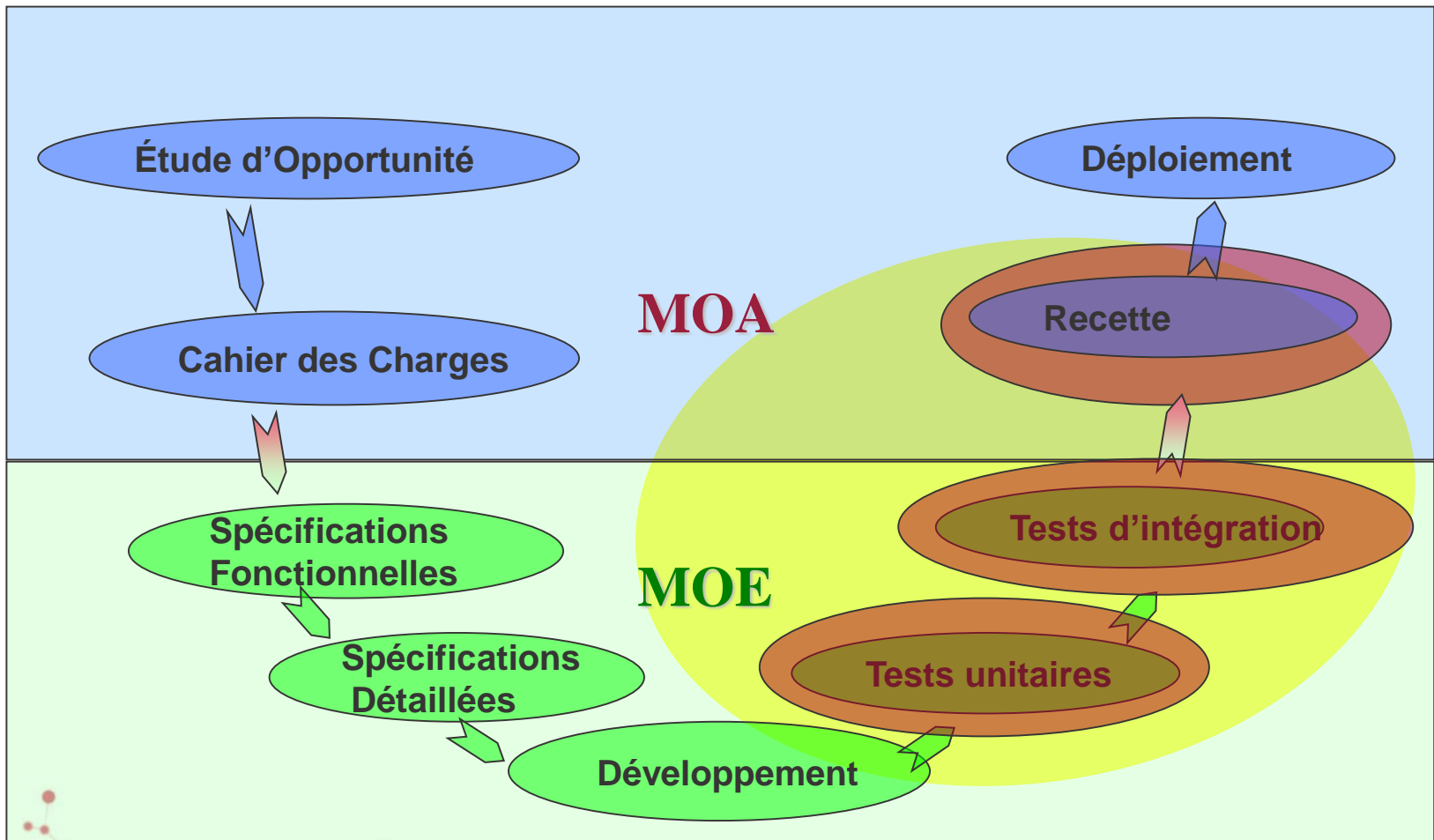
- Vos besoins / attentes ?
- Vos difficultés rencontrées suite à vos projets industriels / stages ?





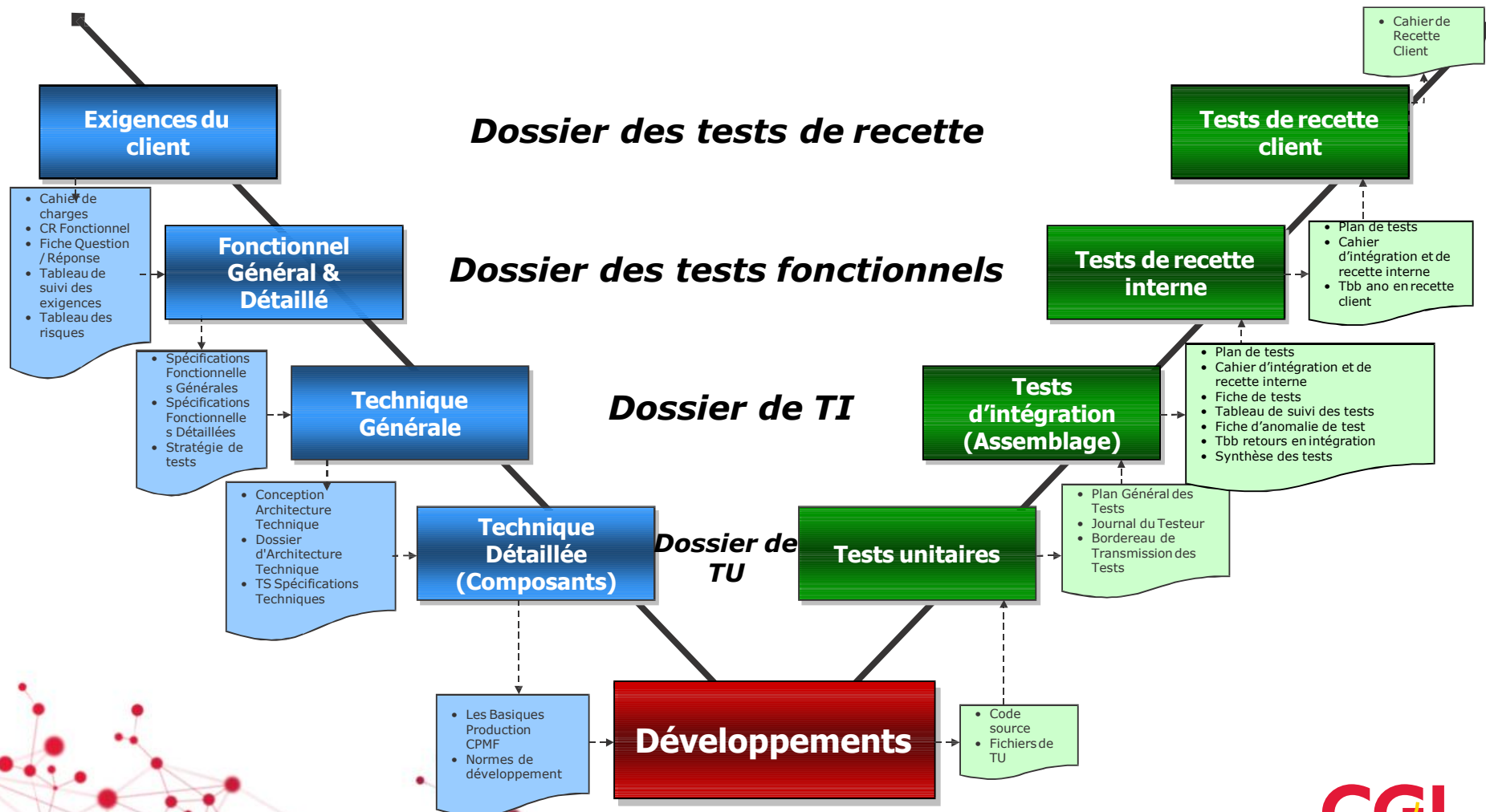
# Le cycle en V

## Intégration dans le cycle de vie du projet informatique



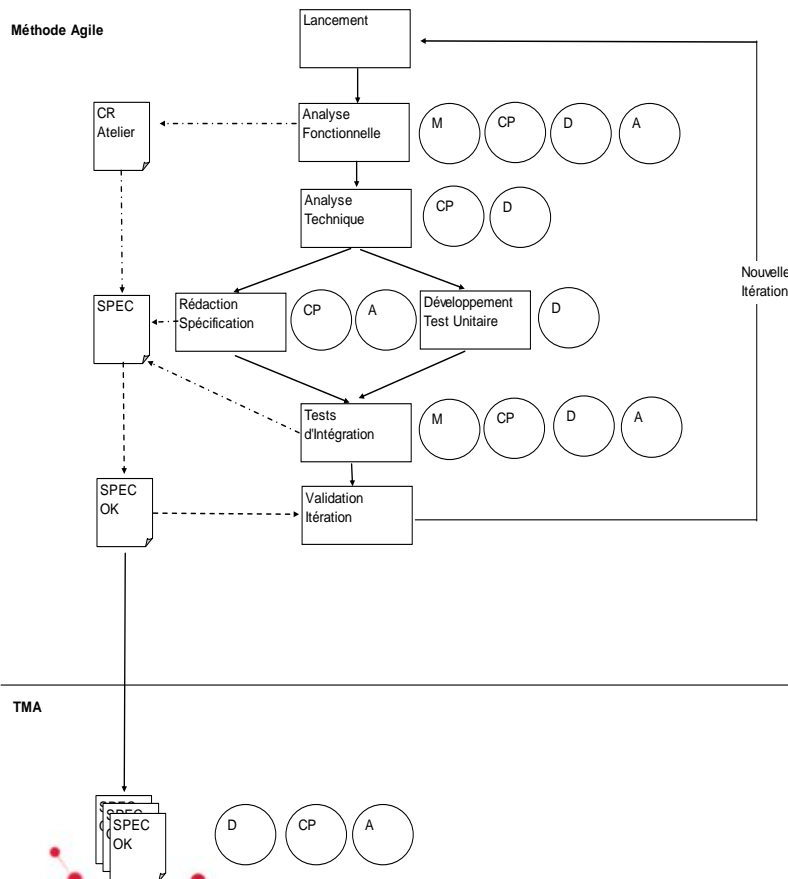
# Le cycle en V

A chaque phase d'analyse/production/conception correspond une phase de tests/vérification => production des scénarios de tests en même temps que les livrables de conception



# Emergence du mode Agile

## Intégration dans le cycle de vie du projet informatique



**Une itération** est un mini-projet dans le projet. Elle consiste à :

- analyser un besoin fonctionnel au périmètre clair avec les interlocuteurs métiers,
- concevoir la solution,
- réaliser les développements et tests unitaires,
- réaliser les tests d'intégration avec le métier,
- valider le résultat avant de passer à l'itération suivante.

Une itération dure au maximum **3 semaines**.

Cycle de développements courts, itératifs et incrémentaux.

Pas le processus linéaire du cycle en « V », chaque cycle de développement est réalisé de façon autonome.

Chaque cycle de développement porte sur la réalisation d'une fonctionnalité bien délimitée, de la *conception* générale à l'intégration de service.

Points de visibilité fréquents, revue régulière des livrables avec le client.

Démarche transparente et partagée.

Grande flexibilité dans les prises de décision et les choix

Pas d'effet tunnel

Meilleure convergence entre le besoin exprimé et la solution réalisée, tout en sécurisant le planning.

# Le mode agile

- Pas de grosse phase de conception en début de projet
- Pas de documents de spécifications de 200 pages à écrire / valider
- Pas d'effet tunnel pour le client lié à 800 jh de développement sans voir ce qui a été produit
- Pas de phase de tests en toute fin de projet quand trois quarts de l'équipe a été destaffée...



# Les valeurs du Manifeste Agile

Les personnes et leurs interactions sont plus importantes que les processus et les outils

Un logiciel qui fonctionne prime sur de la documentation

Le Manifeste Agile

La collaboration est plus importante que le suivi d'un contrat

La réponse au changement passe avant le suivi d'un plan



# Le mode agile : l'équipe

- Le Scrum Master

- veille à l'application des principes de Scrum
- élimine les obstacles rencontrés par l'équipe
- incite l'équipe à devenir autonome
- Bonne connaissance du mode Agile (Scrum)
- Aptitude à comprendre les aspects techniques
- Facilité à communiquer
- Capacité à guider
- Talent de médiateur
- Ténacité
- Inclination à la transparence
- Goût à être au service de l'équipe

- Le Product Owner

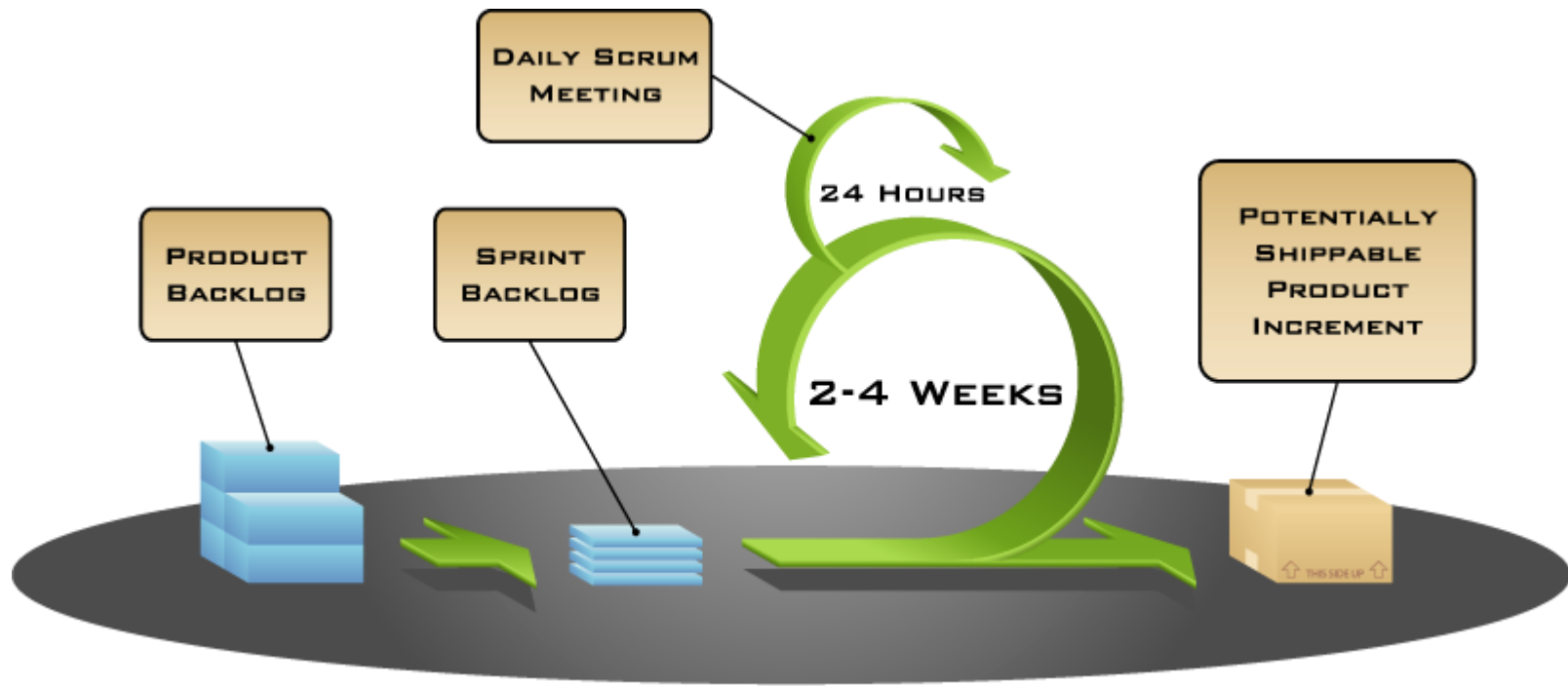
- C'est l'utilisateur côté client qui sera le garant du produit final
- Doit être disponible et impliqué
- Fournit la vision du produit
- Définit le contenu du produit
- Planifie la vie du produit
- Bonne connaissance du domaine métier
- Maîtrise des techniques de définition de produit
- Capacité à prendre des décisions (et rapidement)
- Capacité à détailler au bon moment
- Esprit ouvert au changement
- Aptitude à la négociation

- Les équipiers

- Ce sont eux qui font tout le reste
- Spécifient fonctionnellement et techniquement le produit
- Estiment et s'affectent les tâches
- Développent et passent les tests
- Bonne capacité d'analyse et de réflexion
- Autonomie
- Prise d'initiative



# Le mode Agile : déroulement d'un sprint



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE



# Typologie des tests, concepts, def.

## Les différentes phases de test

- **Tests Unitaires**

- **on élimine les erreurs de programmation**

L'objectif est de détecter les écarts entre les spécifications détaillées d'un composant, et ce qu'il fait effectivement pris unitairement.

- **Tests d'intégration**

- **on élimine les erreurs d'enchaînement**

L'objectif est de détecter les écarts entre les spécifications de la demande ou du projet, et ce que fait effectivement la partie applicative concernée.  
On parle aussi de recette interne et parfois d'homologation.

- **Validation / recette**

- **on élimine les erreurs fonctionnelles**

L'objectif est de détecter les défauts de l'application du point de vue des utilisateurs.

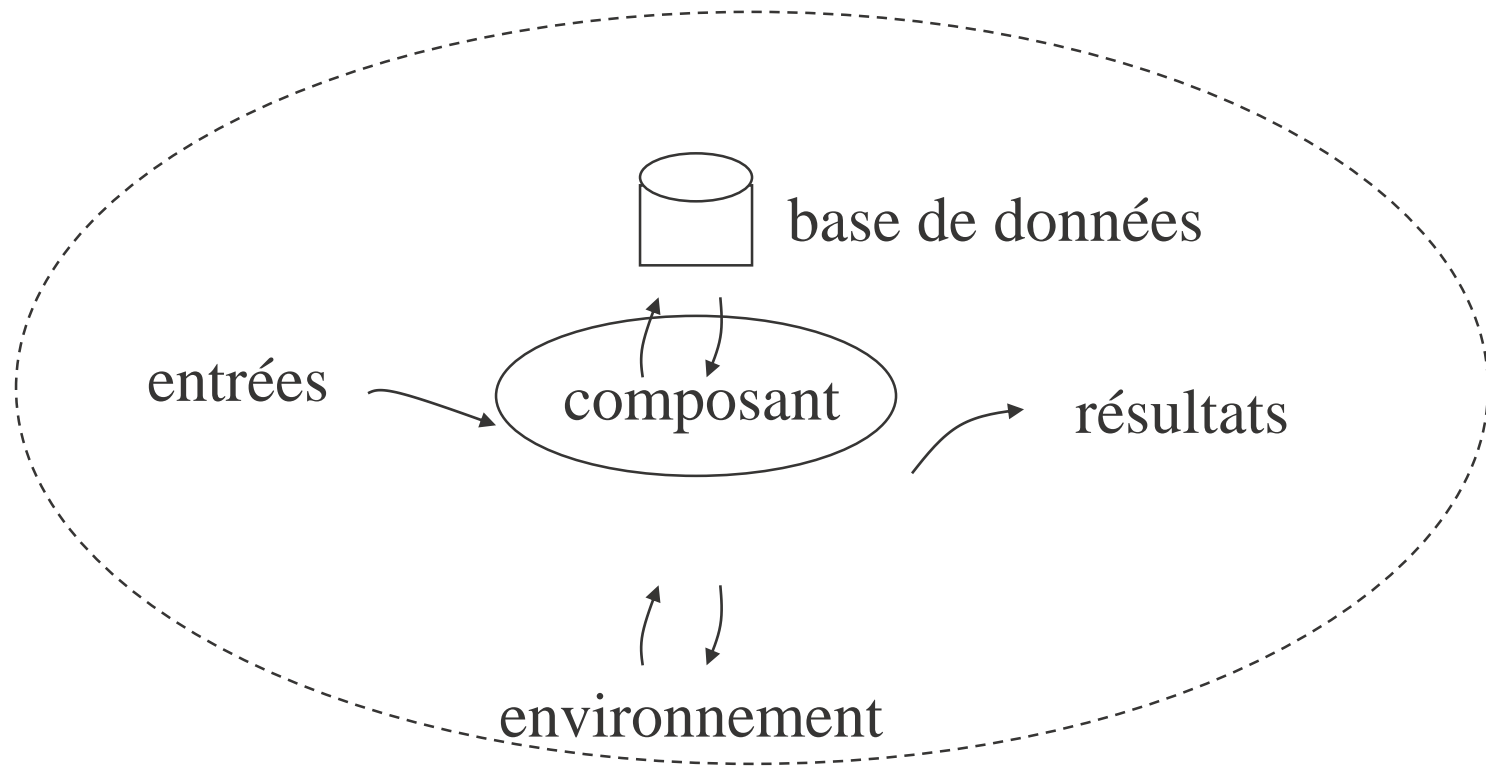




# Typologie des tests, concepts, def.

## Le périmètre de test

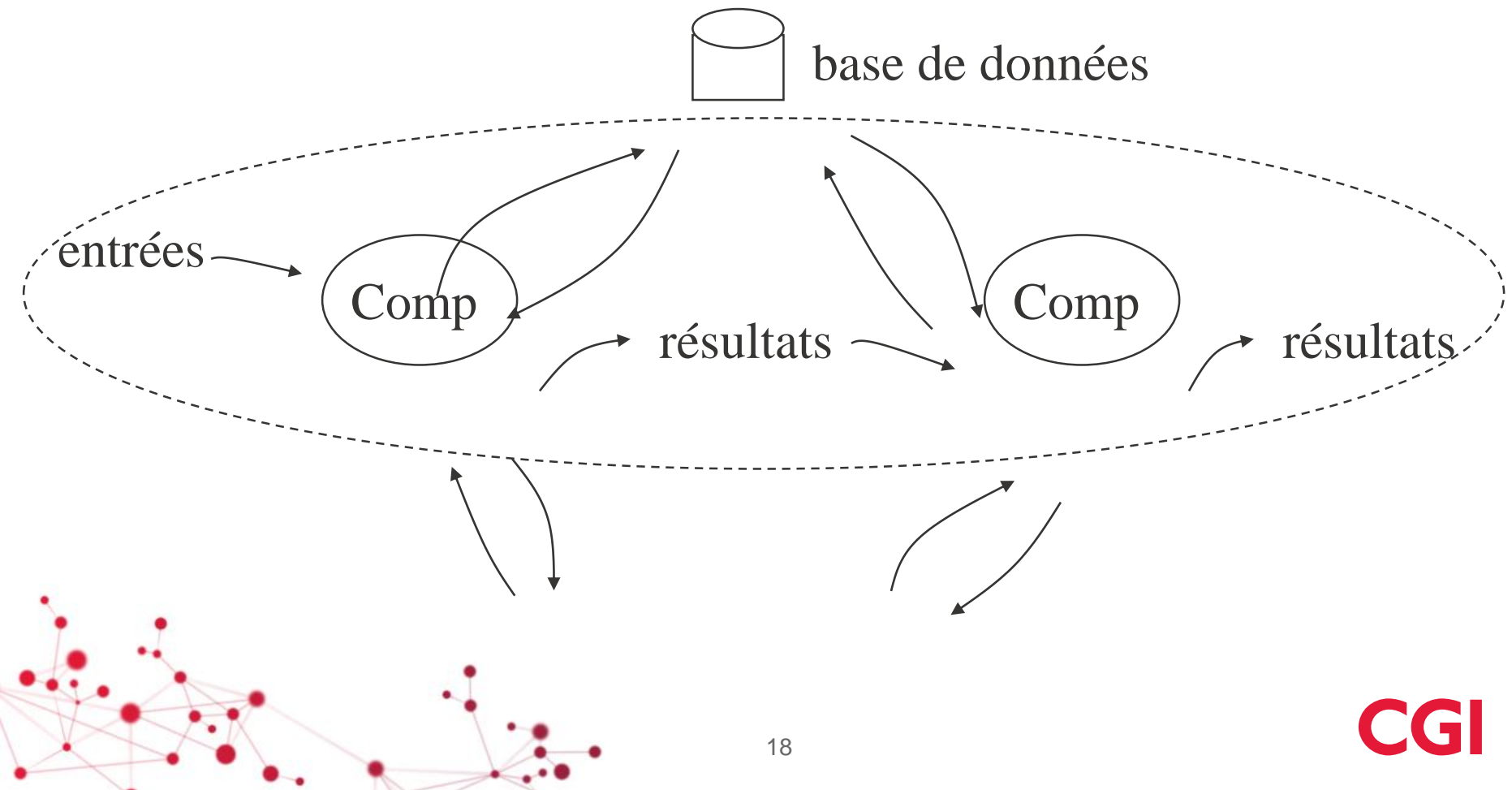
- **Test Unitaire**



# Typologie des tests, concepts, def.

## Le périmètre de test

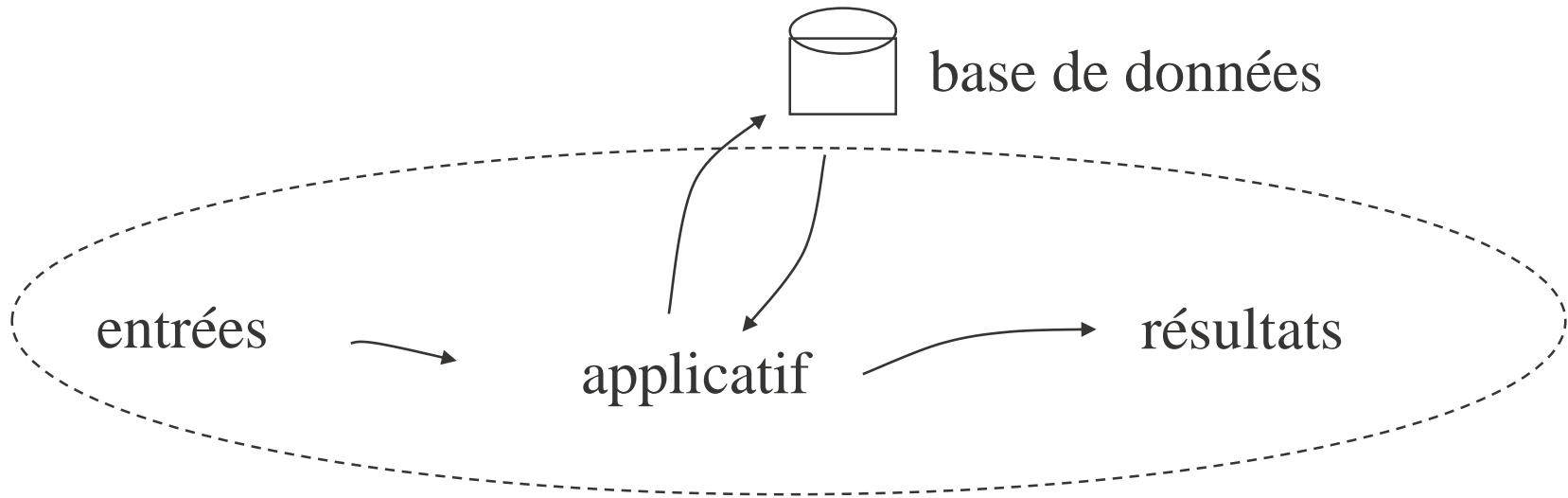
- Tests d'intégration



# Typologie des tests, concepts, def.

## Le périmètre de test

- Validation / Recette



# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

- **Approche globale :**

- **Tests de fonctionnement :**

Ils ont pour but de valider le bon fonctionnement des composants ou des parties applicatives concernées.

- **Tests de non régression :**

Ils ont pour but de vérifier que les composants validés lors des tours précédents n'ont pas subi de dégradation.



# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

- **Approche plus détaillée :**
  - Tests métiers ou fonctionnels (dits « boîtes noires »),
  - tests structurels (dits « boîte blanche »)
  - tests d'ergonomie,
  - tests d'interface,
  - tests de performance,
  - tests de volumétrie,
  - tests de robustesse,
  - tests de sécurité
  - ...



# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

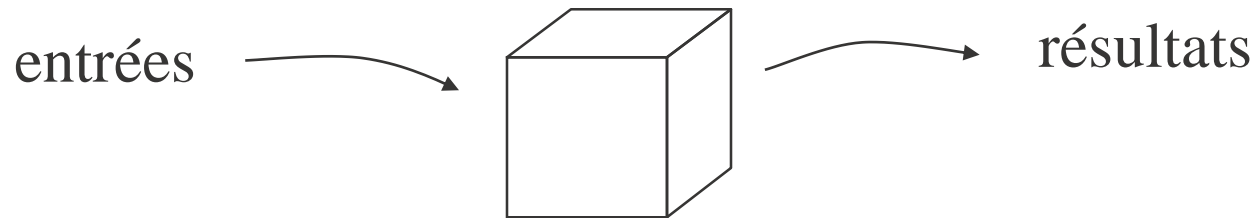
- **Approche « Boîte »:**
  - Test Boîte blanche (ou Test Structurel)
  - Test Boîte noire (ou Test Fonctionnel)



# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

### Les tests « boîte blanche »



- On cherche :
  - Le code non testé par les tests fonctionnels.
  - Le code mort.
  - Les erreurs dans la gestion des conditions.
  - Les erreurs dans la gestion des boucles.

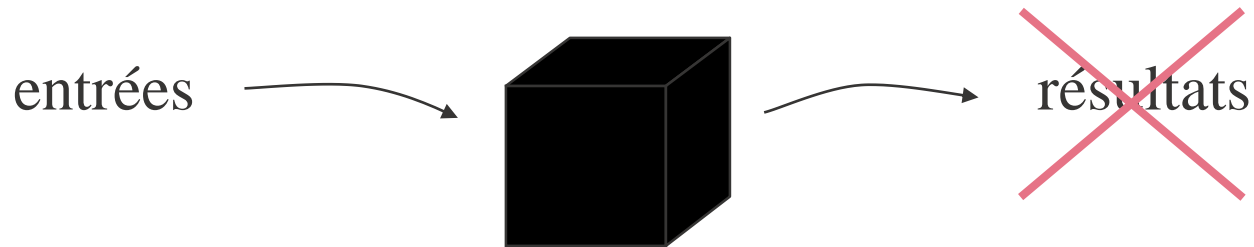


# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

### Les tests « boîte noire »

On cherche les cas dans lesquels le composant produit des résultats non conformes



Les résultats obtenus ne sont pas les résultats attendus.





# Typologie des tests, concepts, def.

## Les différentes natures de tests, plusieurs approches

- **Toutes ces natures de tests peuvent être mises en œuvre dans les différentes phases de tests**
  - unitaires,
  - d'intégration,
  - de recette.



# Pourquoi les TU

## Débat

- **Des exemples :**

- Coût annuel des bugs aux US ~ 60 milliards de dollars. 80% de temps de développement (mise en œuvre) serait consacré aux erreurs
- Fin 2004, en France 2 pannes géantes consécutives à des bugs (FT, Bouygues Telecom)
- Serveurs d'accès moyens de paiement d'un supermarché OUT un 24 décembre

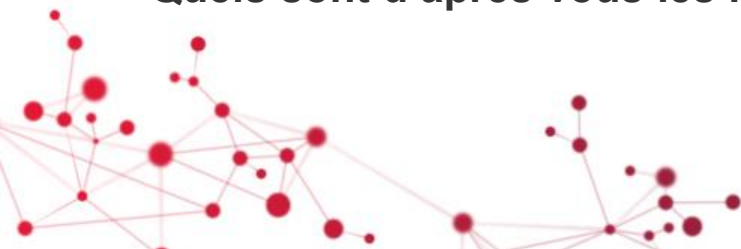
- **Des chiffres :**

- **40%** des incidents en production ont pour cause des erreurs qui auraient pu être évitées par une meilleure campagne de tests
- **60%** des échecs sur les projets informatiques proviennent de faiblesses sur le recueil des besoins et leur analyse
- **90%** des projets informatiques sont livrés en retard

- **La loi du 1 - 10 – 100**

- **Que faire pour éviter cette situation ?**

- **Quels sont d'après vous les impacts à ne pas faire de tests unitaires ?**



# Pourquoi les TU

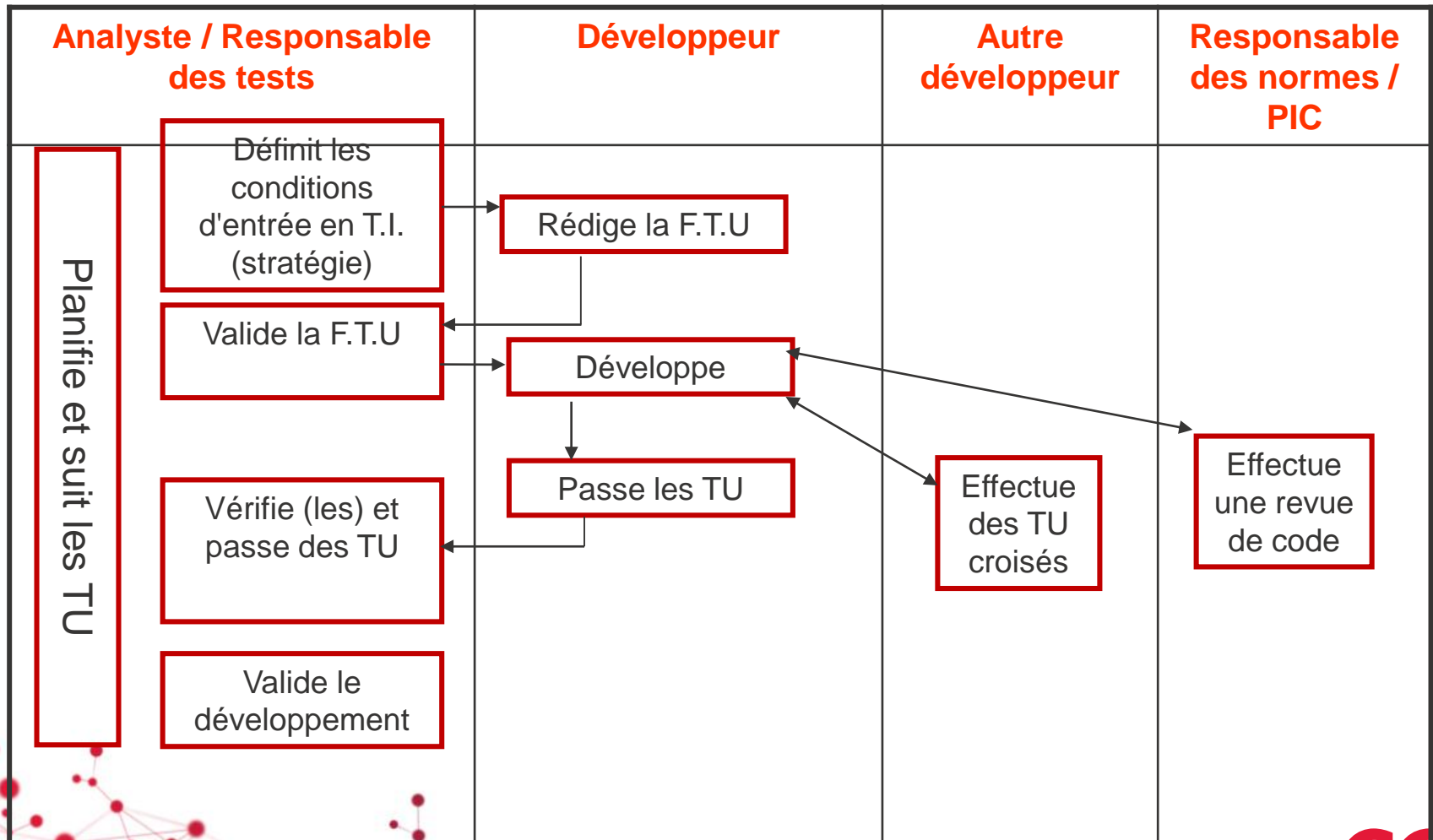
## L'impact de T.U insuffisants ou négligés

- **Impacts Production**
  - Augmentation de la charge et perte de temps
    - Analyser, identifier l'origine
    - Retour sur code compliqué quand ça date
    - Probabilité de régression plus importante
    - Anomalies en T.I. et en recette client
    - Perte de temps pour corriger, relivrer, retester les chaînes d'intégration
    - Baisse de la productivité
- **Impacts Finances**
  - Retard lors des T.I., risques sur la recette client
  - Augmentation des retours, donc de la charge
  - Génération de fiches d'anomalies / volume d'information
  - Dépassement des délais pouvant générer des pénalités (SLA)
  - ...
- **Impacts Business**
  - Satisfaction du client.
  - Risques pour la suite des relations client / projet.- Baisse de confiance
  - Baisse de l'image « Qualité » véhiculée
  - ...
- **Impacts pour l'ingénieur**
  - Image – Frustration – Redondance des tâches – Surcharge de travail
  - Evaluation des compétences : rigueur, autonomie, analyse → progression ralentie
  - Rentabilité à titre individuel



# Les processus de TU

## Synoptique global



# Les processus de TU

## Zoom sur les jeux d'essai

- La problématique des jeux d'essai est souvent passée sous silence. Mais comment tester sans jeu d'essai ?
  - Existe-t-il des jeux d'essai ?
  - Les jeux d'essai disponibles sont-ils adaptés aux cas à tester ?
  - Qui fait les jeux d'essai et qui les valide ?
  - Le client fournit-il des jeux d'essai et sous quelle forme ?
  - Les jeux d'essai sont-ils directement exploitables ou faut-il les resaisir ?
  - Une charge a-t-elle été prévue pour constituer les jeux d'essai ?
  - Les jeux d'essai sont-ils partagés ? Les jeux d'essai sont-ils capitalisés ?



# Les processus de TU

## Zoom sur la non régression

**Vous avez oublié de tester la non régression, pourtant :**

- **Vous avez modifié plus de 10% du code !**
- **La technologie utilisée est connue comme capricieuse et des effets de bords sont à craindre !**
- **Le traitement modifié est super critique et tout dysfonctionnement peut générer un conflit sociale (ex: paie), un arrêt de la production (ex: process), un énorme manque à gagner (facturation / trésorerie) ... !**
- **Le programme modifié est ultra-complexe et toute modifications est susceptible d'engendrer des dysfonctionnements !**
- ...



# Les stratégies de Tests

## Pourquoi une stratégie ?

*La problématique d'une opération de Recette fait apparaître des contraintes antagonistes qu'il est nécessaire d'optimiser :*

*D'une part, l'objectif premier des tests fonctionnels est d'obtenir une application « zéro défaut ». Ceci oblige en théorie à effectuer ces tests de manière exhaustive,*

*D'autre part, pour garder une vision pragmatique des projets informatiques, la période de recette est généralement limitée dans le temps par des contraintes de délai très fortes et très souvent impératives.*

*Il est également indispensable de minimiser les coûts de recette pour assurer un retour sur investissement acceptable.*

*Le coût des tests doit donc toujours être mis en balance avec le coût des impacts d'un dysfonctionnement éventuel (au niveau stratégique, métier et informatique).*



# Les stratégies de Tests

## Analyse de risques

*L'analyse de risques permet d'optimiser le nombre de cas à tester et contribue ainsi à la réduction des coûts.*

*Dimensionner la couverture en fonction des risques et des exigences métier*

*Elle est initialisée par classement des fonctions à tester en trois catégories, selon des critères objectifs (nombre d'utilisateurs, % d'utilisation des fonctions, produits, criticité,..) :*

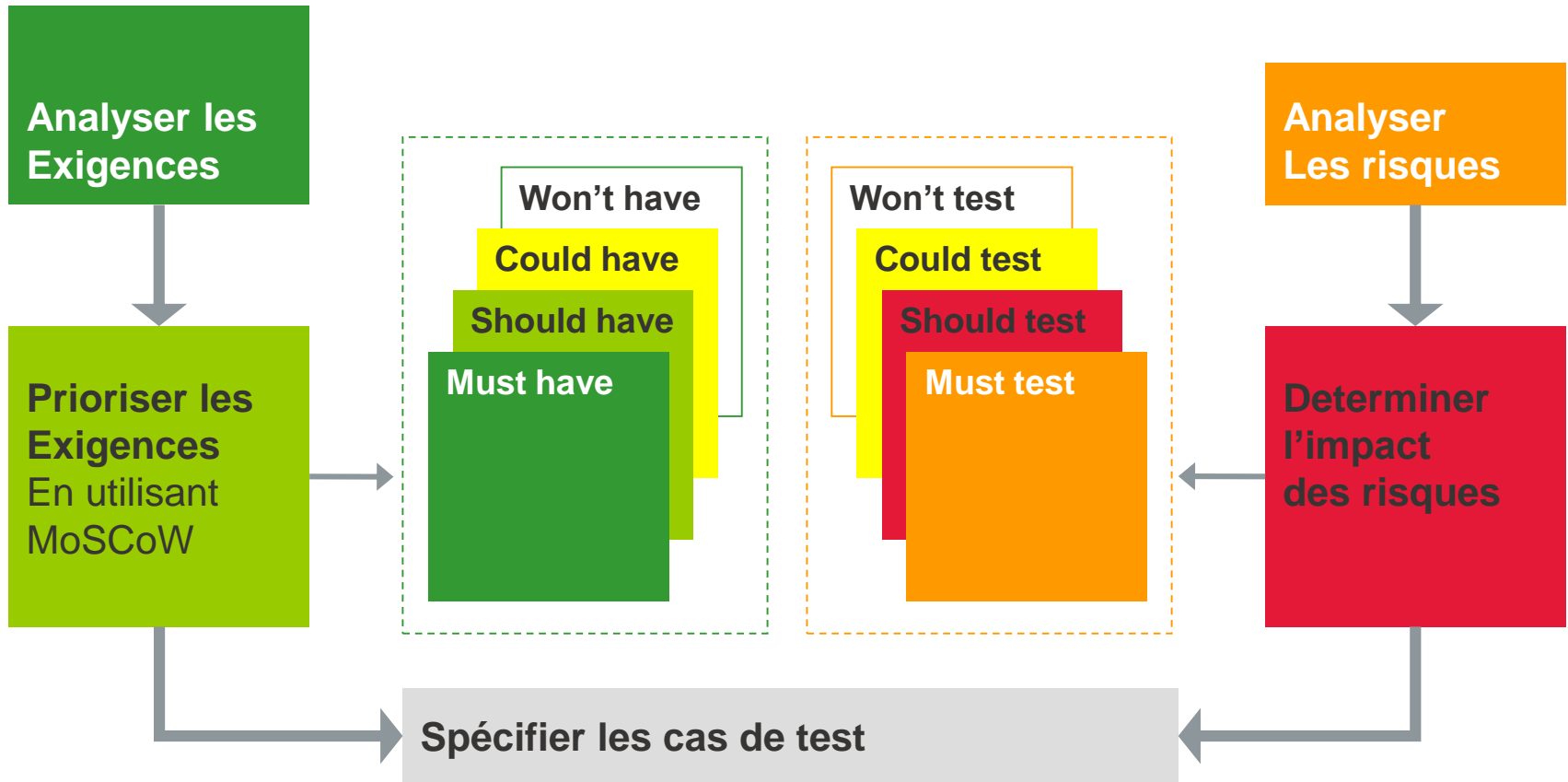
- *Les fonctions et données stratégiques,*
- *Les fonctions et données très utilisées ou très courantes,*
- *Les fonctions et données annexes ou rares et non stratégiques.*

*Dans la phase de réalisation des cas de test, le planning prend en compte l'analyse de risques pour réaliser en priorité les cas de tests les plus stratégiques et respecter les délais impartis pour la recette.*





# Risques & Requirements (exigences Métiers)



MoSCoW signifie « Must Test , Should Test, Could Test, Won't Test », « Must Test » signifiant la plus haute priorité de test et « Won't Test » la plus basse



# Priorisation des risques produits

Si le module contient une erreur ...	... il y a des conséquences financières pour nos clients.	Tous les clients	<b>Must test</b>
		Un seul client	<b>Should test</b>
	... il n'y a pas de conséquence financière pour nos clients.	Tous les clients	<b>Should test</b>
		Un seul client	<b>Could test</b>
	... il y a des conséquences financières en interne.	Pas de contournement	<b>Could test</b>
		Contournement	<b>Won't test</b>
	... il n'y a pas de conséquence financière en interne.	Pas de contournement	<b>Could test</b>
		Contournement	<b>Won't test</b>

# Les stratégies de Tests

## Multi - Campagnes de tests

*La méthode repose sur une stratégie de tests en campagnes, garantissant la fiabilité maximale de l'application qualifiée.*

*Une campagne de tests consiste à exécuter un ensemble de cas de tests sur un périmètre donné, en nombre décroissant au fur et à mesure des campagnes.*

*La couverture des cas de tests exécutés lors des campagnes varie en fonction des domaines concernés. Elle dépend de l'optimisation coût / qualité recherchée.*

- *La 1ère campagne de tests détecte le maximum d'anomalies en exécutant 100% des cas de tests*
- *La 2nde campagne permettra d'identifier les anomalies résiduelles et de valider la correction des anomalies détectées lors de la première campagne en exécutant au moins 75% des cas de tests*
- *La 3ème campagne est une phase de validation globale lors de laquelle au moins 50% des cas de tests sont exécutés*



# Les principales stratégies de tests unitaires

## Approche technique

- **Objectif :**
  - Mettre en place une solution rapide et peu coûteuse pour industrialiser les tests unitaires et homogénéiser le niveau de couverture
- **Solution : Check liste**
  - Points de contrôles systématiques
  - Adaptation à la technologie et au contexte d'exécution
  - Prise en compte des principaux choix d'implémentation :
    - une fonctionnalité codée de manière centralisée pour tous les cas d'utilisation pourrait n'être testée qu'une seule fois
  - Sans oublier l'état de l'art et les tests aux limites
- **Exemples de fiche de tests unitaires:**



@Fichtu...



# Les principales stratégies de tests unitaires

## Exemple de checklist IHM

### TU Standards

#### Ergonomie Générale

Cas de test 1A : Affichage de l'écran
• Vérification de la position de l'écran
• Vérification du positionnement des objets dans l'écran
• Vérification de la conformité des couleurs
• Vérification de la disposition des contrôles
• Vérification de la conformité des polices (Taille, caractéristique)
• Vérification de la conformité des libellés (Taille, emplacement, caractéristiques)
• Vérification de l'affichage des aides (boutons, champs et/ou zones)
• Vérification de l'affichage des menus contextuels (clic long sur Mac)
• Vérification du type de message d'erreur qui s'affiche (pop up ou dans l'écran) et s'il est accompagné d'un signal sonore

Cas de test 1B : Fonctionnement des touches de contrôle
• Gestion des tabulations (changement du focus sur les différents contrôles)
• Gestion des flèches du clavier sur chacun(e) des champs (zones)
• Gestion des touches de raccourci
• Gestion des touches « Entrée » (Touche pavé numérique et touche pavé alpha)
• Gestion de la touche « BackSpace »
• Gestion de la touche « Echap »

#### Fonctionnalité de l'écran

Cas de test 2A : Gestion de l'écran
• Initialisation de tous les champs conforme
• Positionnement du focus à l'ouverture
• Gestion du clic sur les objets (positionnement du focus <u>etc</u> )
• Gestion du double clic sur les objets (création d'un événement d'ouverture d'écran <u>etc</u> )
• Gestion des contrôles associés aux champs de l'écran (actif, inactif, numérique, alphanumérique, date <u>etc</u> )
• Gestion de la forme du curseur sur certains boutons
• Gestion des fonctions d'édition (copier, coller <u>etc</u> ) sur les champs de l'écran
• Gestion des événements suite au clic sur un bouton de l'écran (Prise en compte des modifications dans l'écran, certains champs deviennent inactifs/actifs, ouverture d'une fenêtre, sortie <u>etc</u> )

Cas de test 2B : Gestion périphérique
• Intégration de la fonctionnalité testée au sein de l'application
• Modifications prises en compte dans le fichier commande suite à l'enregistrement de celle-ci
• Conformité avec les règles de gestion décrites dans les <u>spec</u>
• Tests effectués sur des commandes de millésimes différents



# Les principales stratégies de tests unitaires

## Exemple de checklist graphique

CHECKLIST GRAPHIQUE													
OK													
Prio.	Tâche	Type de tâche	Applicable?	Spécifique?	IE7	IE8	IE9	IE10	IE11	FF	CHR	SAF	
10	Vérifier le bon emplacement des différents éléments de la page par rapport aux maquettes	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
20	Vérifier la hauteur des blocs (div)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
30	Vérifier de la largeur des blocs	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
40	Vérifier les espacements entre les blocs	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
50	Vérifier les espacements entre les blocs de texte et les images	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
60	Vérifier les espacements entre les paragraphes de texte	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
70	Vérifier que les blocs de textes sont justifiés à droite si tel est le cas sur la maquette	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
80	Vérifier que le style des inputs soit correct (couleur des bordures, ...)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
90	Vérifier qu'il n'y ait pas d'erreur d'encodage de caractères dans les différents textes du site	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
100	Vérifier la taille des caractères	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
110	Vérifier la décoration (italique, souligné, gras)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
120	Vérifier la couleur du texte	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
130	Vérifier la police de caractères utilisée	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
140	Vérifier le comportement au survol des liens	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
150	Vérifier le comportement au survol des tooltips (infobulles)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
160	Vérifier l'alignement vertical des titres avec les puces dans le cas de listes	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
170	Vérifier le bon fonctionnement des liens hypertextes	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
180	Vérifier la bonne ouverture du client de messagerie par défaut lors d'un clic sur un lien mailto	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
190	Vérifier les champs textes : la longueur maximum doit être respectée	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
200	Vérifier les champs textes : le type de donnée saisi doit être correct (ne pas pouvoir entrer de texte dans un champ numérique...)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
210	Vérifier la qualité des images découpées par rapport aux maquettes (images trop pixélisées ou non transparentes...)	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
220	Vérifier la taille des images générées par rapport à celle des images présente sur la maquette	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
230	Vérifier si un texte est tronqué à cause d'un div avec une hauteur trop faible	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
240	Vérifier le bon affichage de tous ces éléments lors des redimensionnements de fenetre.	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
250	Vérifier le bon emplacement des messages flash ( msg erreur / notice / warning / success )	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
260	Vérifier le style ( longueur/ taille ) des messages flash ( msg erreur / notice / warning / success )	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
270	Vérifier la feuille de style	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	
280	Vérifier la feuille de style utilisée dans les emails	Graphique	oui	non	OK	OK	OK	OK	OK	OK	OK	OK	

# Les principales stratégies de tests unitaires

## Approche fonctionnelle

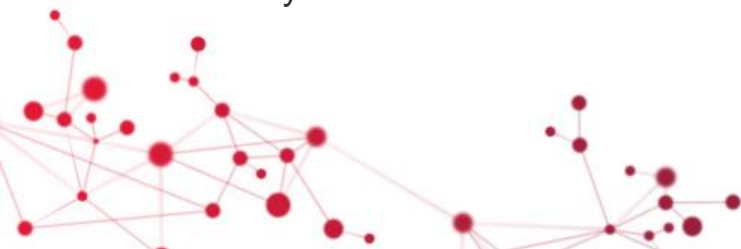
- **Objectif :**
  - Vérifier la bonne implémentation de règles de gestion précises et éventuellement complexes
- **Solution : Scénario de tests**
  - Liste des cas à tester sur la base des spécifications, maquettes et autres règles métier
  - Niveau de détail variable (cas de test ou jeux d'essai complet)
  - Niveau de couverture variable
- **Exemple de fiche de scénario de test :**   
Fiche Scénario de Tests
- **Exemple de journal de tests :**   
Journal de tests



# Les principales stratégies de tests unitaires

## Approche « outillage »

- **Objectif :**
  - Sur des architectures complexes (ex: n-tiers)
    - Réduire les coûts de développement des outils complémentaires utiles au test de couches intermédiaires
  - Face à des applications très critiques
    - Améliorer la qualité par une analyse objective de la couverture des tests
    - Réduire le risque de régression sans allonger les délais, en facilitant la « rejouabilité » des tests.
    - Identifier les parties à optimiser en priorité
  - Etc...
- **Solution : Outillage**
  - Développements spécifiques ou produits du marché
- **Exemples**
  - Développement des tests avec JUnit sur J2EE/NUnit sur .Net
  - Analyse de couverture avec Ncover ou CoverageEye





# Les principales stratégies de tests unitaires

## Approche « performance »

- **Objectif :**
  - S'assurer du niveau de performance requis
- **Solutions**
  - Utiliser des jeux d'essai représentatifs en volume
  - Identifier les points critiques grâce éventuellement à des outils de profiling
  - Utiliser des outils de stress (ex: JMeter, Load Runner...)



# Les principales stratégies de tests unitaires

## Ce qu'il faut retenir

- **Pas de solution idéale généralisable**
- **Chaque approche à elle seule est insuffisante**
- **La stratégie de test doit être construite spécifiquement pour chaque projet en fonction des objectifs et des moyens**
- **L'utilisation de check listes complétées par quelques cas de test bien choisis offre globalement de bons résultats**
- **Pragmatisme et souplesse sont de rigueur :**
  - Dimensionner la couverture par rapport au risque
  - Garder à l'esprit que le coût d'un dysfonctionnement augmente de manière exponentielle plus nous avançons dans le projet
  - Penser aux revues de code
  - 0 défaut comme à NASA est illusoire avec nos ratios habituels
  - L'automatisation complète des tests est coûteuse, penser à la rentabilité
  - En optimisant 90% du code n'utilisant que 10% du temps CPU on ne gagne pas grand chose. Mieux vaut axer ses efforts sur les 10% restant occupant 90% du temps CPU.



# Les outils

## Aujourd'hui

Dans une démarche d'industrialisation et de gain de productivité, la mise en œuvre d'un outil quelconque est fortement recommandée, voir indispensable, dans un processus DevOps.



# DevOps

Le DevOps vise à trouver **des leviers d'améliorations** autour des processus, méthodes et organisations afin de **fluidifier** les échanges du développement à la production. Ceci passe par la mise à disposition d'outils permettant d'industrialiser le cycle de vie d'une application.

## Avantages

- ✓ Réduction du *time to market* : capacité à améliorer l'expérience client et à innover en continu
- ✓ Accélération du retour sur investissement avec des cycles plus courts
- ✓ Equipes déchargées des tâches répétitives pouvant ainsi se concentrer sur les tâches à forte valeur ajoutée
- ✓ Capacité à mettre en place des déploiements continus
- ✓ Capacité à mettre en place des tests automatisés



# Déploiement automatique

Les solutions de déploiement automatique visent à livrer de manière **automatique** et **rapide**. Ces solutions apportent également un **gain qualitatif indéniable** en s'assurant que les gestes qui seront appliqués sur un environnement le seront à l'**identique** quelques soit l'**environnement** ou la **branche applicative**

## Avantages

- ✓ Rapidité de déploiement
- ✓ Possibilité de déployer automatiquement (toutes les nuits par exemple ) et profiter d'un environnement testable tous les matins
- ✓ Sécurisation des déploiements en testant les opérations à l'identique sur plusieurs environnements
- ✓ Facilité de mise en œuvre des *Rollback*



# Qualimétrie

Dans une approche DevOp, une application évolue dans le temps. Il faut cependant **s'assurer que la qualité est au rendez-vous** tout au long des versions. Mise en place de solution Qualité permettant de déployer et **suivre des indicateurs** pertinents.

## Avantages

- ✓ Contrôle de la qualité de l'application
- ✓ Timeline du suivi qualité tout au long du projet
- ✓ Maîtrise de la dette technique
- ✓ Facilité de mise en place de plan de résorption
- ✓ Rapport et dashboard en temps réel
- ✓ Audit accessibilité et bonne pratique SEO



# Monitoring

Toute application stratégique doit être **mise sous surveillance**. Les solutions de monitoring permettent de contrôler que les parcours **essentiel au business soient opérationnels** et remontent des alertes le cas échéant pour **accélérer** la prise en compte d'indisponibilité d'un brique SI.

## Avantages

- ✓ Application monitorée
- ✓ Mise en place et suivi d'indicateur comme le taux de disponibilité ou des temps par parcours
- ✓ Possibilité de monitorer une application à partir de points géographiques différents
- ✓ Mise à disposition de traces permettant de meilleures analyses

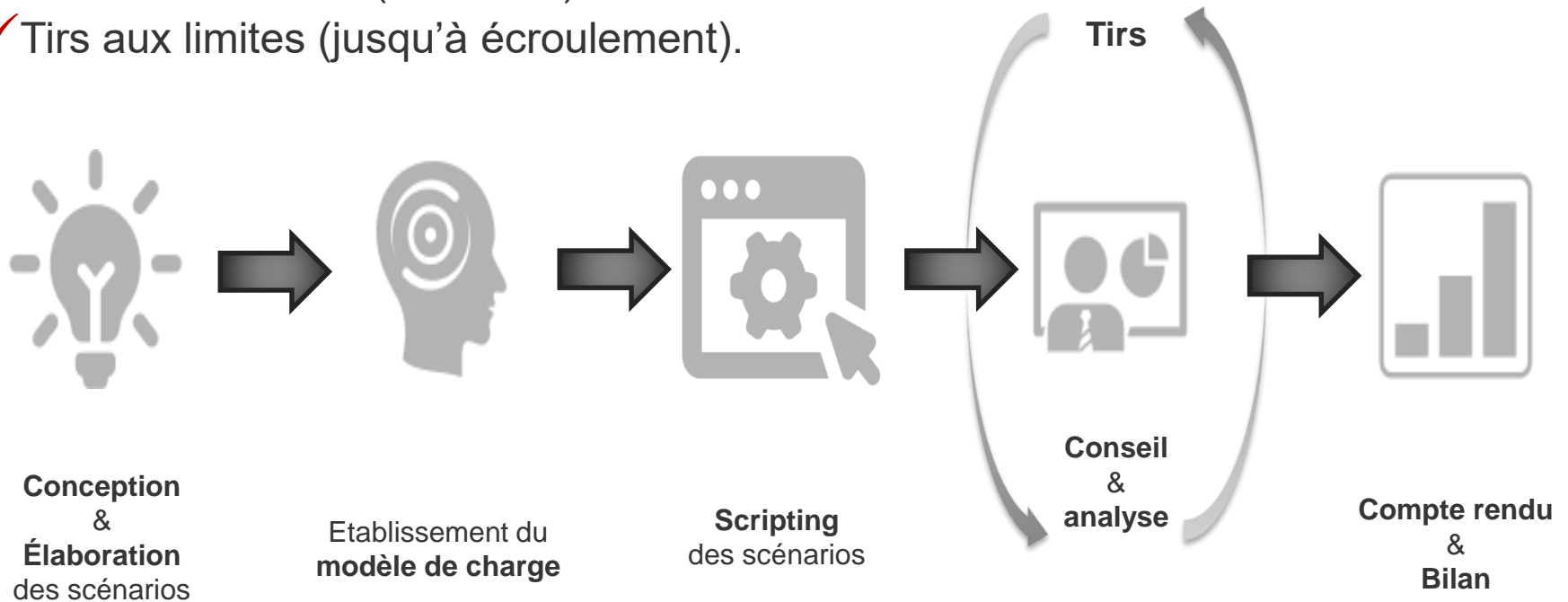


# Test de performance

Les experts performances peuvent accompagner et exécuter **toutes les phases d'une campagne de tirs** de performances.

Une campagne de tirs complètes se compose de 3 types de tirs :

- ✓ Tirs nominaux (2 à 3H de plateau),
- ✓ Tirs d'endurances (12 à 24H),
- ✓ Tirs aux limites (jusqu'à écroulement).





# Sécurité

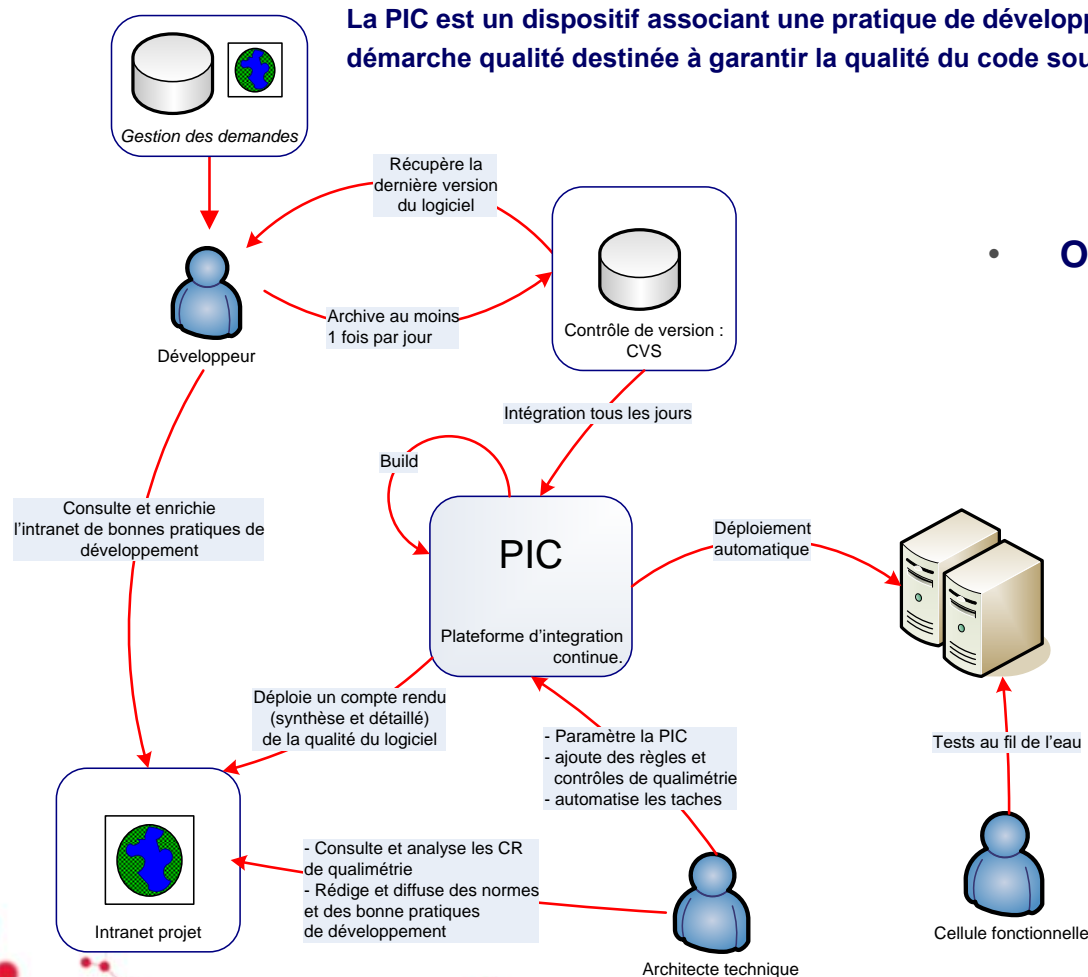
La sécurité ainsi que la préservation des données est un **enjeu majeur** de toute transformation digitale. C'est pour cela que la mise en place d'une **solution complète** de conseil, d'audit et de suivi de sécurité applicative est nécessaire.

## Avantages

- ✓ Alerte au plus tôt sur les dépendances potentiellement exposées
- ✓ Sensibilisation des équipes
- ✓ Sécurisation des développements
- ✓ Prise en compte de la sécurité au plus tôt dans les développements



# PIC : La plateforme d'intégration continue

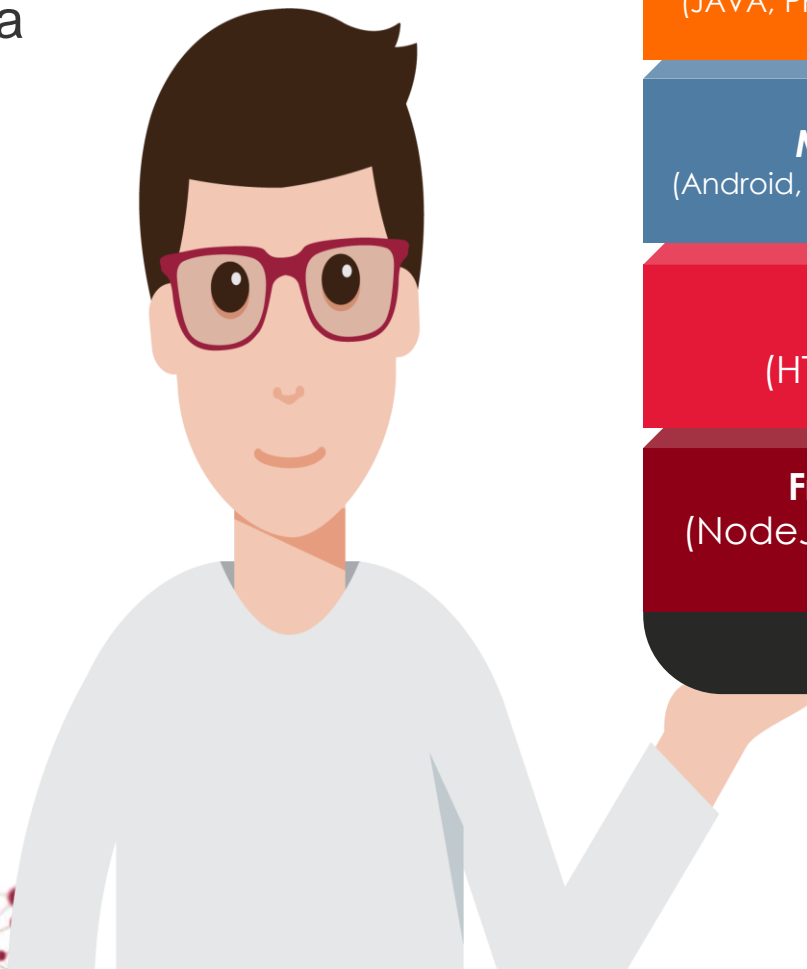


## Objectifs de la PIC

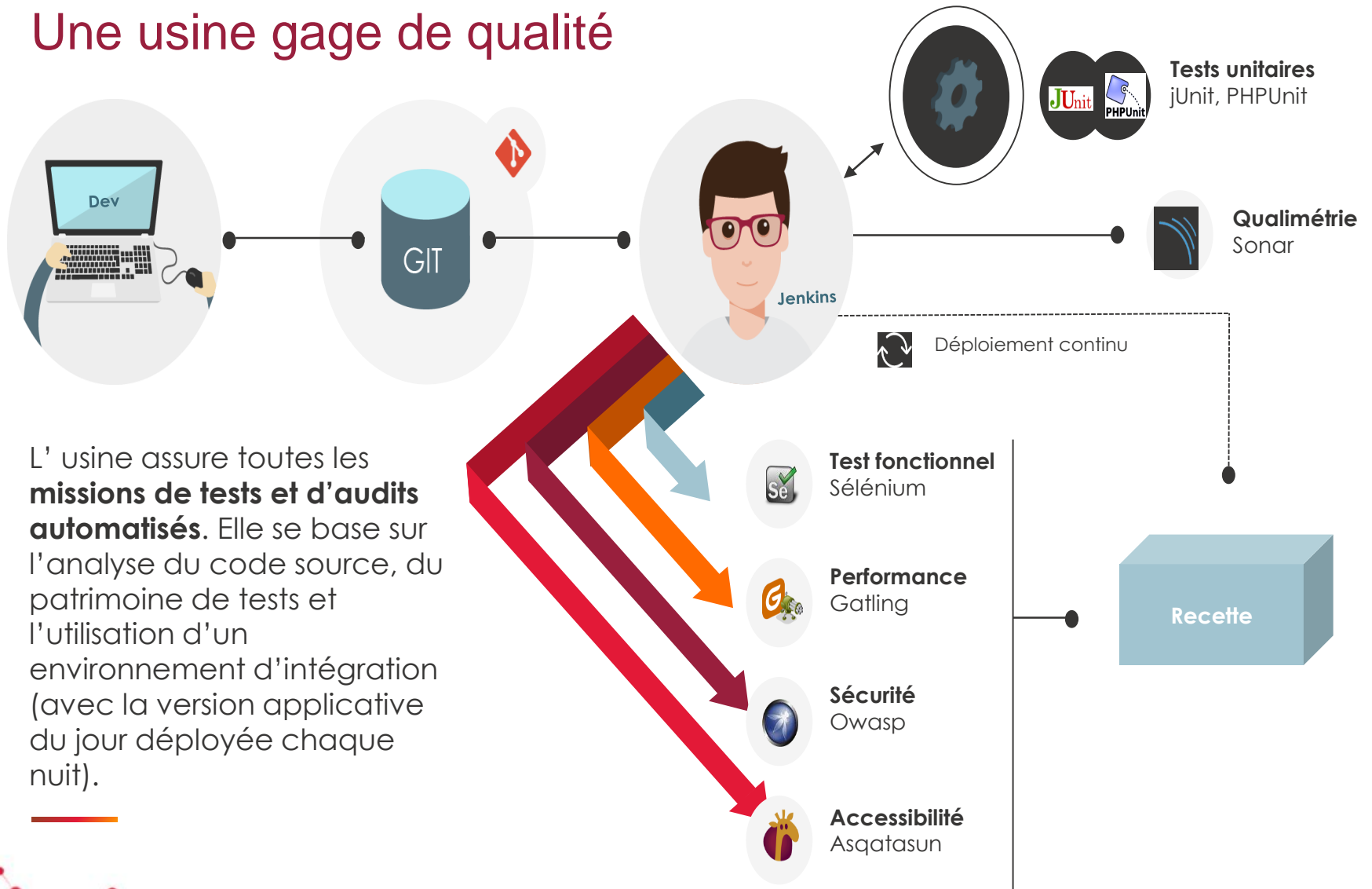
- Éliminer les sessions intensives de débogage.
- Détecter les problèmes généralement le jour même où ils ont été introduits
  - La qualification de la cause en est facilitée
  - L'impact en est limité
- Obtenir un code de qualité pour une application facilement maintenable et évolutive.

# Une plateforme au cœur de la transformation digitale

La **PIC** est construite sur des containers Docker. Elle est donc multi-technologies et supporte les problématiques de la transformation digitale.



# Une usine gage de qualité



L'usine assure toutes les **missions de tests et d'audits automatisés**. Elle se base sur l'analyse du code source, du patrimoine de tests et l'utilisation d'un environnement d'intégration (avec la version applicative du jour déployée chaque nuit).

# Conclusion

3 minutes  
pour déterminer  
3 idées fortes  
retenues lors de ce cours

