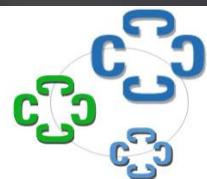




**Cognizant**  
Passion for building stronger businesses

Cognizant

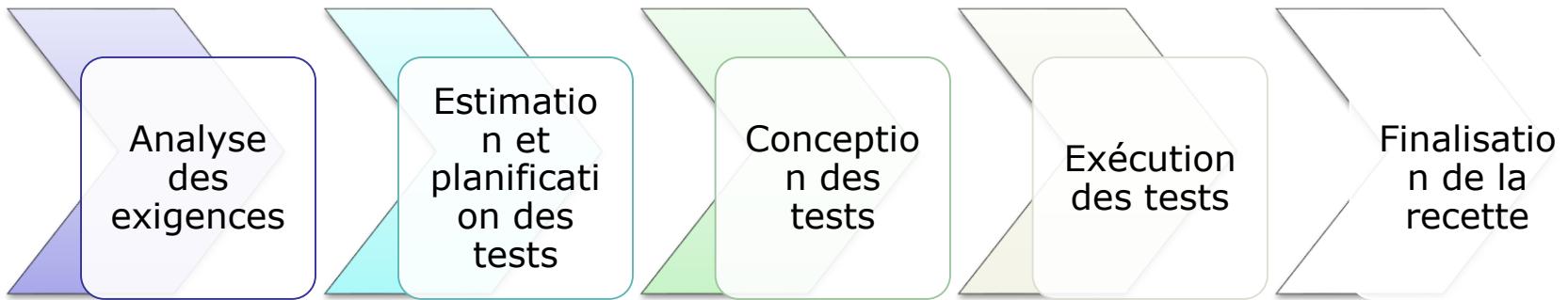
## Les phases du test en détails



A  
**FORTUNE  
1000  
COMPANY**

# Déroulement d'une recette fonctionnelle

- 5 phases essentielles :



# Sommaire

- **Analyse des exigences**
  - » La gestion des exigences
  - » Définition
  - » Catégories d'exigences
  - » Priorisation des exigences
  - » Attributs d'une bonne exigence
  - » Revue d'exigence
- Estimation et planification des tests
- Couverture de test
- Conception des tests
- Techniques de conception de test
- Exécution des tests
- Finalisation de la recette



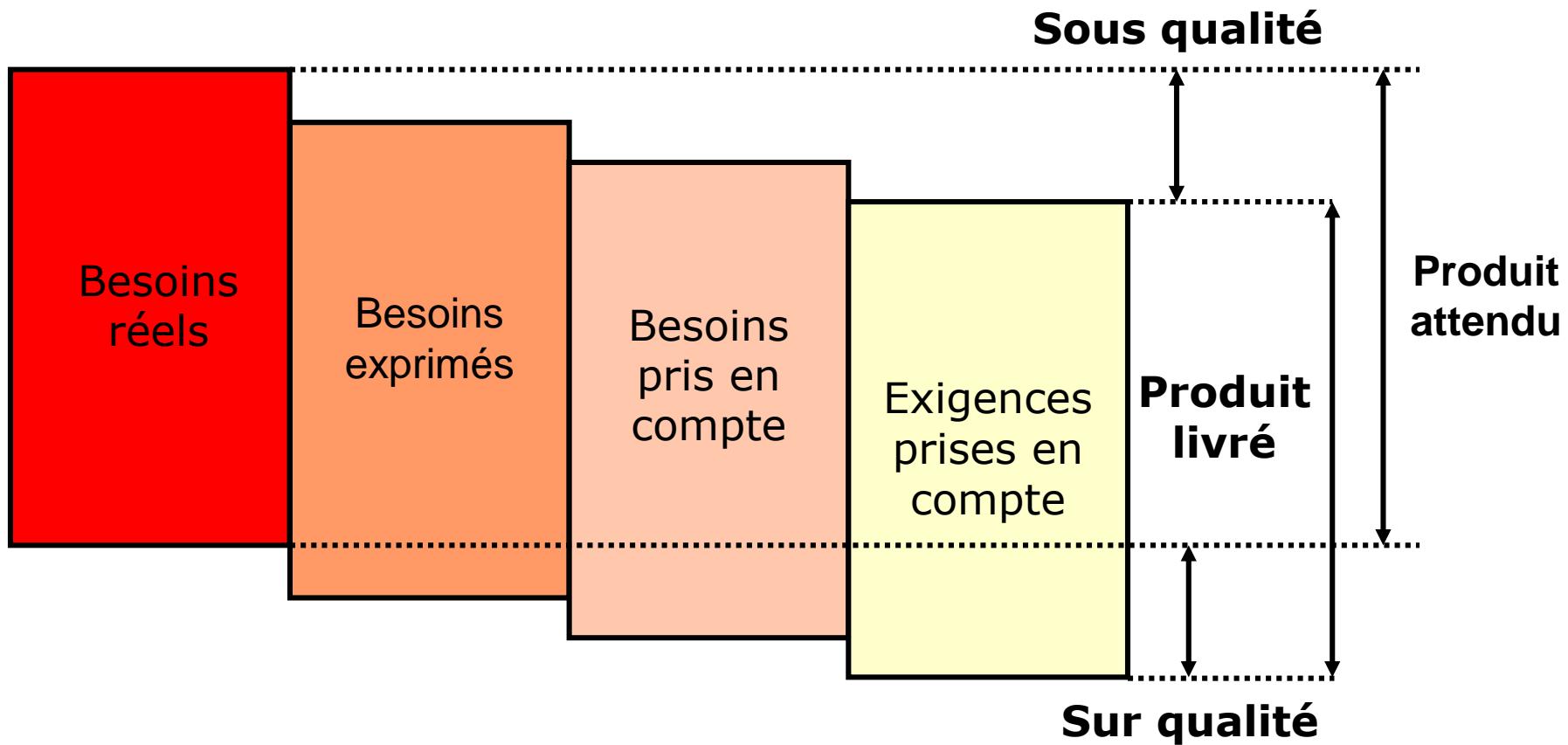
# *La gestion des exigences*

- La Gestions des exigences est aujourd’hui une activité majeure dans l’ingénierie logicielle. C’est une discipline qui se structure de la même façon que le test.
  - ✓ **Association IREB** : International Requirements Engineering Board
  - ✓ **Certification CPRE** : (Certified Professional of Requirements Engineering)
  - ✓ **CMMI** : Le Capability Maturity Model Integration décrit les activités liées à la gestion des exigences dans certains modèles de conception logicielle
- La gestion des exigences consiste à gérer les exigences hiérarchisées d'un projet, à détecter les incohérences entre elles et à assurer leur **tracabilité**.

# *La gestion des exigences*

- Une cause d'échec pour un projet : **les exigences**
  - ✓ Mauvaise identification des besoins
  - ✓ Les fonctionnalités et non fonctionnalités ne sont pas clairement décrites par les parties prenantes.
  - ✓ Problèmes d'exploitabilité et de maintenabilité par manque d'anticipation.
  - ✓ Communication entre le métier / la MOA et l'équipe projet. Les besoins ne sont pas connus.
  - ✓ Manque de méthode pour garantir la cohérence, la précision et la compréhension des exigences.
  - ✓ Changements durant le projet
  - ✓ Les évolutions ne sont pas correctement prises en compte.

# *La gestion des exigences*



# *La gestion des exigences*

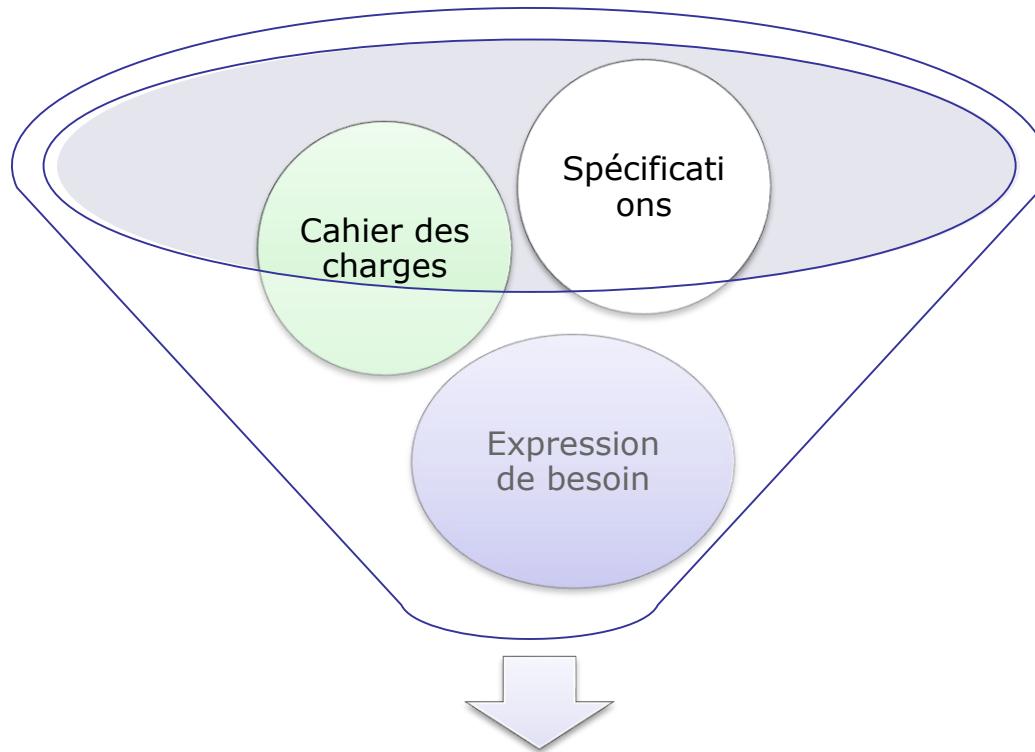
**Objectif :** Garantir la satisfaction du client et de l'équipe projet.  
S'assurer de la cohérence entre ce qu'il était attendu que l'application fasse et ce qu'elle fait réellement.

« **Gérer les exigences** » signifie:

- ✓ Identifier, clarifier et formaliser le besoin initial du client,
- ✓ Comprendre les exigences, qu'elles soient implicites ou explicites,
- ✓ Négocier pour prendre en compte des demandes de changement réalistes,
- ✓ Adapter le périmètre fonctionnel et non fonctionnel du projet en accord avec l'ensemble des parties prenantes,
- ✓ Fournir une meilleure compréhension des exigences à l'équipe projet,
- ✓ Vérifier la conformité du produit avec les exigences du client.

# *La gestion des exigences*

- Entrées nécessaire au commencement de la phase d'analyse des exigences :



Analyse des exigences

# Definition

Une exigence est une représentation du besoin de l'utilisateur

- Une condition ou une capacité nécessaire à un utilisateur pour résoudre un problème ou atteindre un objectif
- Une condition ou une capacité que doit posséder un système afin de satisfaire aux termes d'un contrat, d'une norme ou d'une spécification formellement imposée.
- La représentation documentée de cette condition ou capacité

Une spécification d'exigence comprend...

- Les fonctionnalités
- Les interfaces externes
- La performance
- Les attributs du système
- Les contraintes de conception

# Définition

**Exigence** : Condition ou capacité que doit présenter un système pour satisfaire un contrat, un standard, une spécification ou tout autre document formel imposé »

*Définition de la norme IEEE 729-1983*

L'exigence est donc

- ✓ un contrat entre un fournisseur et son client.
- ✓ doit être décrite sous la forme d'une action. Elle précise ce que l'on veut faire.

# Définition

## Besoin vs Exigence

**Besoin** : l'expression par un utilisateur d'un manque, d'une insatisfaction, d'une nécessité, d'un désir.



J'ai besoin de ...  
Je veux que ...  
Il me faut un ...



**Exigence** : une caractéristique à laquelle doit obligatoirement répondre la solution.



L'utilisateur PEUT [verbe d'action]...  
Le système DOIT [verbe d'action]...



# Catégories d'exigences

- Il existe deux catégories principales d'exigences :

## Exigences fonctionnelles

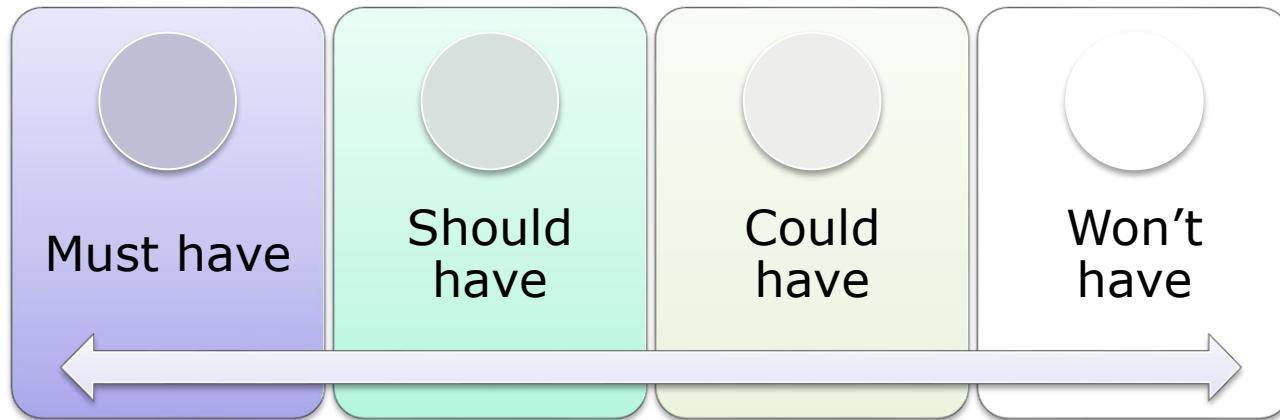
- A quoi sert le système
- Ce que doit faire le système, les fonctions utiles

## Exigences non fonctionnelles

- Performance, disponibilité, sécurité, portabilité...
- Chercher des critères mesurables

# Priorisation des exigences

- Toutes les exigences, fonctionnelles et non fonctionnelles doivent être classées et priorisées.
- Une exigence peut avoir une priorité haute dans le projet mais plus dans une itération.
- Technique de priorisation MoSCoW :



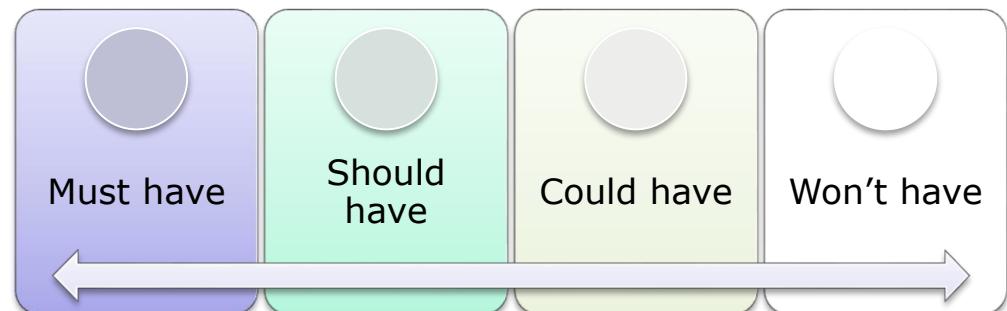
- *La gestion et l'analyse des exigences contribuent fortement à la qualité des produits.*

# Priorisation des exigences - Exercice

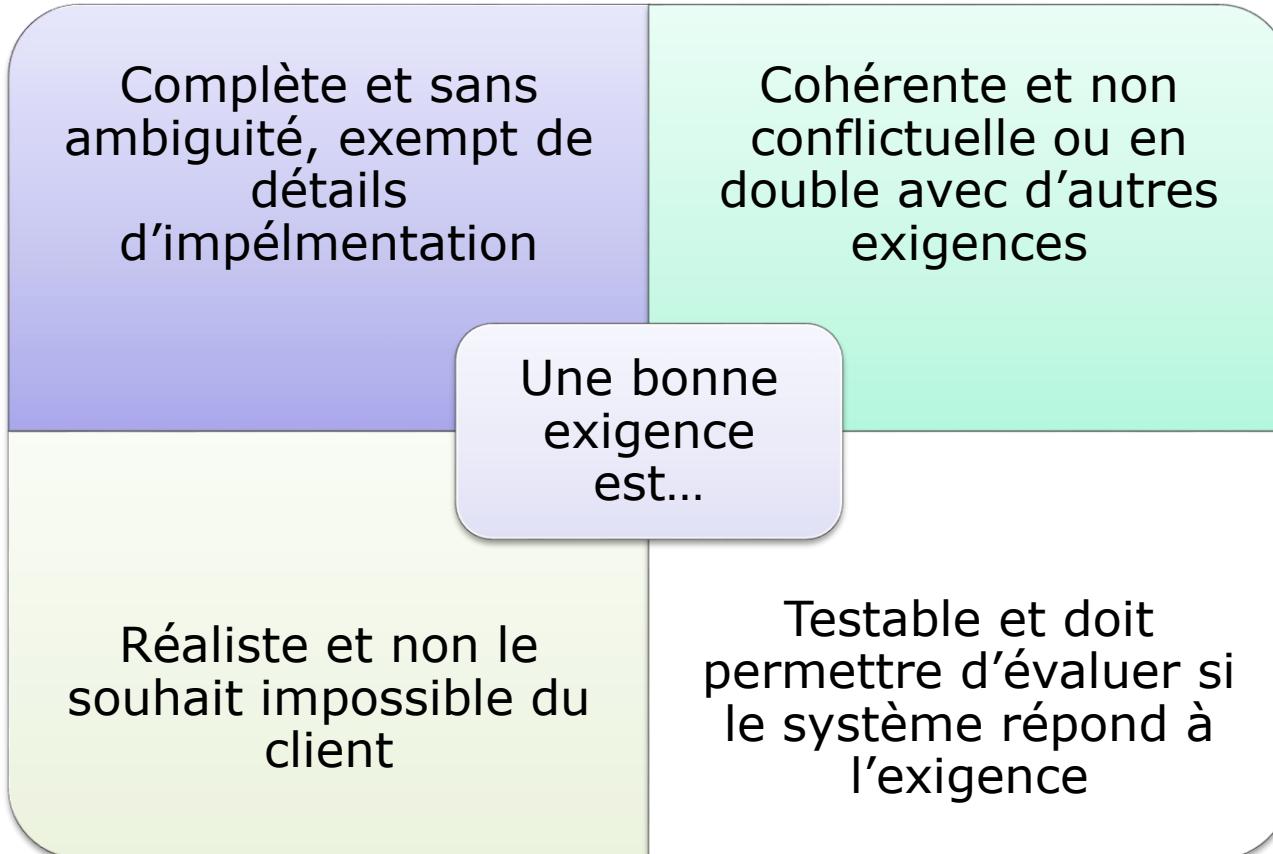
Soit un logiciel de gestion de tâches comprenant les fonctionnalités suivantes :

1. Les utilisateurs peuvent se connecter au logiciel.
2. Les utilisateurs doivent pouvoir demander un nouveau mot de passe par email s'ils l'ont oublié.
3. Les utilisateurs peuvent créer des tâches.
4. Un utilisateur peut envoyer un email au logiciel et cet email sera attaché à la bonne tâche.
5. Lorsqu'un utilisateur clique sur un n° de téléphone enregistré dans le logiciel, le numéro est automatiquement composé.

Priorisez les exigences  
selon la technique  
MoSCoW



# *Les attributs d'une bonne exigence*



# *Les attributs d'une bonne exigence*

Exemple d'exigence :

Dans l'Espace client, la catégorie de pollution du véhicule est calculée automatiquement à partir de la saisie l'euroclass et de la présence ou non du filtre anti-particules.

*En tant que <utilisateur>, je souhaite <faire quelque chose> afin de <atteindre un objectif>.*

# Revue d'exigence

## Clarification

- Une bonne compréhension des exigences est à la base de tout projet. S'il y a un manque de clarté, cela va à l'encontre du but du projet, et tous les efforts dépensés vont en vain.

Or, on ne peut pas s'attendre à ce que le client fournisse des exigences claires, complètes et sans ambiguïté.



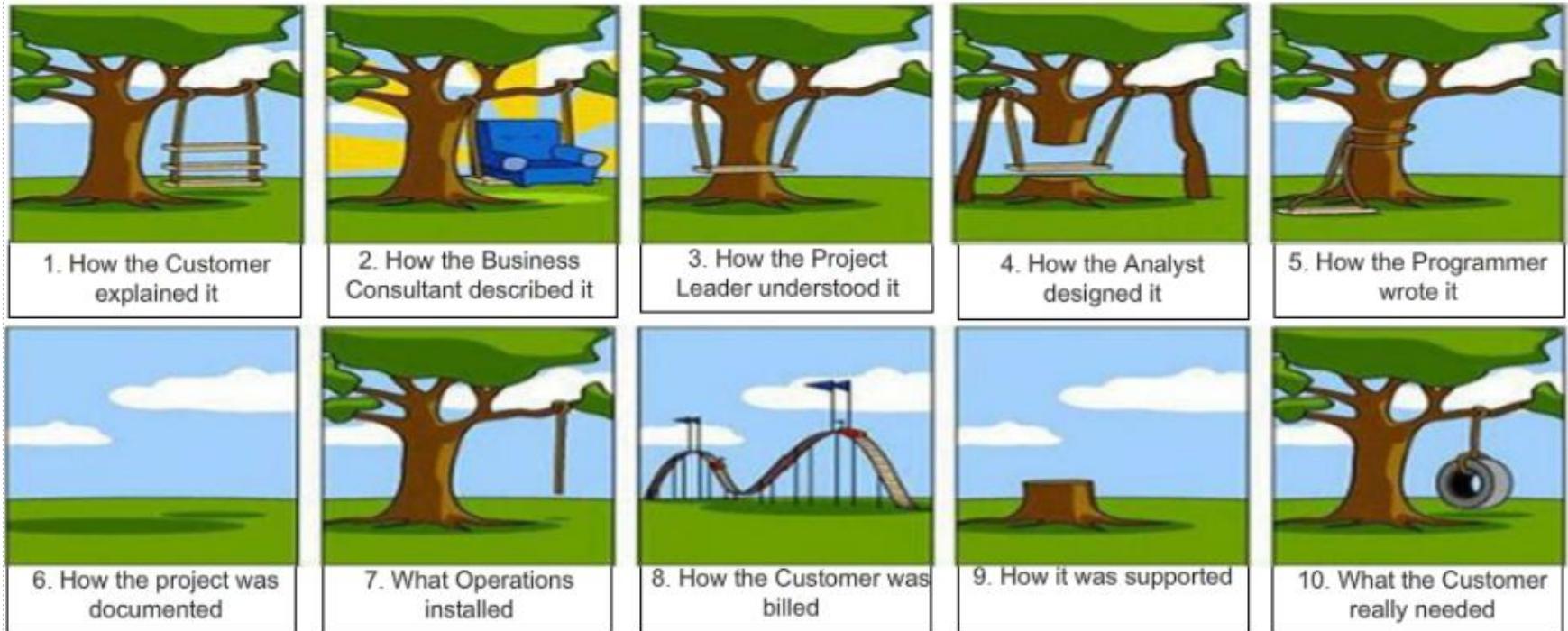
C'est pourquoi les clarifications sont essentielles au bon déroulement du projet.

## Définition

- Une clarification consiste à demander plus d'informations, plus de détails sur une exigence fournie par le client.

# Revue d'exigence

- La revue des exigences permet donc de confirmer que les exigences réunies sont claires et sans ambiguïté.
- C'est ce processus qui garantit qu'il n'y a pas de conflit à la fin du projet, sur ce qui a été fourni au client, par rapport à ce qui a été demandé par le client



# *Exigences et traçabilité*

Le lien entre la gestion des exigences et les tests prend son sens à travers la notion de **träçabilité**.

La traçabilité permet deux choses

- Une assurance de couverture
- Une analyse d'impact immédiate

## Couverture

- Toute exigence doit être couverte par au moins 1 test
- Chaque test doit au moins être lié à une exigence

## Analyse d'impact

- Le lien de traçabilité va lier toute anomalie à son exigence

# Sommaire

- Analyse des exigences
- **Estimation et planification des tests**
  - » Estimation des tests
  - » Introduction à la planification
  - » Contenu d'un plan de test
  - » Facteurs qui influencent la planification
  - » Activités de planification
- Couverture de test
- Conception des tests
- Techniques de conception de test
- Exécution des tests
- Finalisation de la recette

# *Estimation des tests basée sur des projets similaires*

## Pré requis

- Capitalisation des expériences sur les projets effectués afin d'établir une base de comparaison

## Comparaison par rapport à des caractéristiques principales

- Méthode de développement
- Environnement de développement
- Niveau d'expérience des développeurs
- ...

# *Principaux facteurs impactant l'estimation*

## Caractéristiques du produit

- Qualité des exigences et autres documents
- Taille du logiciel
- Complexité du domaine
- Exigences de fiabilité, sécurité...

## Processus de développement

- Stabilité de l'organisation
- Outils
- Processus de test
- Savoir-faire des personnes impliquées
- Contraintes de temps

## Résultats du test

- Nombre de défauts
- Volume de reprises exigées



# *Introduction*

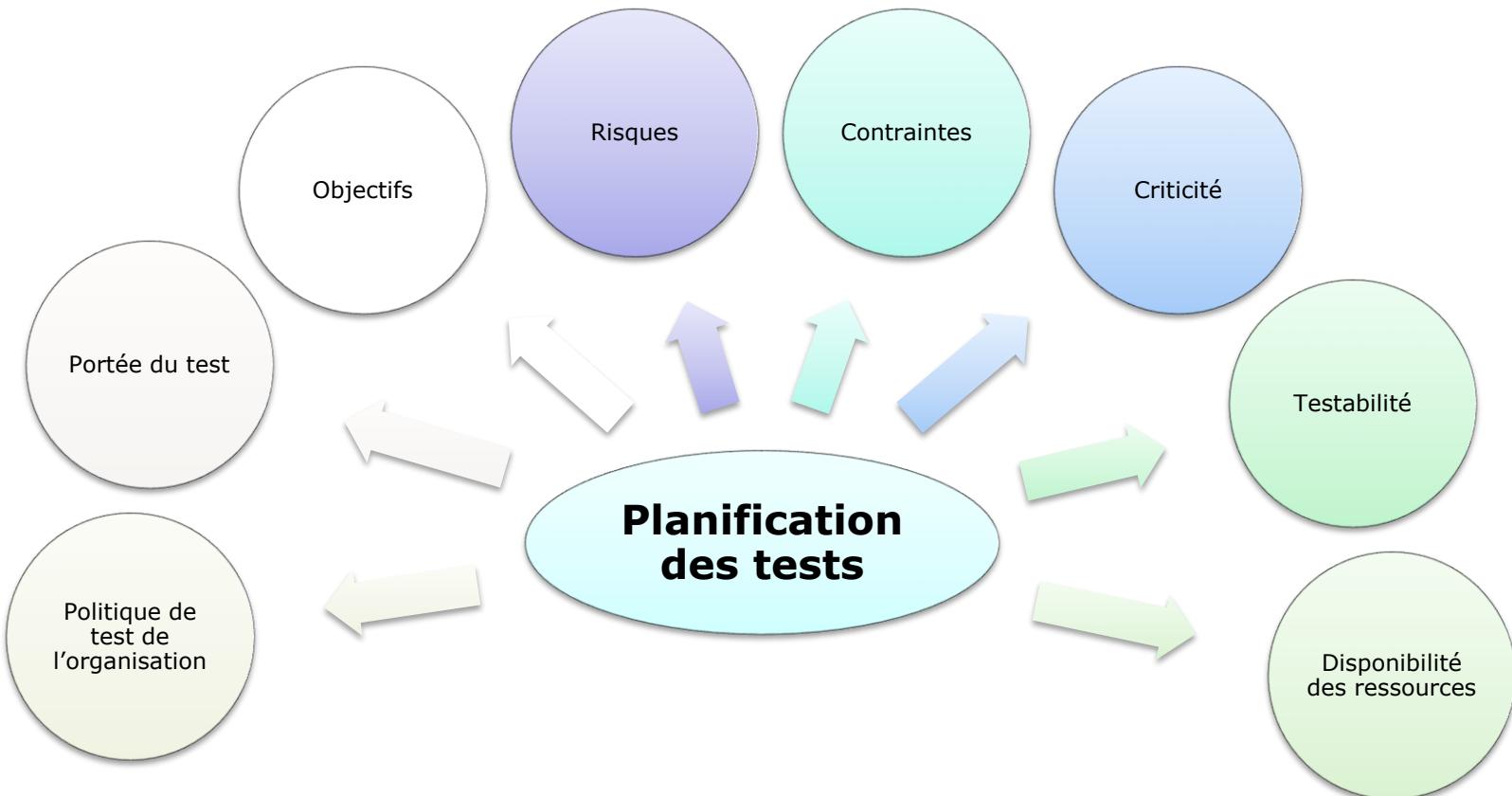
## Qu'est ce la planification ?

- Cette activité est celle qui est pratiquée au tout début de la phase de test.
- Elle permet de définir la stratégie qui sera mise en place tout au long de la phase de test.

# *Contenu d'un plan de test*

Objet	Définition
ID	Identifiant unique du document
Introduction	Résumé des éléments à tester, références aux autres documents
Périmètre de test	Fonctions à tester Eléments hors du périmètre de test
Approche	Stratégie à appliquer, préparation nécessaire, techniques utilisées pour la conception des tests
Critères d'acceptation ou de rejet	Critères définissant si l'élément à tester a passé le test avec succès ou pas
Livrables de test	Plans de test, résultats des tests...
Tâches	Tâches nécessaires pour la préparation et l'exécution des tests
Liste des environnements	Caractéristiques spécifiques essentielles et désirables des environnements de test, les outils, documentation...
Calendrier	Jalons de test et de transmission des éléments des tests
Risques	Risques liés à la phase de test et la gestion des risques
Approbation	Définir les personnes autorisées à valider les livrables intermédiaires et finaux et de valider la fin des tests

# Facteurs qui influencent la planification



# *Activités de planification des tests*

## Stratégie de test

- Définir l'approche générale du test

## Coordination

- Intégrer et coordonner des activités de test dans les activités du cycle de développement

## Plan de test

- Ce qui doit être testé, responsabilités, quand et comment ces activités doivent être exercées, comment évaluer les résultats des tests et quand arrêter les tests

## Ressources

- Assigner les ressources aux différentes tâches définies

## Documentation

- Définir le volume, le niveau de détail, la structure et les modèles pour la documentation du test

## Suivi et contrôle

- Sélectionner des mesures pour suivre et contrôler la préparation et l'exécution du test, l'élimination des défauts et la résolution des problèmes relatifs aux risques

## Niveau de détail

- Déterminer le niveau de détail pour les procédures de test (script de test) qui permet un exécution reproductible des tests

# Sommaire

- Analyse des exigences
- Estimation et planification des tests
- **Couverture de test**
  - » Définition
  - » Matrice de couverture
  - » Sur qualité et sous qualité
  - » Traçabilité des exigences
- Conception des tests
- Techniques de conception de test
- Exécution des tests
- Finalisation de la recette

# *Couverture de test*

Qu'est ce la couverture de test ?

- Cette activité consiste à s'assurer que les test effectués sont les bons
  - En quantité
  - En qualité

# Couverture de test

## Matrice de couverture

- Chaque exigence doit être au moins couverte par un test.
- Plus l'exigence est critique, plus les tests doivent être nombreux.

	Exigence A	Exigence B	Exigence C
Test A	X		X
Test B		X	
Test C			X
Test D	X	X	X

# Couverture de test

## Sous-qualité et sur qualité

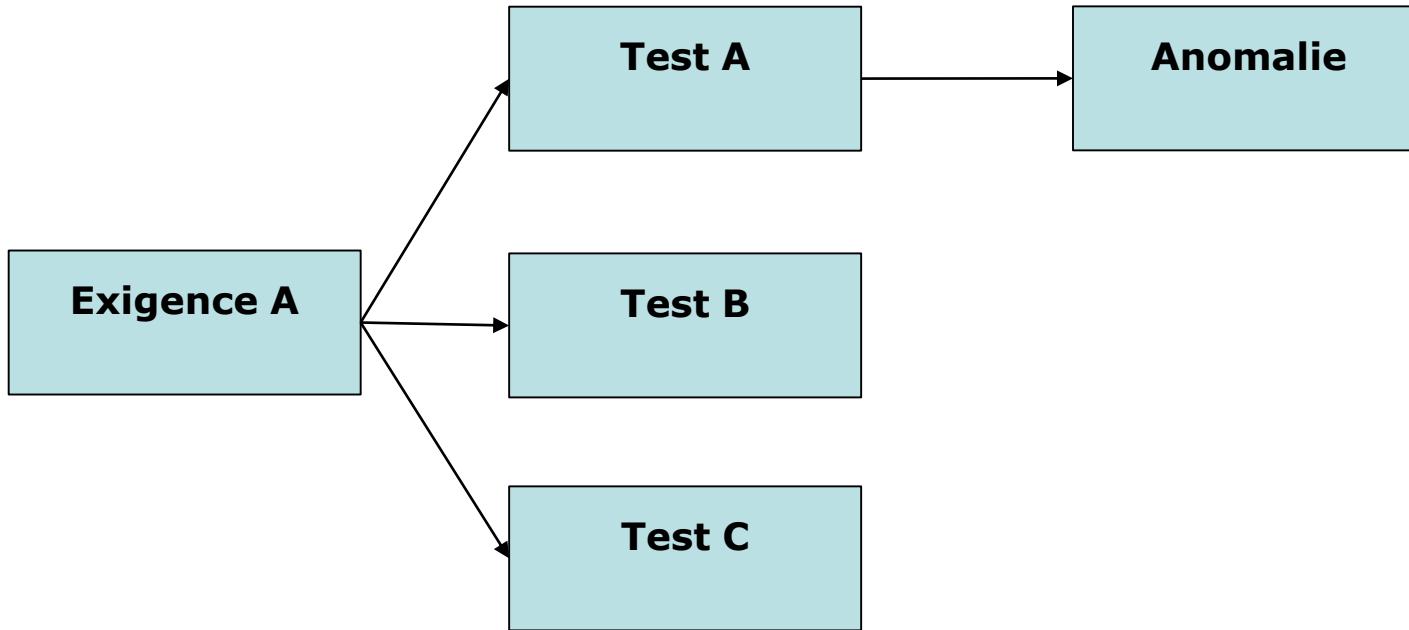
- Si une exigence n'est pas couverte, on parle de sous qualité
- Si un test ne couvre aucun exigence, on parle de sur qualité

	Exigence A	Exigence B	Exigence C
Test A			
Test B		X	
Test C			X
Test D		X	X

# Couverture de test

## La traçabilité des exigences

Ce sont les exigences qui vont permettre à tout instant de remonter au besoin initial.



# *Cahier de test*

## **Cahier de test**

Le cahier est un document qui permet de définir de façon non détaillée ce qui va être testé. Cela permet de valider une stratégie et une vision de la couverture des tests avant de lancer les opérations de création des tests.

## **Critères importants d'un test plan**

- Nom du scenario
- Nom du test
- Description du test
- Prérequis du test
- Complexité du test
- Exigence associée
- Commentaires

# Exemple de cahier de test

M&#xA7;	ID	Catégorie / Fonctionnalité	Description du cas de test	Prérequis / Jeux de données	Reuse	Complexité	Enigences	Commentaires Cognizant
061	XNET_	Liste des badges						
	001	Liste des badges - TD DEU		- Un client ayant un contrat avec le TD allemand - Le client doit avoir des badges actifs avec le TD DEU activé				
001	XNET_001	Liste de badges - Affichage badge ayant le TD DEU	Vérifier que la liste des badges d'un client qui possède des OBU activé avec le TD allemand s'affiche correctement. Récupérer le un badge actif ayant le TD allemand et vérifier qu'il apparaît bien dans la liste des badges actifs		New	S		
	001	Détail des badges						
	001	Affichage nouveaux champs d'identification de véhicule - Consultation		- Un client ayant un contrat avec le TD allemand - Le client doit avoir des badges actifs avec le TD DEU activé				
002	XNET_002	Consultation badge	Se rendre sur XNET et consulter les information d'un badge depuis la liste des badges en cliquant sur le N° d'un badge		New	S		
003	XNET_003	Affichage Champ Cop Value	Vérifier que le champ Cop value est bien présent et qu'il contient les bonnes information.		New	S	EXT-MET-BAG-006 EXT-MET-BAG-007 EXT-MET-BAG-008	
004	XNET_004	Affichage Champ V.I.N	Vérifier que le champ V.I.N est bien présent et qu'il contient les bonnes information		New	S	EXT-MET-BAG-009 EXT-MET-BAG-010 EXT-MET-BAG-011	

# Sommaire

- Analyse des exigences
- Estimation et planification des tests
- Couverture de test
- **Conception des tests**
  - » Introduction
  - » Identification et création des scénarios
  - » Bonnes pratiques
  - » Priorisation des scénarios
  - » Préparation des jeux de données
  - » Préparation de l'environnement de test
- Techniques de conception de test
- Exécution des tests
- Finalisation de la recette

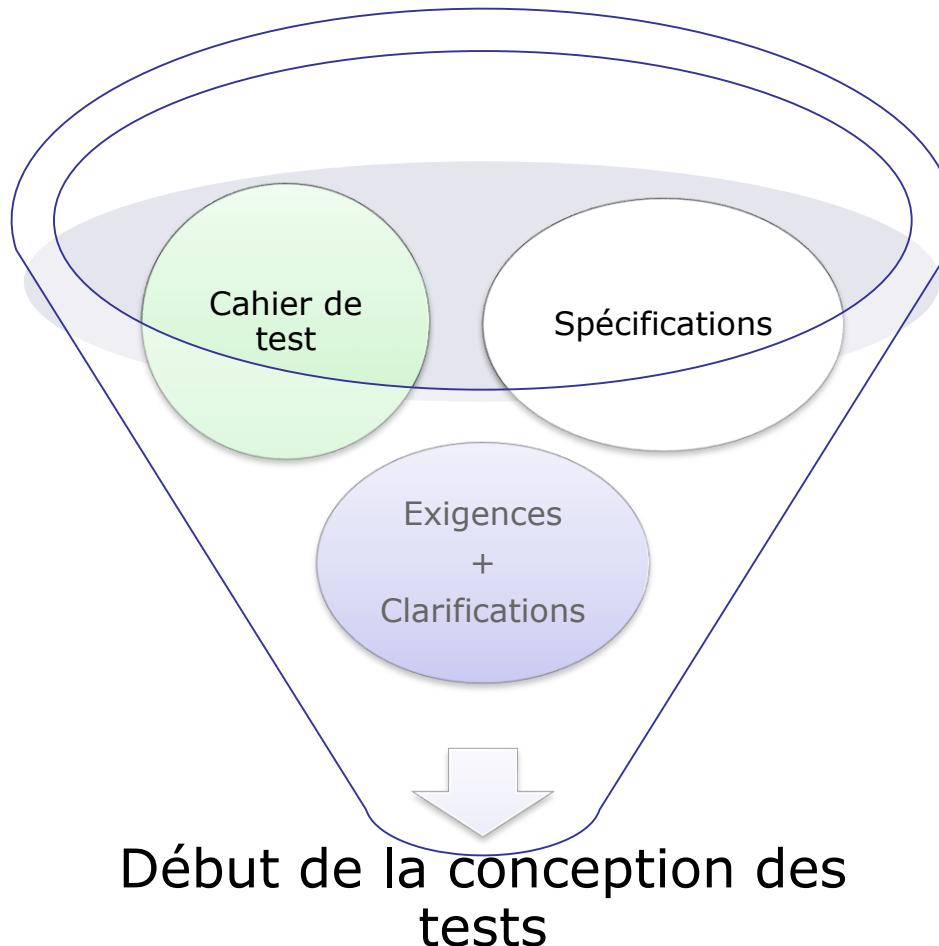
# *Introduction*

## Qu'est ce la conception ?

- Cette activité consiste en la rédaction des tests qui seront joués.
- Elle définit pour chaque test à exécuter
  - Les prérequis à posséder pour effectuer le test
  - Les actions qu'il faudra mener
  - Les résultats auxquels on s'attend

# *Introduction*

- Entrées nécessaire au commencement de la phase de conception :



# *Identification et création des scénarios*

- Pourquoi créer des scénarios ?

Différentes combinaisons fonctionnelles possibles & différentes conditions



Différentes fonctionnalités

- Exemple :

Connexion

Fonctionnalités de connexion

Fonctionnalité de réinitialisation du mot de passe

Page d'accueil

Fonctionnalité de recherche



Cognizant

Passion for building stronger businesses

# Cas de tests – Les questions à se poser

- 5 questions obligatoires à se poser :

Que veut-on tester ?

- « Je veux tester la fonctionnalité qui permet de... »

Dans quel cas ?

- « Je veux tester la fonctionnalité qui permet de [...] lorsque le système est dans l'état... »

Que dois-je obtenir ?

- « Lorsque le système est dans cet état et que j'exécute cette fonctionnalité, le résultat que je dois obtenir est... »

Quel déroulement ?

- « Pour exécuter complètement la fonctionnalité, de dois réaliser... »

Quelles sont les opérations préliminaires à effectuer ?

- « Que dois-je faire initialement pour être en situation d'exécuter cette fonctionnalité ? »

# *Identification et création des scénarios*

VS

## Scénario

Description de haut niveau ou condition relative à la fonctionnalité de l'application

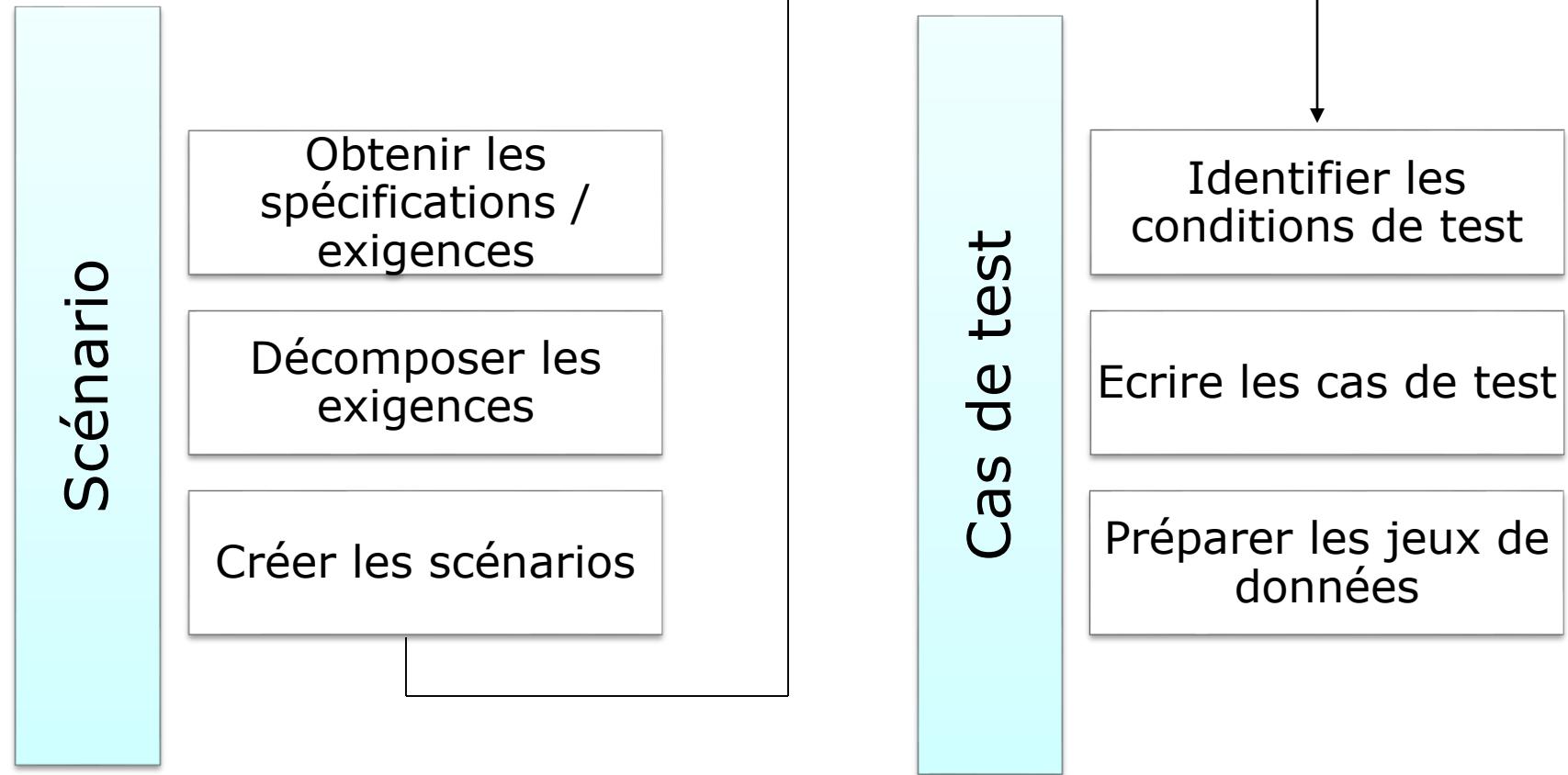
Peut être divisé en un ensemble de cas de tests couvrant les possibilités de tester un ensemble de conditions dans son intégralité (au maximum)

## Cas de test

Un cas de test est destiné à effectuer un test tel qu'indiqué dans un seul but pour tester une fonctionnalité

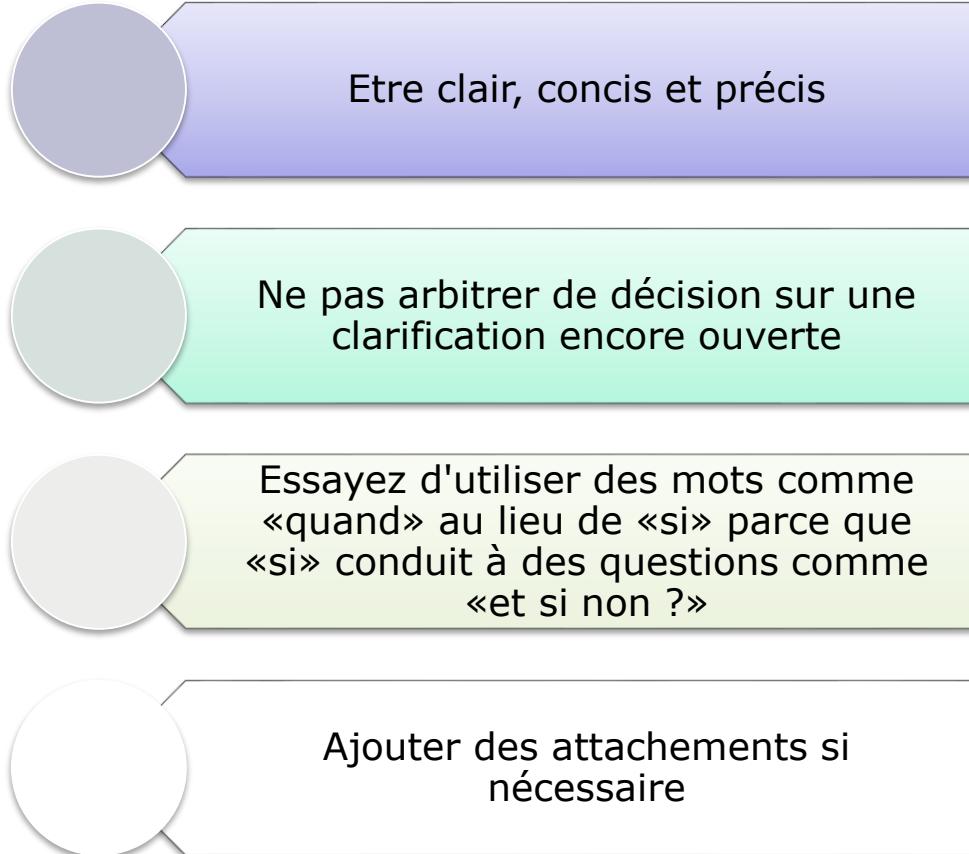
Contient les étapes pour exécuter un scénario particulier et inclut les données pour accéder au résultat attendu et le comportement réel de l'application

# *Identification et création des scénarios*



# Bonnes pratiques

- Bonnes pratiques à suivre lors de l'identification des scénarios



# Priorisation des scénarios

## Qu'est ce la priorisation ?

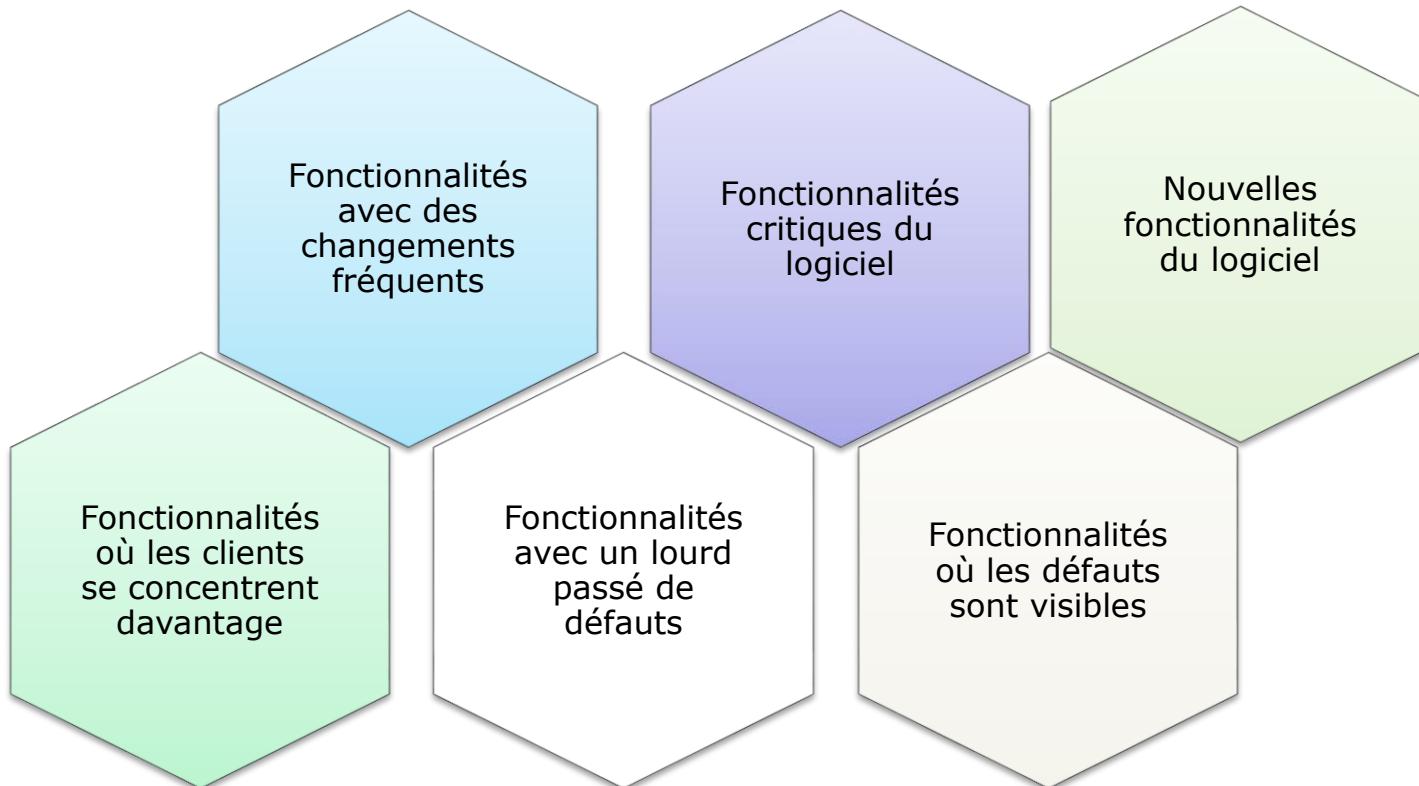
- Ranger les scénarios identifiés par ordre d'importance ou d'urgence est appelé Priorisation des scénarios.

## Pourquoi est-ce important ?

- A cause de la complexité d'un logiciel, le test exhaustif est impossible.
- Il est nécessaire de hiérarchiser les scénarios afin d'assurer la qualité du logiciel en tenant compte des facteurs tels que les exigences critiques du client, son coût et son temps.
- La priorisation aide à augmenter l'efficacité du test pour atteindre un objectif de performance.

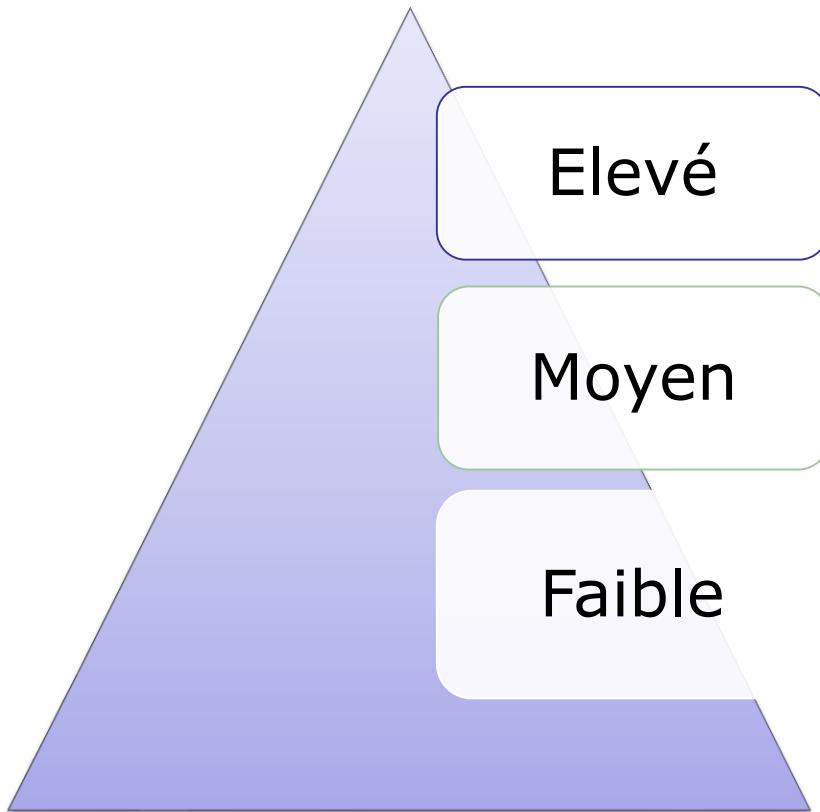
# Priorisation des scénarios

- Quels sont les facteurs à considérer pour prioriser les scénarios ?



# Priorisation des scénarios

- Niveaux de priorisation :



- Scénarios couvrant les fonctionnalités critiques
- Scénarios couvrant les fonctionnalités qui ont un impact modéré sur le logiciel
- Scénarios couvrant les fonctionnalités qui ont un faible impact sur le logiciel

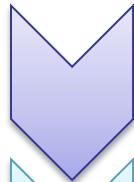
# *Préparation des jeux de données*

- Les jeux de données font partie intégrante des tests. Elaborer un jeu de données demande une connaissance fonctionnelle, aussi bien sur la nature des données que sur le scénario de test envisagé.

## Importance du jeu de données

- Pour une recette fonctionnelle optimale, il est important de disposer de données « réelles », c'est à dire représentatives de ce qui sera rencontré en production.
- Cette étape n'est pas aussi simple qu'elle n'y paraît, et est souvent négligée.

# Préparation de l'environnement de test



- S'informer auprès du client des besoins matériels et logiciels



- Obtenir les accès appropriés à l'environnement



- S'assurer que l'environnement et sa configuration sont similaires à ceux de production



- S'assurer que l'environnement soit isolé de celui dédié au développement



- S'assurer que les jeux de données utilisés soient similaires à ceux utilisés par les développeurs



- Etre tenu informé des livraisons prévues sur l'environnement de test

# Sommaire

- Analyse des exigences
- Estimation et planification des tests
- Couverture de test
- Conception des tests
- **Techniques de conception de test**
  - » Techniques boîte noire
  - » Techniques boîte blanche
  - » Techniques statiques
  - » Techniques basées sur l'expérience
  - » Formalisme des cas de test
- Exécution des tests
- Finalisation de la recette

# *Techniques de conception de tests*

Différentes techniques de conception sont utilisées pour répondre aux questions :

- Quels cas de tests ?
- Combien de cas de tests ?
- Comment identifier les cas de tests ?
- Comment réduire l'effort de test ?

# *Techniques de conception de tests*

## 3 grandes catégories de techniques de conception

### Techniques de conception boîte noire

- Conception basée sur les spécifications

### Techniques de conception boîte blanche

- Conception basée sur la structure

### Techniques de conception basées sur l'expérience

- Conception basée sur l'expertise

# *Techniques de conception de tests*

Les techniques de type boite noire sont les plus communément utilisées. Ce sont les techniques dites de test fonctionnel

Les techniques de type boite blanche sont utilisées en complément des techniques boite noire. Elles permettent d'améliorer la couverture et s'appliquent particulièrement aux systèmes critiques ou complexes.

Les autres techniques : statiques ou basées sur l'expérience sont plus de l'ordre de la bonne pratique et moins documentées.

# *Techniques Boite Noire*

## **Exemples de techniques Boîte Noire**

- Test de cas d'emploi
- Partitions d'équivalence
- Analyse des valeurs limites
- Test par table de décision
- Test par transition des états

# *Partitions d'équivalence*

## Définition

- Une partition d'équivalence est un ensemble de valeurs d'entrée bornées occasionnant un comportement identique d'un composant ou système.

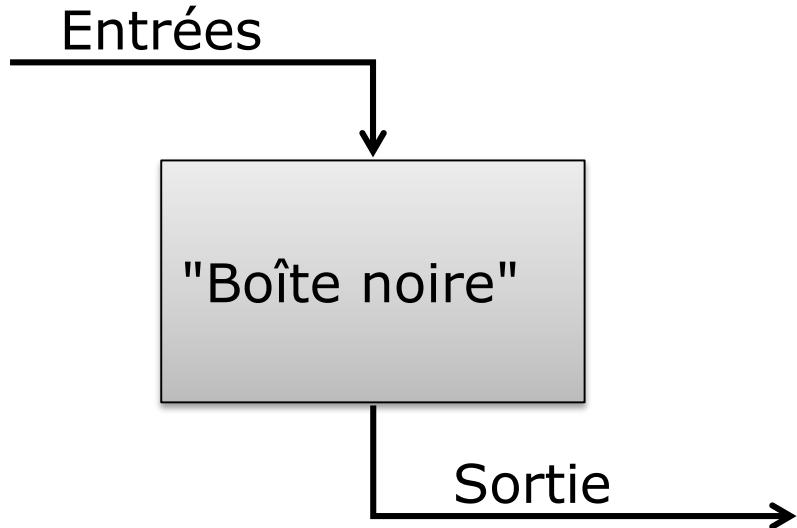
## Caractéristiques

- Les partitions (ou classes) d'équivalence peuvent être trouvées pour des données valides et des données invalides, c'est à dire des valeurs qui devraient être rejetées.

## Utilisation

- Les partitions d'équivalence sont applicables à tous les niveaux de tests.

# *Partitions d'équivalence* Exemple

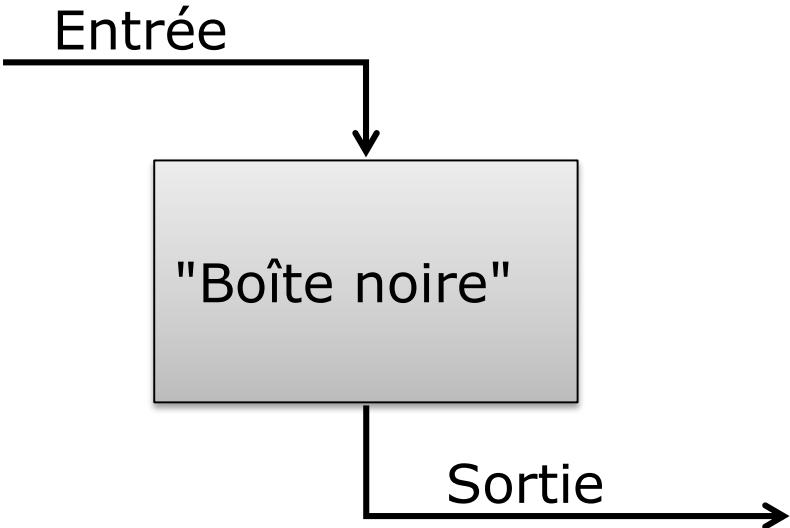


16 combinaisons d'entrées différentes

Combien y a-t-il de classes d'équivalences ?

Entrée				Sortie
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	2
1	0	1	1	2
1	1	0	0	2
1	1	0	1	2
1	1	1	0	2
1	1	1	1	n/a

# Partitions d'équivalence Exemple



16 combinaisons d'entrées différentes

4 classes d'équivalences :

- 3 cas de tests valides,
- 1 cas de test invalide.

Entrée	Sortie
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	2
1 0 1 1	2
1 1 0 0	2
1 1 0 1	2
1 1 1 0	2
1 1 1 1	n/a



# *Partitions d'équivalence* Exercice

## Enoncé

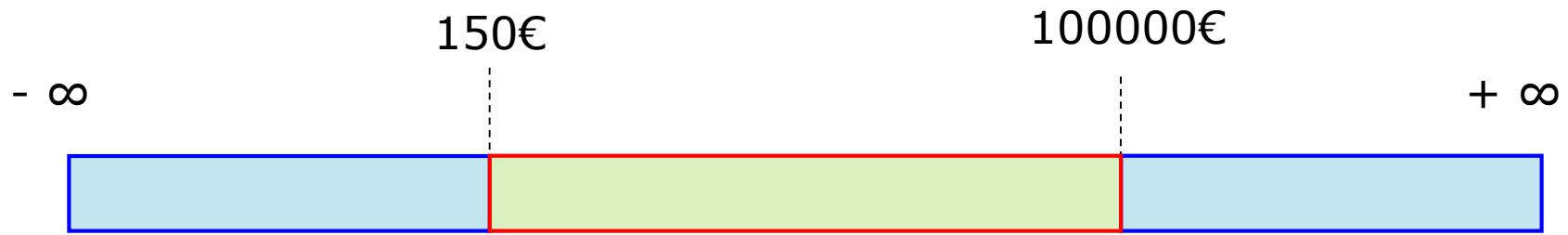
- » L'alimentation initiale sur un compte d'épargne doit être :
  - " $\geq 150\text{€}$ ",
  - " $\leq 100\ 000\text{€}$ ".

## Questions

- » Combien y a-t-il de classe d'équivalence ?
- » Combien de cas de tests faut-il pour couvrir l'exigence ?

# Partitions d'équivalence Exercice

## Réponse



2 classes d'équivalences :

- 1 cas de tests valides,
- 2 cas de test invalide.

# *Analyse des valeurs limites*

## Définition

- Une valeur d'entrée ou de sortie qui est au bord d'une partition d'équivalence, ou à la distance minimale d'un incrément de chaque côté de cette limite, par exemple le minimum ou le maximum d'une plage de valeurs.

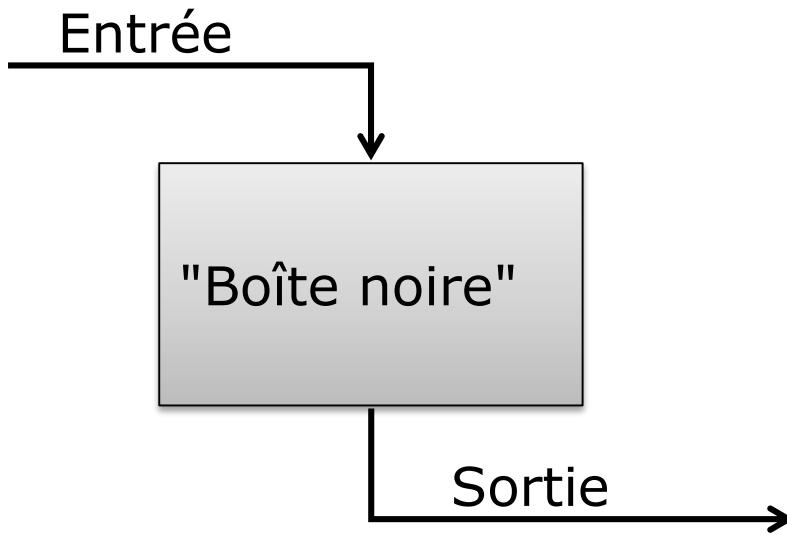
## Caractéristiques

- Une valeur limite pour une partition valide est une valeur limite valide; la limite d'une partition invalide est une valeur limite invalide.
- Des tests peuvent être conçus pour couvrir les valeurs limites valides et invalides. Quand on conçoit des cas de tests, une valeur de chaque limite est sélectionnée.

## Utilisation

- L'analyse des valeurs limites peut être appliquée à tous les niveaux de tests.

# Analyse des valeurs limites Exemple

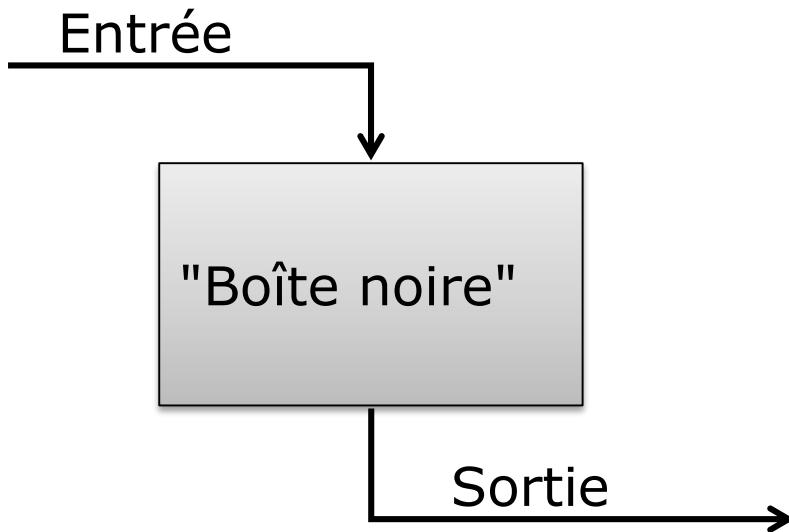


16 combinaisons d'entrées différentes

- 4 classes d'équivalences
- combien de cas de tests aux limites ?

Entrée				Sortie
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	2
1	0	1	1	2
1	1	0	0	2
1	1	0	1	2
1	1	1	0	2
1	1	1	1	n/a

# Analyse des valeurs limites Exemple



16 combinaisons d'entrées différentes

- 4 classes d'équivalences
- 7 cas de tests aux limites

Entrée	Sortie
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	2
1 0 1 1	2
1 1 0 0	2
1 1 0 1	2
1 1 1 0	2
1 1 1 1	n/a

# Analyse des valeurs limites Exercice

## Enoncé

- » L'alimentation initial sur un compte d'épargne doit être  $> 150\text{€}$  et  $\leq 100\,000\text{€}$  »

Classe valide	Classes invalides
$[ 150 ; 100.000]$	$] -\infty ; 150[$
	$] 100.000 ; +\infty [$

## Questions

- » Quelles sont les valeurs limites valides et invalides ?

# Tableau de Karnaugh

## Définition

- Tableau montrant la combinaison des entrées (conditions ou causes) pour chaque sortie (actions ou effets) possible, utilisé pour concevoir des cas de tests.

## Caractéristiques

- La couverture utilisée avec les tableaux de Karnaugh est d'avoir au moins un test par regroupement de combinaisons, ce qui implique que toutes les combinaisons de conditions de déclenchement sont couvertes.

## Utilisation

- Cela peut être appliqué à toutes les situations quand les actions du logiciel dépendent de plusieurs décisions logiques, notamment pour les règles de gestion complexes.

# Tableau de Karnaugh Exemple

a	b	c	d	s
0	0	0	0	1
0	0	0	1	1
0	0	1	0	
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	
0	1	1	1	
1	0	0	0	1
1	0	0	1	1
1	0	1	0	
1	0	1	1	
1	1	0	0	1
1	1	0	1	1
1	1	1	0	
1	1	1	1	1



- 10 combinaisons → 10 cas de tests
- $S = \neg a \cdot \neg b \cdot \neg c \cdot \neg d + \neg a \cdot \neg b \cdot \neg c \cdot d + \neg a \cdot \neg b \cdot c \cdot d + \neg a \cdot b \cdot \neg c \cdot \neg d + \neg a \cdot b \cdot \neg c \cdot d + a \cdot \neg b \cdot \neg c \cdot \neg d + a \cdot \neg b \cdot \neg c \cdot d + a \cdot b \cdot \neg c \cdot \neg d + a \cdot b \cdot \neg c \cdot d + a \cdot b \cdot c \cdot d$
- On peut donc essayer de réduire l'effort de test par l'utilisation du tableau de Karnaugh

# Tableau de Karnaugh Exercice 1

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	0	1	1	0
	1 1	0	1	1	0
	1 0	1	0	0	1

• S = ?

# Tableau de Karnaugh Exercice 1

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	0	1	1	0
	1 1	0	1	1	0
	1 0	1	0	0	1

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	0	1	1	0
	1 1	0	1	1	0
	1 0	1	0	0	1

- Rassemblement vert :  $b \cdot d$
- Rassemblement bleu :  $\neg b \cdot \neg d$
- $S = b \cdot d + \neg b \cdot \neg d$

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	0	1	1	0
	1 1	0	1	1	0
	1 0	1	0	0	1

# Tableau de Karnaugh Exercice 2

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	1	1	1	1
	1 1	1	0	0	0
	1 0	0	0	0	0

•  $S = ?$

# Tableau de Karnaugh Exercice 2

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	1	1	1	1
	1 1	1	1	0	0
	1 0	0	0	0	0

- Rassemblement vert :  $\neg a \cdot d$
  - Rassemblement bleu :  $\neg c \cdot d$
  - Rassemblement rouge :  $\neg b \cdot \neg c$
- $$S = \neg a \cdot d + \neg c \cdot d + \neg b \cdot \neg c$$

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	1	1	1	1
	1 1	1	1	0	0
	1 0	0	0	0	0

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	1	1	1	1
	1 1	1	1	0	0
	1 0	0	0	0	0

S		a b			
		0 0	0 1	1 1	1 0
c d	0 0	1	0	0	1
	0 1	1	1	1	1
	1 1	1	1	0	0
	1 0	0	0	0	0

# *Tests par table de décision*

## Définition

- Tableau montrant la combinaison des entrées (conditions ou causes) et de leurs sorties (actions ou effets) associées, utilisé pour concevoir des cas de tests.

## Caractéristiques

- La couverture utilisée avec les tables de décisions est d'avoir au moins un test par colonne, ce qui implique que toutes les combinaisons de conditions de déclenchement sont couvertes.

## Utilisation

- Cela peut être appliqué à toutes les situations quand les actions du logiciel dépendent de plusieurs décisions logiques, notamment pour les règles de gestion complexes.

# Tests par table de décision

Exemple avec l'assurance tout risque d'une moto: ces trois causes avec deux valeurs possible (Oui/Non) donnent huit combinaisons de test ( $2^3$ ).

Causes : questions (Oui/Non)							
Accident responsable ces 2 dernières années						O O O O	N N N N
Jeune conducteur						O O N N O O	N N N N
Age de la moto > 5 ans						O N O N O N	O N O N
Effets							
Msg : refus d'assurance tout risque						X X X X	
Msg : Agrément d'assurance Prix normal							X
Msg : Agrément d'assurance Prix réduit						X X	X X

Causes : questions (Oui/Non)			
Accident responsable ces 2 dernières années	O	N	N
Jeune conducteur	-	O	O
Age de la moto	-	O	N
Effets			
Msg : refus d'assurance tout risque	X		
Msg : Agrément d'assurance Prix normal		X	
Msg : Agrément d'assurance Prix réduit		X	X

Après réduction,  
les huit possibilités de test sont  
simplifiées à 4 cas de tests.



# Tables de décision Exercice 1

## Enoncé

- Pour pouvoir débloquer sa participation, il faut :
  - que le capital ait fructifié pendant 5 ans,
  - ou avoir acheté une résidence principale,
  - ou s'être marié.

## Questions

- Lister les causes et les effets
- Calculer le nombre de combinaisons possibles
- Remplir les colonnes avec toutes les combinaisons possibles
- Réduire la table de décisions
- Identifier le nombre de cas de tests passant et non passant

# Tables de décision Exercice 2

## Enoncé

- Pour être remboursé partiellement (60%), un médicament :
  - doit être prescrit par un médecin
  - doit figurer à la liste des médicaments remboursables,
- Pour être remboursé à 100 %, le client doit être adhérent d'une mutuelle.
- Une mutuelle prend en charge partiellement (40%) les médicaments prescrits non remboursables.

## Questions

- Lister les causes et les effets
- Calculer le nombre de combinaisons possibles
- Remplir les colonnes avec toutes les combinaisons possibles
- Réduire la table de décisions
- Identifier le nombre de cas de tests passant et non passant

# *Tests de cas d'utilisation (use case)*

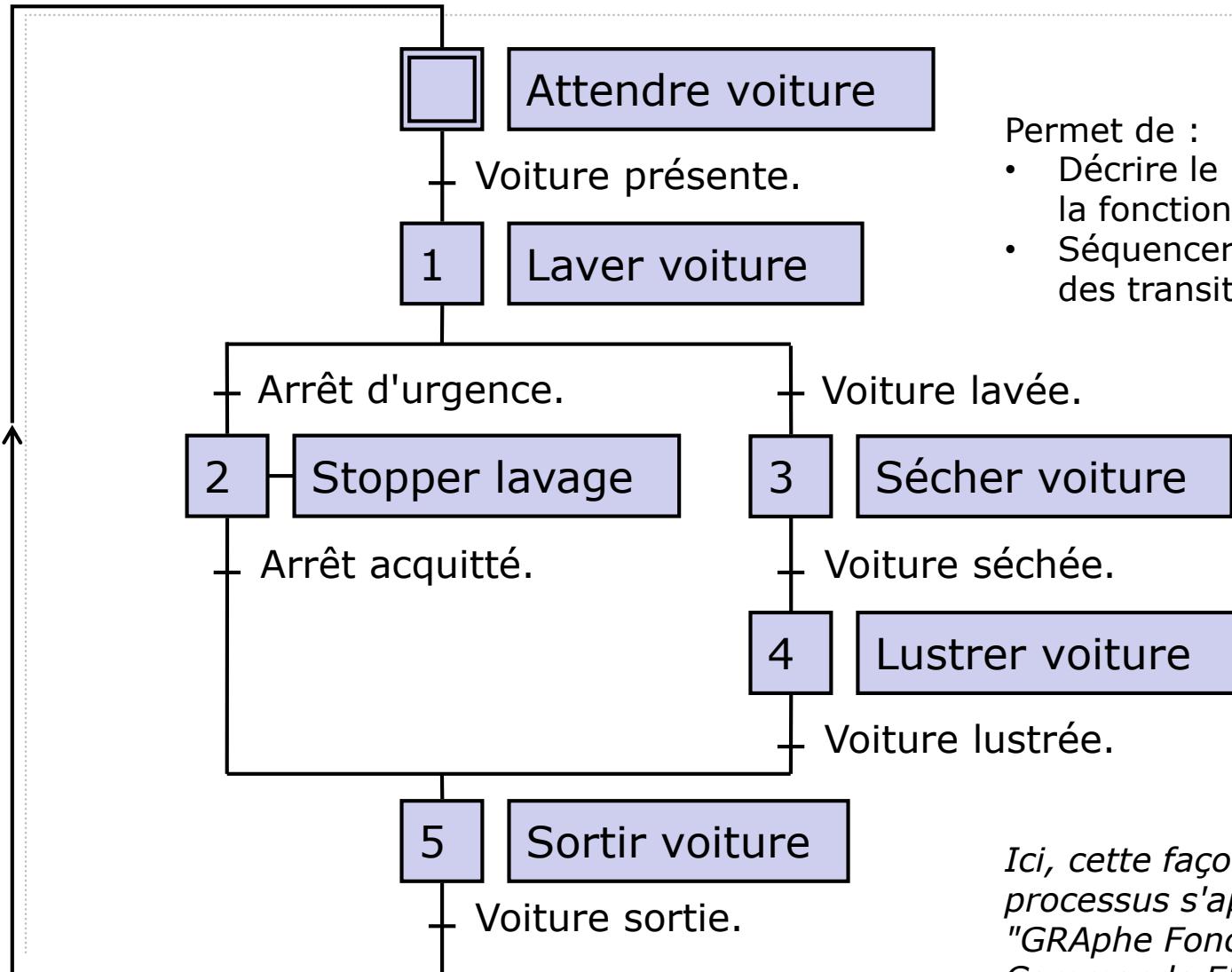
## Définition

- Un cas d'utilisation définit une manière d'utiliser le système et permet d'en décrire les exigences fonctionnelles
- Chaque cas d'utilisation a des pré-conditions et se termine par des post-conditions, qui sont les résultats observables et l'état final du système après la fin de l'exécution du cas d'utilisation. Un cas d'utilisation a généralement un cas de test dominant (c'est à dire le plus plausible), et parfois des branches alternatives.

## Utilisation

- Ils permettent de découvrir des défauts d'intégration causés par l'interaction et les interférences entre divers composants, ce que ne découvrent pas les tests individuels de composants.

# Tests de cas d'utilisation



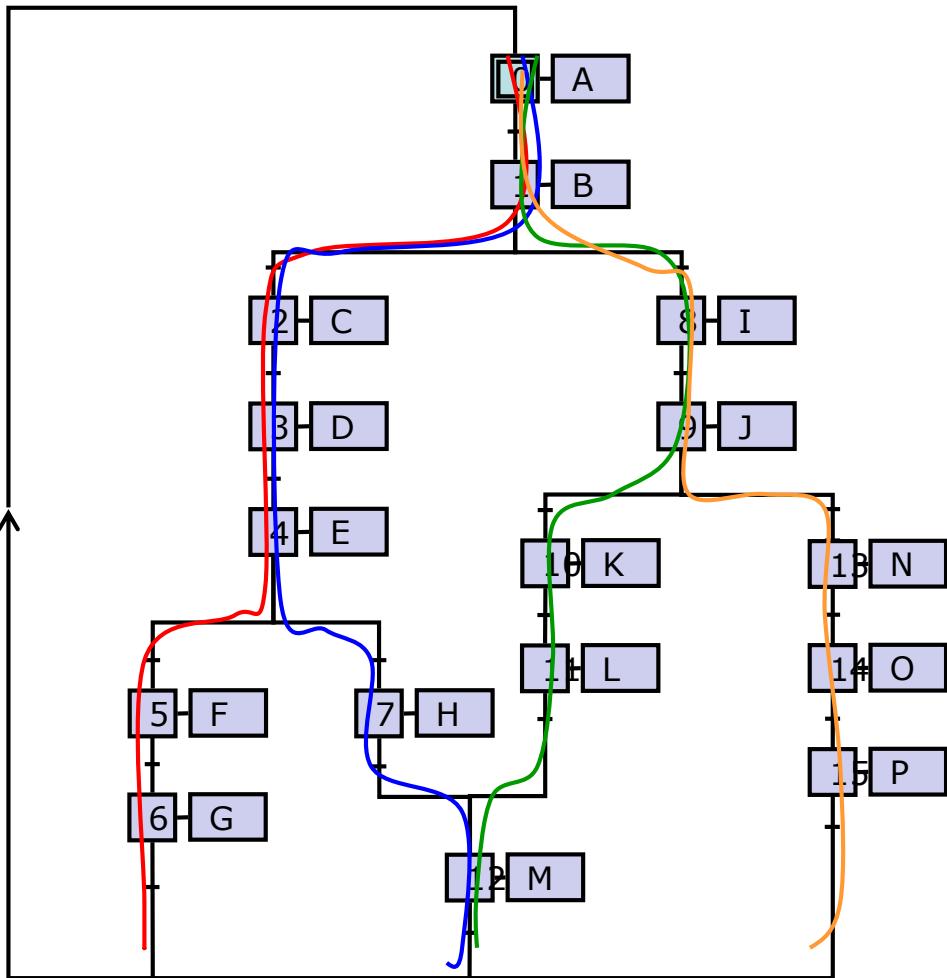
Permet de :

- Décrire le processus métier de la fonctionnalité à qualifier.
- Séquencer les états stables par des transitions logiques.

Ici, cette façon de décrire le processus s'appelle un Grafcet : "GRAphe Fonctionnel de Commande Etapes/Transitions"



# Tests de cas d'utilisation



Définir les cas d'emploi de façon à parcourir le processus sans oublier une branche, tout en faisant le minimum de tests.

4 cas d'emploi = 4 cas de test :

- cas 1
- cas 2
- cas 3
- cas 4

# Tests de cas d'utilisation Exercice

## Enoncé :

→ Achat / livraison pour un site marchand

Trois acteurs sont en relation dans un processus d'achat / livraison d'un site marchand : le client, le service des ventes, le service des stocks. En fonction de l'état du processus, les acteurs peuvent être des individus humains ou des systèmes informatiques.

Les tests du processus d'achat demandent de prendre en compte les relations entre ces entités. Il faudra alors identifier la totalité des processus qui sont partagés entre elles avant d'identifier les cas de tests suffisant à concevoir.

Après recueil auprès des futurs utilisateurs du workflow et analyse des spécifications, vous avez identifié le processus (cas de fonctionnement nominal) et quatre cas d'exception (comportements alternatifs) :

## Question :

- Présenter le processus de cas d'emploi sous la forme d'un "Grafcat"
- Identifier les cas de test nécessaires et suffisants aux tests du processus complet.

## Processus du site :

### Cas nominal :

1. Le client choisit ses articles,
2. Le service des ventes constitue le panier,
3. Le service des stocks contrôle les quantités disponibles,
4. Les ventes calculent le prix total,
5. Le client valide sa commande,
6. Les ventes valident le paiement,
7. Les stocks livrent la commande,
8. Le client accepte la livraison.

### Cas d'exception 1 : "Les quantités disponibles sont insuffisantes" :

- Le client refait le choix de ses articles.

### Cas d'exception 2 : "Le paiement est refusé par le service des ventes" :

- Le client refait la validation de la commande.

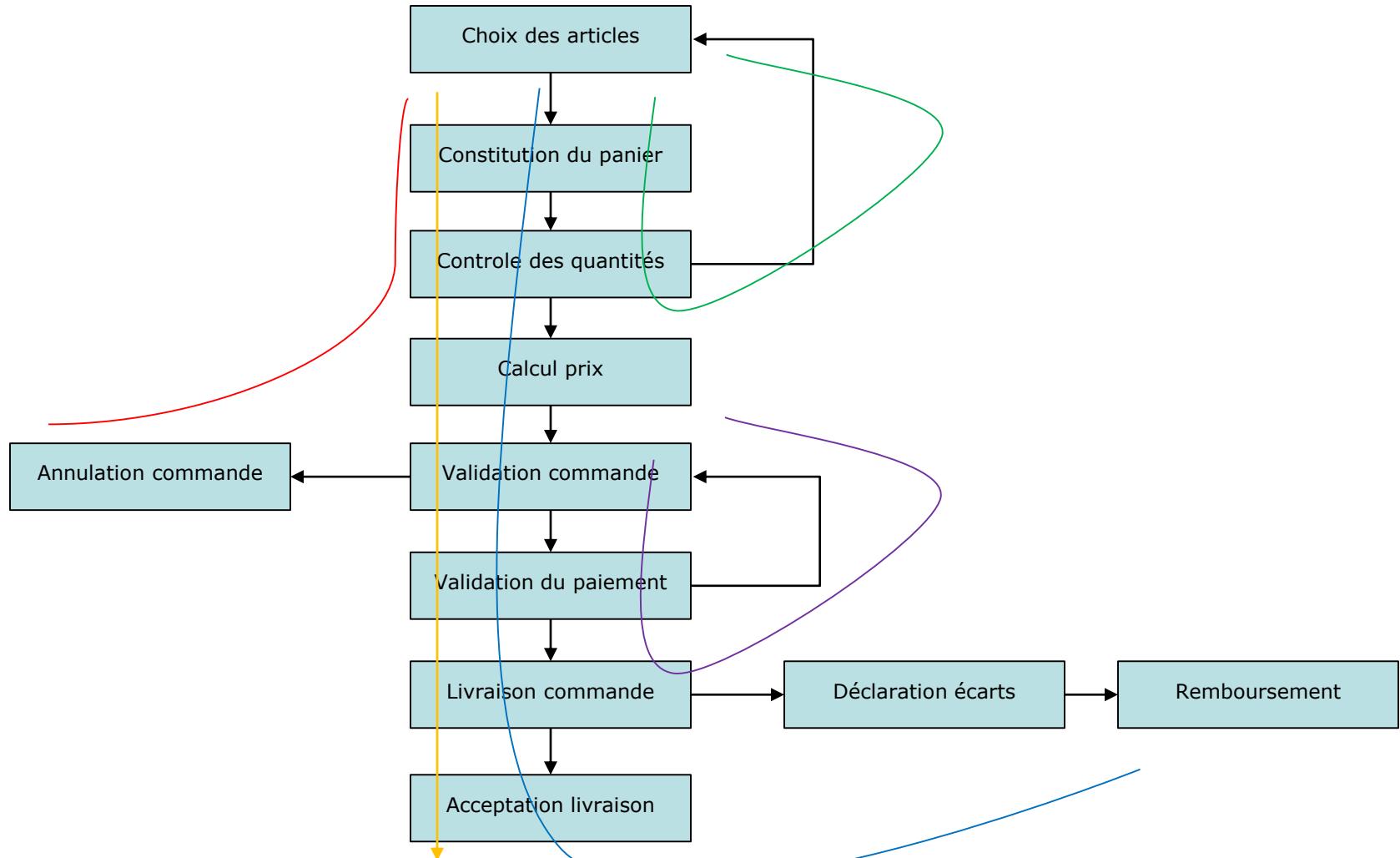
### Cas d'exception 3 : "Le client annule sa commande" :

- Le client ne valide pas sa commande.

### Cas d'exception 4 : "La livraison n'est pas conforme" :

- Le client déclare les écarts de livraison,  
→ Les ventes remboursent le client.

# Tests de cas d'utilisation Exercice



# *Techniques Boite Blanche*

## **Exemples de techniques Boîte Blanche**

- Test des instructions
- Test des décisions
- Test des conditions
- Test structurel

# *Techniques Boite Blanche*

## Instruction

- Une entité dans un langage de programmation, qui est typiquement la plus petit unite indivisible d'exécution

## Décision

- Un point dans un programme où le flux de contrôle a deux ou plus chemin possibles. Un noeud avec deux ou plus liens vers des branches séparées

## Condition

- Eléments d'une decision qui retourne une valeur booléenne

# Tests structurels (chemins)

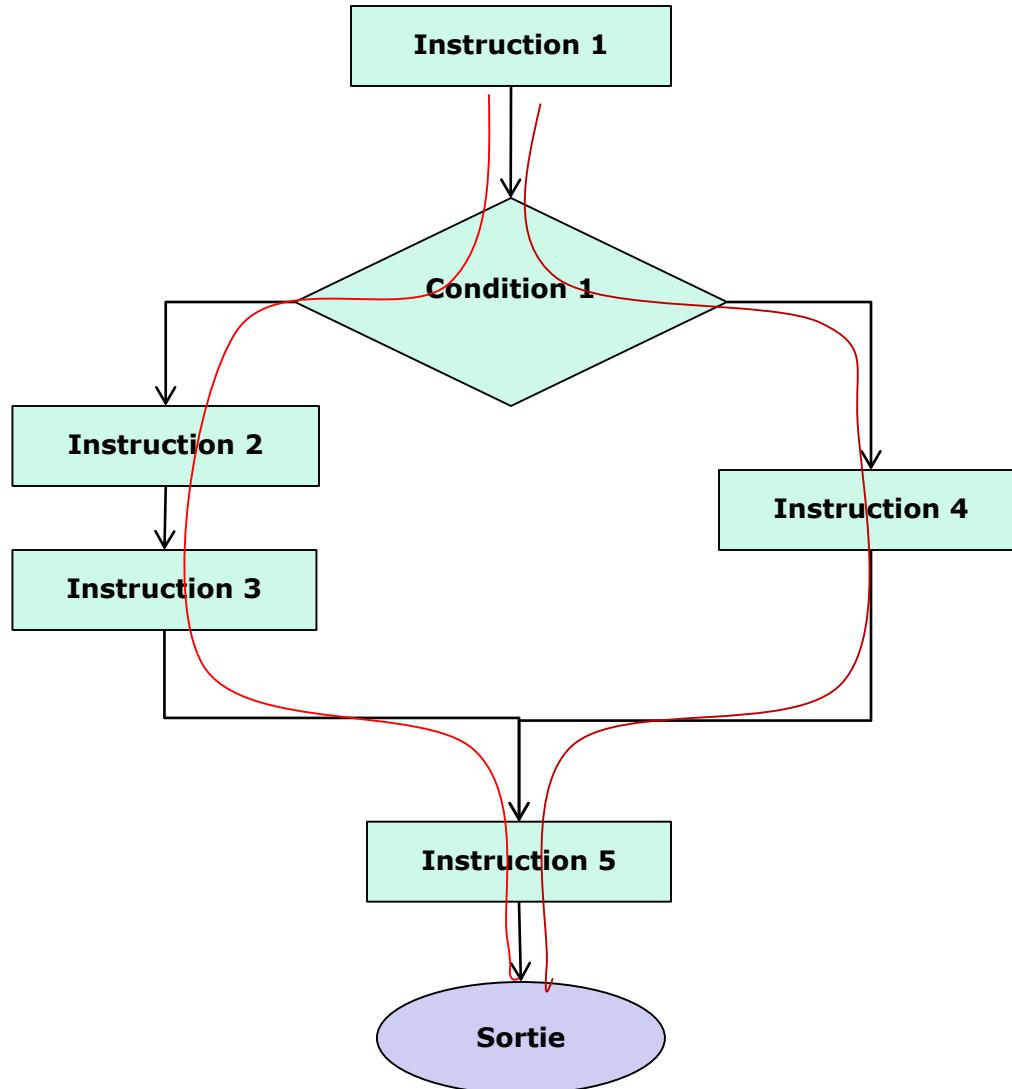
## Définition

- Technique de test basée sur la structure de l'application (boîte blanche).
- L'objectif est de représenter l'application sous la forme de graphes permettant de mettre en évidence les instructions, les branches, les décisions et les conditions

## Mise en place du graphe

- Instruction : entité dans un langage de programmation, qui est typiquement la plus petite unité indivisible d'exécution.
- Branche : bloc de base qui peut être sélectionné pour exécution, basé sur un « case, jump, go to, if then else ... ).
- Décisions : un nœud avec 2 ou plus liens vers des branches séparées.
- Conditions : expression logique qui peut être évaluée à Vrai ou Faux.

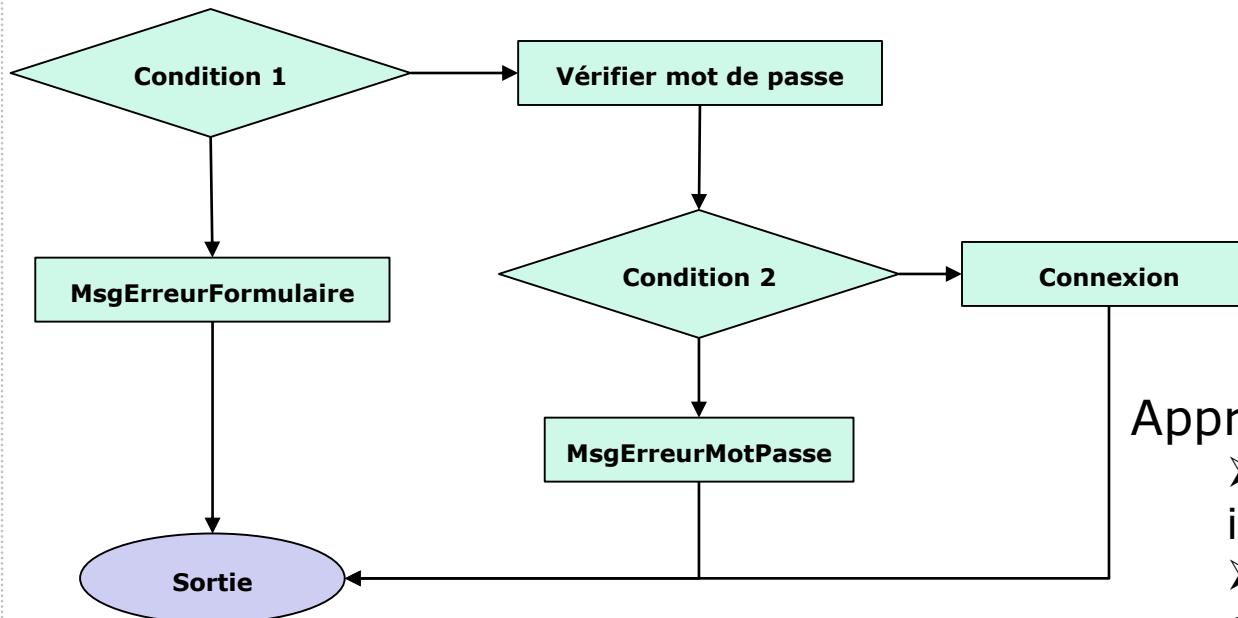
# Tests structurels (chemins)



Nombre de chemins possibles ?



# Tests structurels (chemins)



Approches possibles :

- Test de couverture des instructions
- Test de couverture des décisions
- Couverture des conditions

Condition 1 : Si Login non vide and Password non vide  
Condition 2 : Si Password OK

# *Techniques statiques*

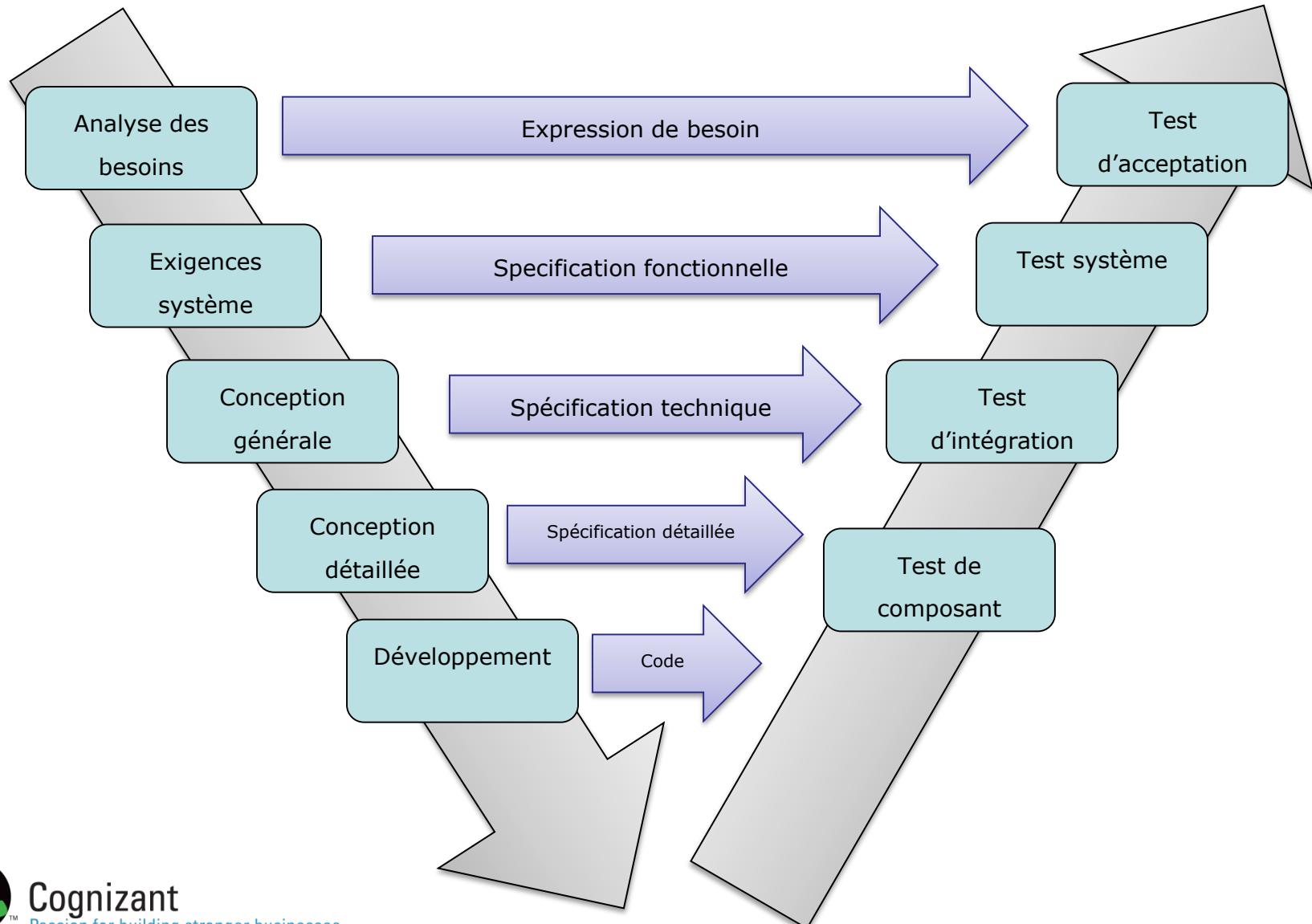
Ce sont des techniques qui n'exécutent pas le code du logiciel

- Revue du code
- Revue des spécifications de conception
- Revue des plan de test, ...

L'analyse statique peut être outillée (analyse de code)

Les revues trouvent des défauts plutôt que des défaillances

# Techniques statiques



# *Techniques basées sur l'expérience*

## **Test exploratoire**

Pas de scénario défini, utilisation de l'application à l'aide d'un expert . L'approche peut être basée sur l'estimation des erreurs.

## **Raisons**

- Manque documentation
- Manque de temps
- Complémentarité avec des test plus formel
- Permet de vérifier le processus de test

# *Formalisme des cas de test*

- Logiciels utilisés :
  - Excel
  - Outils d'aide à la gestion des tests (Testrail, ALM QC...)
- Contenu d'un cas de test :

<b>Objet</b>	<b>Définition</b>
Identifiant du cas de test	Identifiant, fonctionnalité, titre
Objectif du cas de test	<ul style="list-style-type: none"><li>• Objectif clairement énoncé</li><li>• Référence à une exigence spécifiée (spécifications)</li></ul>
Prérequis	Besoin de l'environnement (matériel et logiciel), configuration, conditions de départ (jeu de données)
Actions / Vérifications	A chaque action correspond une vérification et son résultat (passed, failed, blocked ...)
Défaut	Permet de lier un défaut et le cas de test qui l'a mis en évidence

# *Formalisme des cas de test*

## Règles de conception

- **Identification du cas test**
  - Dépend de l'outillage,
  - Dépend du projet
  - Très structurant, permet de visualiser la couverture
- **Objectif :**
  - Texte court qui permet à un testeur de comprendre la finalité du test. Les RG ou règle de calcul doivent être mentionnées
- **Prérequis**
  - Description des données nécessaires au test. Le jeu de donnée peut être fabriqué à l'avance et décrit par un identifiant.

# *Formalisme des cas de test*

## Règles de conception

- **Actions/Vérification**
  - La description des actions à effectuer doit toujours être fait de la manière suivante :
    - Actions à effectuer > Résultat attendu
  - Le niveau de détail de la description des actions à effectuer doit s'adapter au contexte du projet.
  - Le niveau de détail du résultat attendu doit être le plus fin possible afin d'éviter tout risque d'interprétation des résultats. Des pièces jointes/images peuvent être ajoutées.
- **Défaut**
  - La description du défaut est en général faite à l'aide de l'ID du défaut associé. Le défaut est décrit dans un logiciel dédié.

# Sommaire

- Analyse des exigences
- Estimation et planification des tests
- Couverture de test
- Conception des tests
- Techniques de conception des tests
- **Exécution des tests**
  - » Introduction
  - » Exécution des test
  - » Définition d'un défaut
  - » Sévérité d'un défaut
  - » Cycle de vie d'un défaut
  - » Suivi des défauts et rejeux
  - » Defect leakage
  - » Rapport de test
- Finalisation de la recette



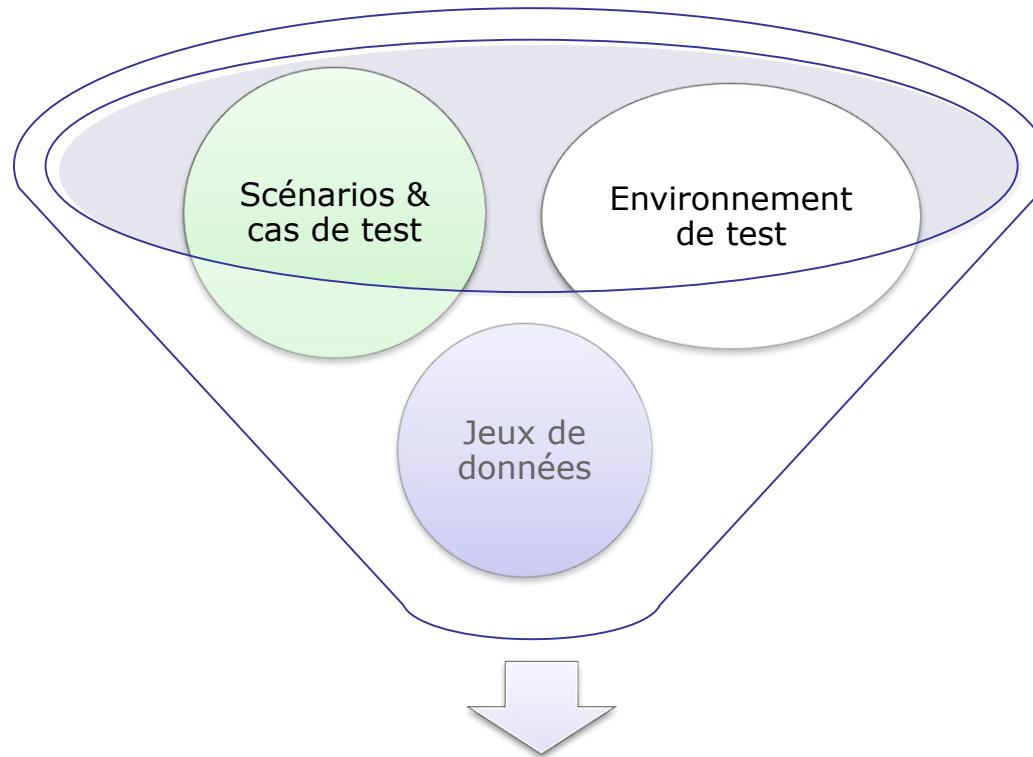
# *Introduction*

## Qu'est ce l'exécution ?

- Cette activité est le test à proprement parler du logiciel.
- Les testeurs exécutent les scénarios préalablement définis avec les jeux de données préparés, dans l'environnement de test livré.
- Dans le cas où un comportement inhabituel est détecté lors de cette phase, celui-ci est décrit dans une fiche d'anomalie qui permet de conserver une trace du problème.

# *Introduction*

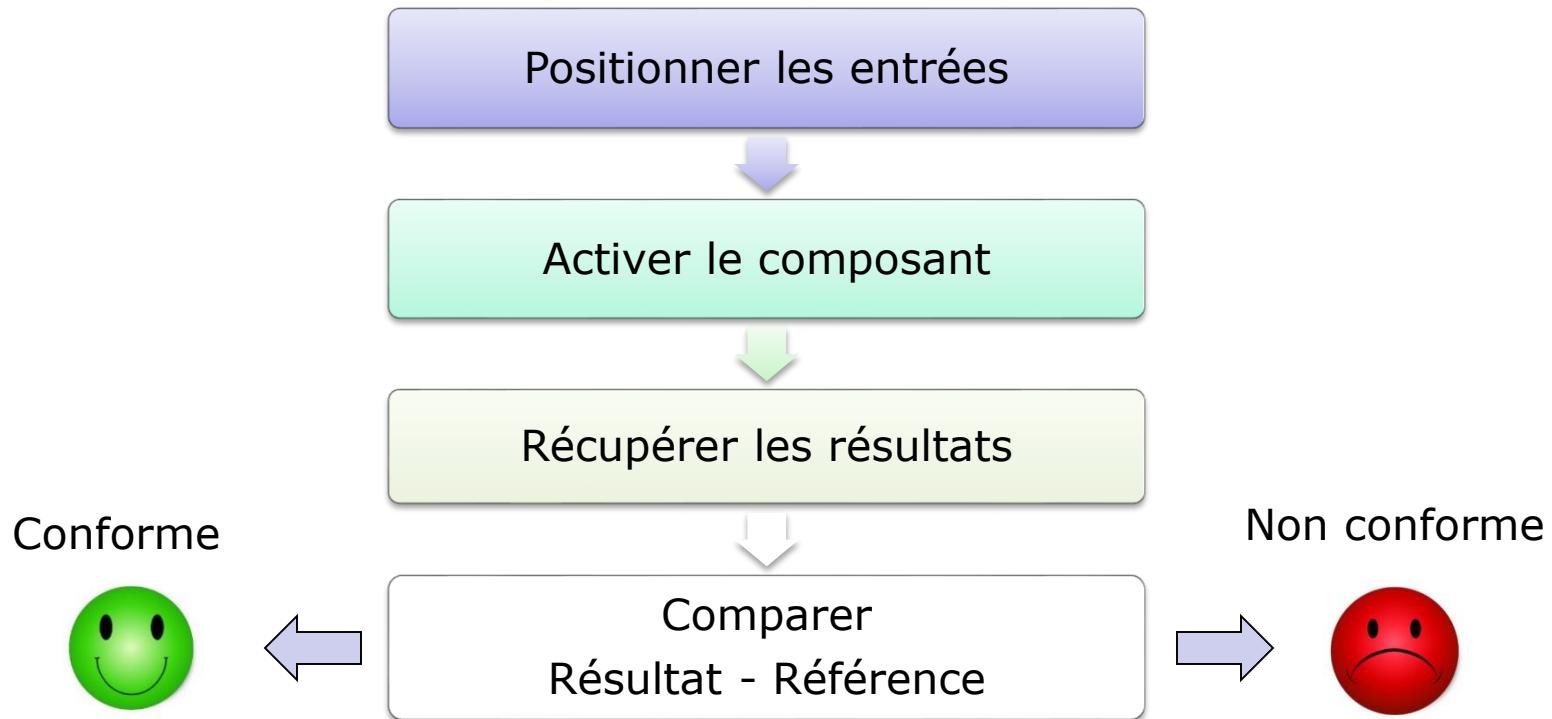
- Entrées nécessaire au commencement de la phase d'exécution :



Début de l'exécution des tests

# Exécution

- Exécution d'un cas de test :



# Exécution

- Statuts des cas de test :

## No run

- Statut initial du test avant d'être exécuté par le testeur

## Passed

- Statut obtenu lorsque les résultats obtenus sont conformes aux résultats attendus

## Not completed

- Statut du test lorsque son exécution n'est pas terminée

## Blocked

- Statut du test lorsque le testeur est bloqué par un facteur externe au test

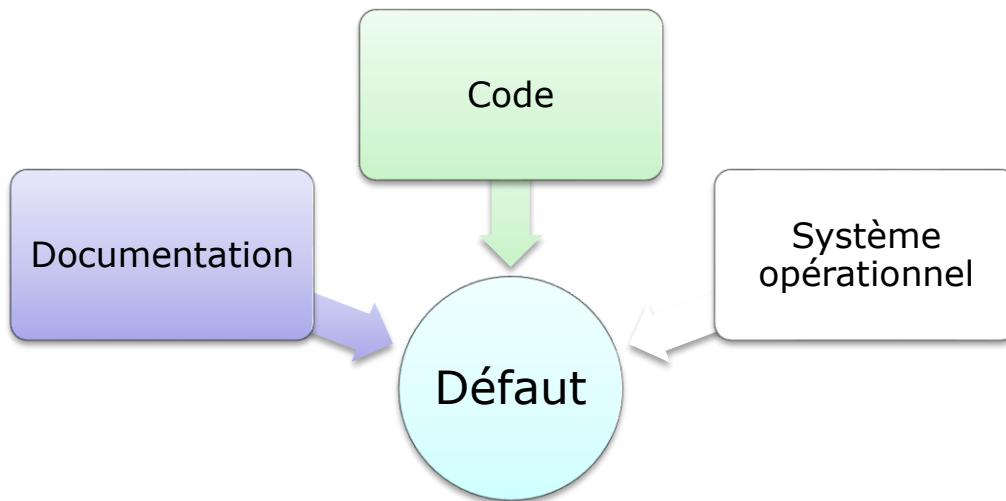
## Failed

- Statut obtenu lorsque les résultats obtenus ne sont pas conformes aux résultats attendus

# Défaut

## Qu'est ce qu'un défaut ?

- Une imperfection dans un composant ou un système qui peut conduire à ce qu'un système n'exécute pas les fonctions requises.
- Un défaut, si rencontré lors de l'exécution, peut causer la défaillance d'un composant ou d'un système
- Il survient à cause de problème dans :



# Défaut

## Que faire en cas de défaut ?

- Vérifier que le test est bien correct, en accord avec les spécifications et les exigences
- Vérifier que le problème n'est pas déjà répertorié
- Créer un nouveau défaut

## Pourquoi ouvrir des défauts ?

- Fournir aux développeurs un retour sur le problème concerné pour en permettre l'identification, la localisation et la correction nécessaire
- Fournir aux responsables du test le moyen de suivre la qualité du système et l'avancement du test
- Fournir des idées pour l'amélioration du système testé

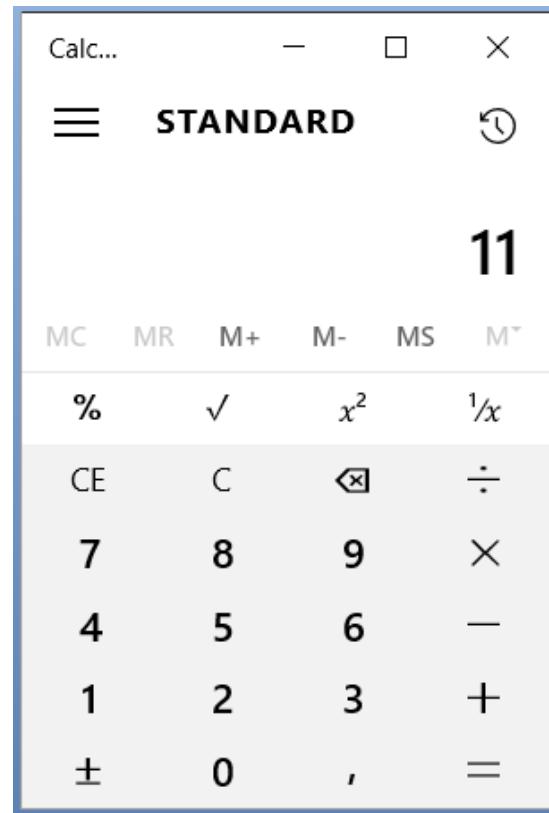
# Contenu d'un défaut

Objet	Définition
ID	Identifiant unique du défaut
Synthèse	Résumé de l'incident avec des mots clés
Description	<ul style="list-style-type: none"><li>- Jeux de données utilisé</li><li>- Environnement dans lequel le test a été exécuté</li><li>- Etapes de reproduction du défaut / procédure de test</li><li>- Résultat attendu</li><li>- Résultat observé</li><li>- Reproductibilité</li></ul>
Attachement	Capture d'écran du défaut observé
Sévérité	Niveau d'impact et d'importance du défaut
Date et heure	Date et heure d'ouverture du défaut
Statut	Statut du défaut (nouveau, ouvert, qualifié, rejeté, livré, fermé...)

# Exercice – Ouvrir un nouveau défaut

En utilisant les informations disponibles ci-dessous, créez un défaut le plus complet possible :

Etape	Action	Résultat attendu
1	Ouvrir l'application Calculatrice	L'application est ouverte
2	Appuyer sur le bouton « 1 »	La valeur « 1 » est affichée
3	Appuyer sur le bouton « + »	-
4	Appuyer sur le bouton « 1 »	La valeur « 1 » est affichée
5	Appuyer sur le bouton « = »	La valeur « 2 » est affichée
6	Appuyer sur « x » pour fermer l'application	L'application est fermée



# Sévérité d'un défaut

## Critique

- Défaut qui entraîne la terminaison du logiciel complet et provoque la corruption étendue des données

## Majeur

- Défaut qui entraîne la fin de l'un ou plusieurs composants du logiciel et provoque une corruption étendue des données

## Moyen

- Défaut qui n'entraîne pas la terminaison, mais qui provoque des résultats incorrects, incomplets ou incohérents

## Mineur

- Défaut qui ne conduit pas à la terminaison et n'endommage pas la convivialité du système
- Les résultats souhaités peuvent être facilement obtenus en contournant le défaut

## Cosmétique

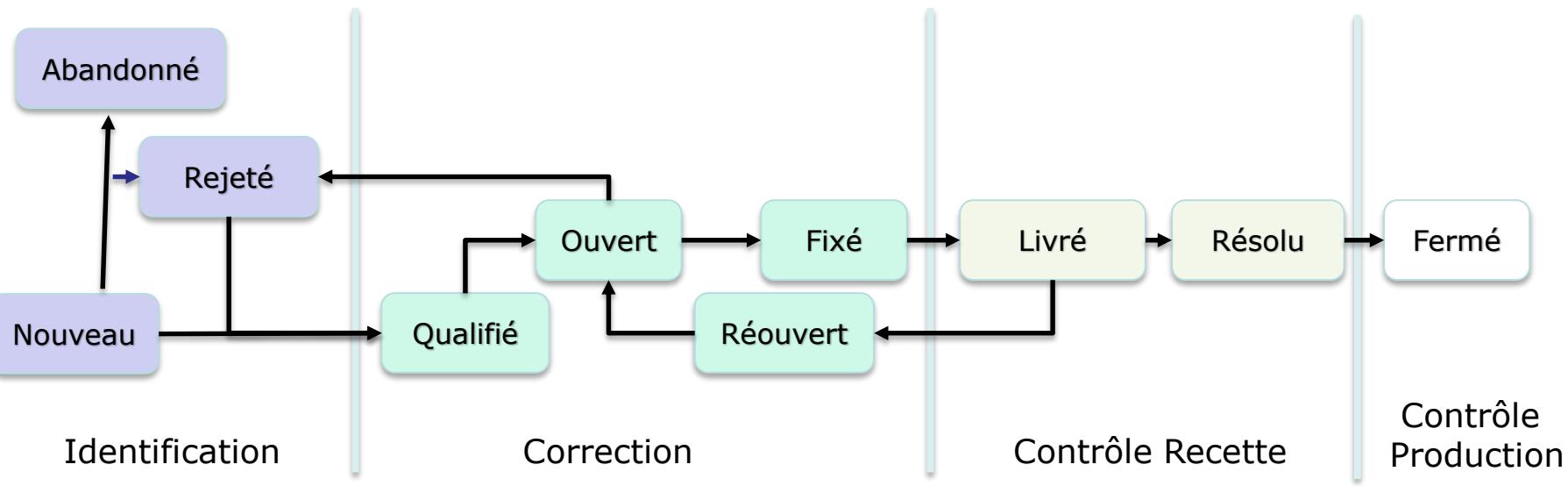
- Défaut lié à l'amélioration du logiciel où les changements sont liés à l'apparence de l'application



Cognizant

Passion for building stronger businesses

# Cycle de vie d'un défaut



# *Suivi des défauts et rejeux*

- Pourquoi suivre les défauts ?
  - Débloquer les tests liés à ces défauts
  - Vérifier que les défauts soient corrigés et limiter ainsi les réserves en fin de recette
- Quand faut-il rejouer les tests liés aux défauts identifiés ?

Défauts au statut  
« Livré »



La correction est-elle  
effective ?

Défauts au statut  
« Rejeté »



Le défaut est-il reproductible  
? Le scénario décrit dans le  
défaut est-il suffisamment  
complet ?

# Defect leakage

- Un défaut trouvé par le client ou l'utilisateur final après la mise en production du système testé est appelé un defect leakage.

Définition

Pourquoi ?

- Requirement manquant
- Couverture de test incomplète
- Inefficacité du testeur

Prévention

Impacts

- Impliquer les testeurs dès la phase d'analyse pour avoir une connaissance approfondie des exigences.
- Préparer une bonne couverture de test.
- Test efficace

- Mauvaise impression de l'utilisateur final
- Coût élevé pour corriger le défaut
- Affecte le planning
- Client mécontent

# Rapport de test

Objet	Définition
ID	Identifiant unique du document
Synthèse	Résumé de ce qui a été testé et des résultats des tests. Cette rubrique fait également référence à la documentation des tests associée (plan de test, etc.)
Variance	Liste des écarts entre le plan de test, spécification de test et ce qui a vraiment été exécuté
Evaluation de la complétude	Evaluation de la couverture de test
Résumé des résultats	Résumé des résultats, incluant les défauts résolus avec actions prises et les défauts restants
Evaluation	Evaluation de la qualité et des fiabilités des tests
Synthèse des activités	Synthèse des principales activités et des événements, des ressources utilisées
Visas	Validation/signature du rapport par les personnes désignées

# Sommaire

- Analyse des exigences
- Estimation et planification des tests
- Couverture de test
- Conception des tests
- Techniques de conception des tests
- Exécution des tests
- **Finalisation de la recette**
  - » Introduction
  - » Mise à jours des cas de tests
  - » Post mortem
  - » Métriques
  - » Livrables
    - PV de recette
    - Bilan de recette

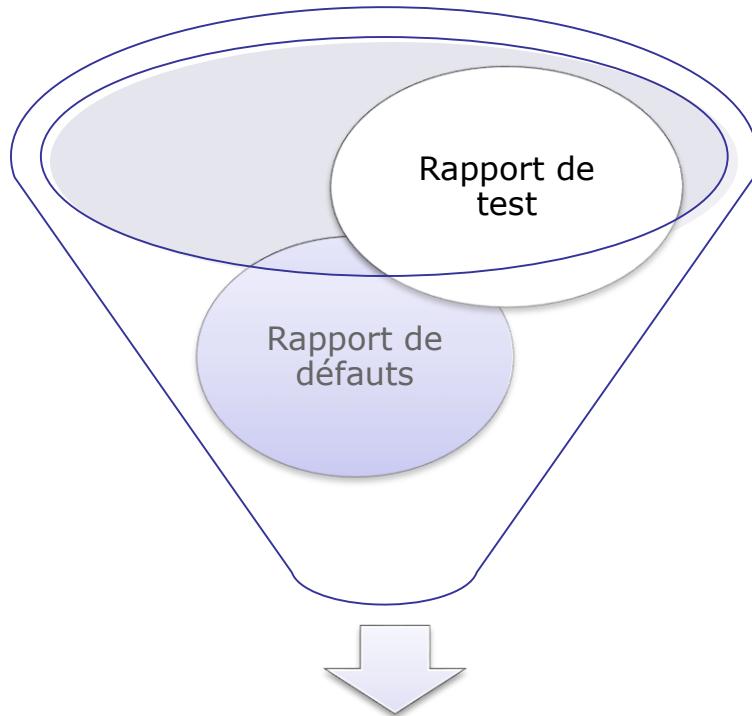
# *Introduction*

## Qu'est ce la finalisation ?

- Cette activité permet de synthétiser la phase de test quand elle est terminée.
- Elle décrit tous les éléments survenus lors du test et s'accompagne de recommandations pour l'utilisation ou l'évolution du logiciel testé.

# *Introduction*

- Entrées nécessaire au commencement de la phase de finalisation :

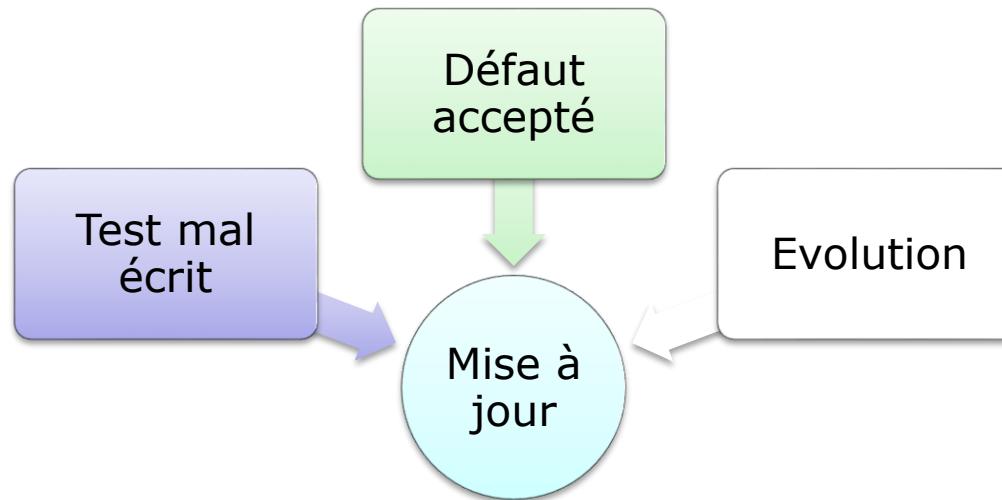


Début de la phase de finalisation

# Mise à jour des tests

## Pourquoi mettre à jour les cas de tests ?

- 3 raisons principales :



- Les tests sont aménés à être réutilisés dans les itérations suivantes ou lors d'une phase de non régression plus importante.
- Ils doivent être à jour et compréhensibles de tous les testeurs, y compris novices sur le projet.

# Métriques

## Que sont les métriques ?

- Une métrique, par définition, est tout type de mesure utilisée pour mesurer un certain composant quantifiable de performance.
- Pour mettre en place un programme de métriques efficace, il est important de mettre du temps de côté pour planifier ces choses dans cet ordre:

Quelles informations rassembler ?

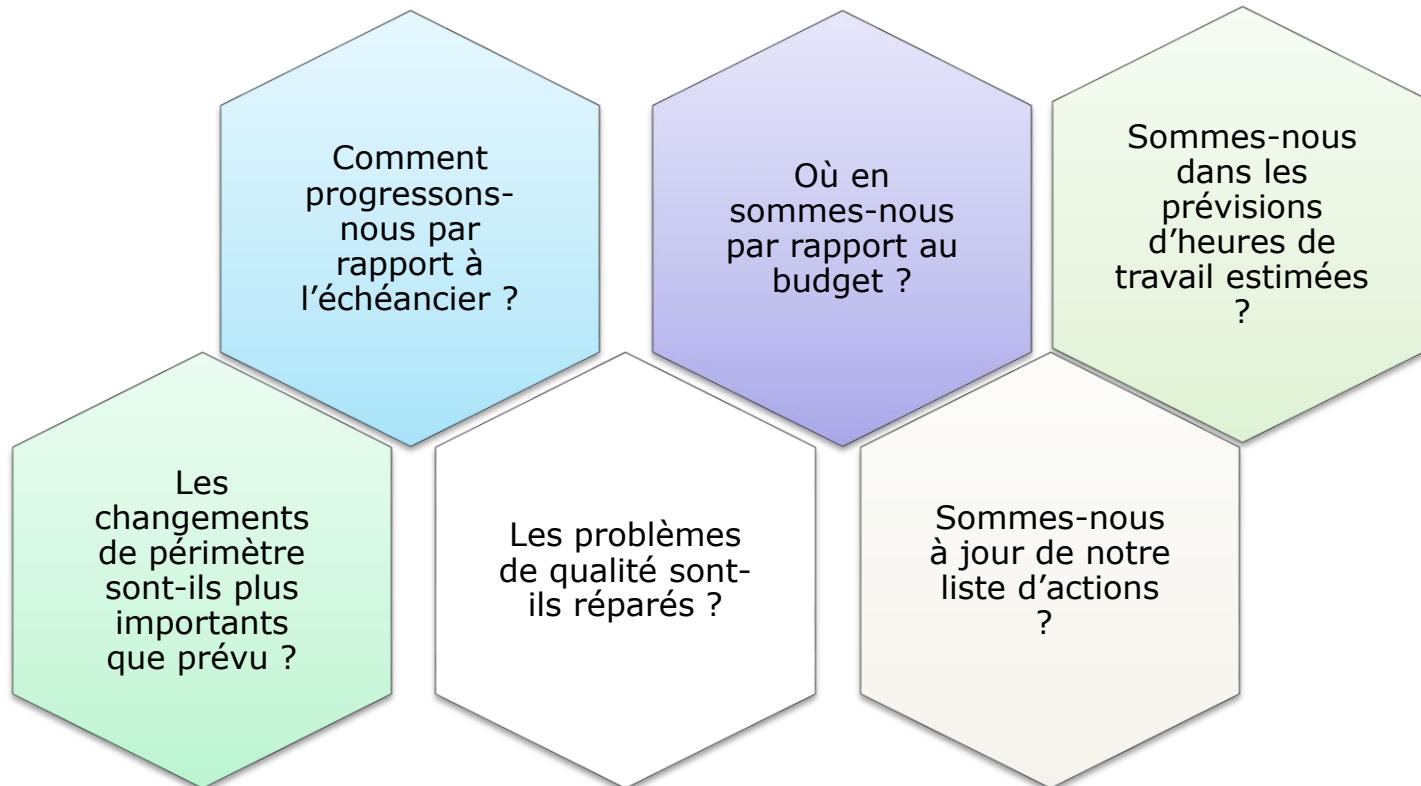
Comment collecter les informations ?

Quelles méthodes utiliser pour traiter et analyser les informations ?

Comment faire le rapport sur les résultats ?

# Métriques

- Quelques métriques communes :



# *Post mortem*

## Quand intervient le post mortem ?

- Le post mortem se situe à la fin d'un projet, lorsqu'il est nécessaire de tirer un bilan des actions passées et des résultats obtenus.

## Qu'est ce que le post mortem ?

- Le post mortem est l'analyse du décalage entre le cycle de vie idéal et le cycle de vie réel d'un projet.

## Pourquoi faire un post mortem ?

- Pour améliorer les points futurs en :
  - Corrigeant les points faibles
  - Capitalisant les points forts d'un projet achevé

# *Post mortem*

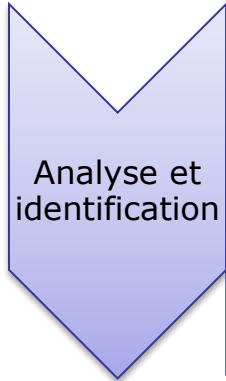
## Questions niveau tactique

- 1. Quels ont été les étapes-clés du projet ?
- 2. Quelles étapes ont rencontré des difficultés ou ont été mal vécues ?
- 3. Quelle est la satisfaction globale de l'équipe du projet et du client ?
- 4. Quels processus de travail se sont révélés pertinents, efficaces ou intéressants ?

## Questions niveau stratégique

- 1. Le « workflow » des processus est-il logique et optimisé ?
- 2. Les interactions et coordination entre les différents projets et équipes sont-elles suffisamment transparentes et claires ?
- 3. Nos offres de services reflètent-elles la réalité des projets ?

# Post mortem



Analyse et  
identification

- Le post-mortem se prépare en amont. Chaque personne ayant participé au projet doit se faire la réflexion sur des points précis:
  - Le respect des spécifications du projet
  - L'efficacité et transparence de l'information
  - Le temps prévu versus le temps passé
  - A-t-on eu ou perçu des signaux d'alarme (interne ou client) ?



Réunion

- **Réunion pour l'équipe du projet :** Lors de la réunion, les participants soulèvent les bons et mauvais points du projet.
- **Réunion avec l'équipe client :** Les mêmes points sont soulevés, mais cette fois ci avec l'équipe client.



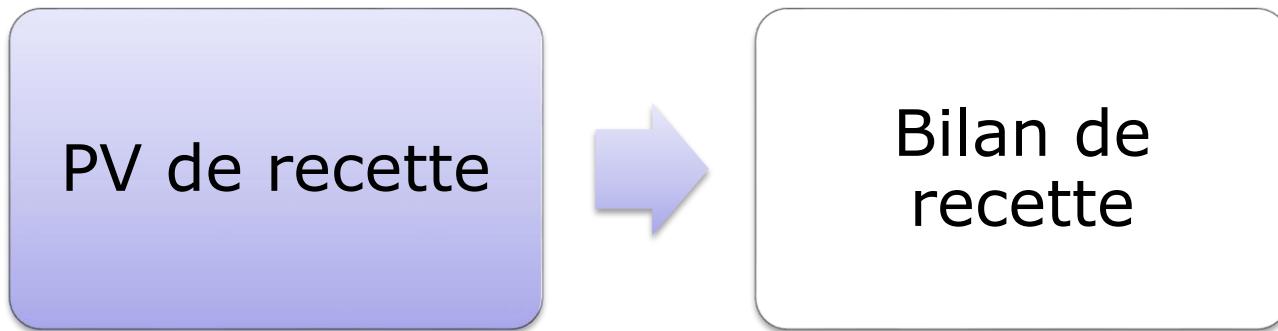
Plan d'action

- Le post-mortem indique la fin du projet, mais surtout le début de vie du produit. Les actions prises permettront aux futurs projets (avec un même client ou avec de nouveaux) d'éviter de reproduire les mêmes enjeux.



# Livrables

- Que faut-il livrer au client en fin de recette ?



# PV de recette

Pour clore chaque étape de la procédure de recette, un procès-verbal est rédigé.

Celui-ci a pour objet de :

- valider la livraison du projet
- mentionner les réserves émises

En général, il est rempli par les testeurs, puis revu et signé par le chef de projet.

PV de  
recette

- Présentation générale du projet
- Réserves émises
- Décision de go / non go
- Date et signature du document

# Bilan de recette

Le bilan de recette est un document power point qui regroupe toutes les informations de la recette.

Celui-ci a pour objet de :

- Résumer toutes les données de la recette
- Proposer des améliorations pour les recettes suivantes

En général, il est rempli par les testeurs, puis revu et envoyé par le chef de projet.

Bilan de  
recette

- 
- Introduction
  - Progression
  - Défauts
  - Métriques
  - Defect leakage
  - Feedbacks
  - Liste des améliorations

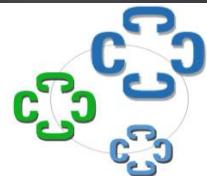


Cognizant  
Passion for building stronger businesses

## Questions / réponses



[olivier.octobre@cognizant.com](mailto:olivier.octobre@cognizant.com)



A  
FORTUNE  
1000  
COMPANY