

TESTING

5'',

ATELIER EN ÉQUIPE



Définissez ce que représente
à priori pour vous le Test

Énoncé

Par table, retenez 6 idées majeures ou éléments structurants et venez les exposer

Donner aussi la charge de travail associée à l'activité de Test

QU'EST-CE QUE LE TEST

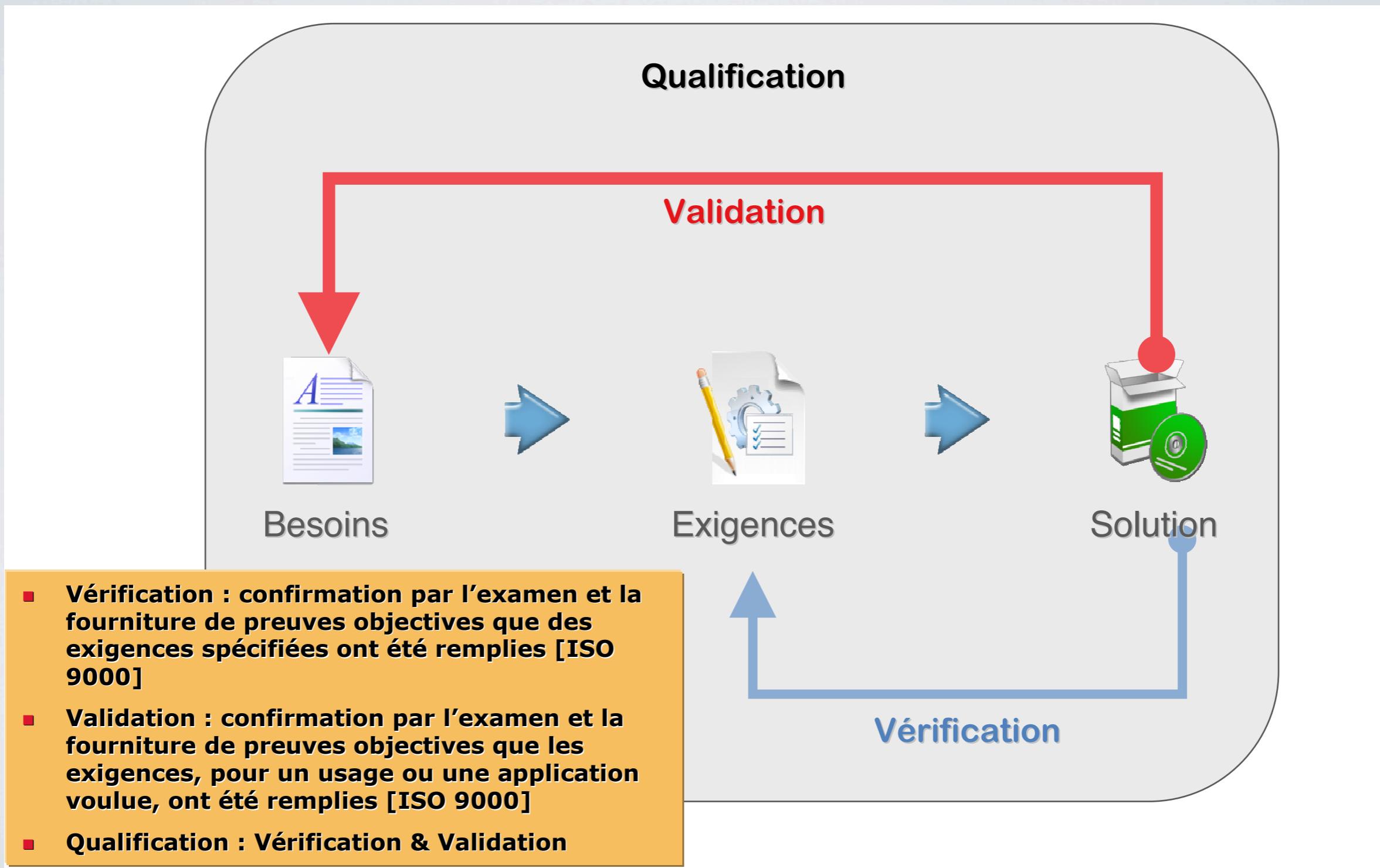
- « Processus consistant en toutes les activités du cycle de vie, statiques et dynamiques, concernant la planification et l'évaluation de produits logiciels et produits liés pour déterminer s'ils satisfont aux exigences spécifiées, pour démontrer qu'ils sont aptes aux objectifs et pour déceler des anomalies »

QUE FAUT-IL TESTER ?

- Les éléments à tester (dans la limite de notre fourniture):
- Composants / services
- Applications / IHM
- Transactions structurantes
- Processus métier (Use Case)
- Habilitations
- Chaînes de liaison
- Interfaces / flux
- Batches
- Données migrées, ponts migratoires
- Procédures d'exploitation
- Master d'installation
- Processus de mise en service
- ...

Mais il faut aussi savoir
s'arrêter

QUALIFICATION LOGICIELLE



DÉFINITION

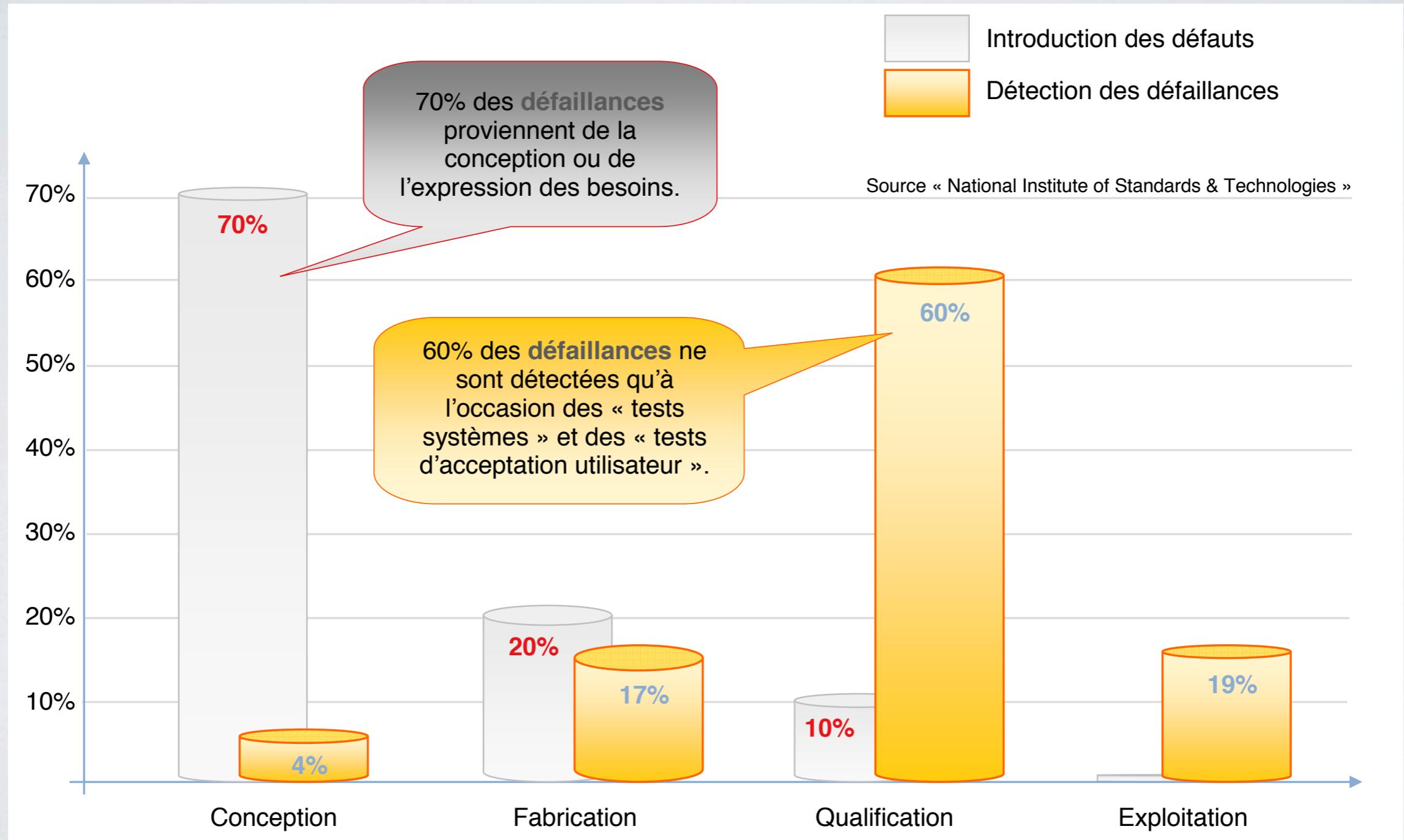
- La qualification logicielle a pour finalité la vérification de la conformité d'un produit au regard de ses spécifications et exigences clients
- Cette activité concerne :
 - la MOE pour contrôler la qualité de sa production
 - la MOA pour vérifier la conformité des éléments qui lui sont livrés



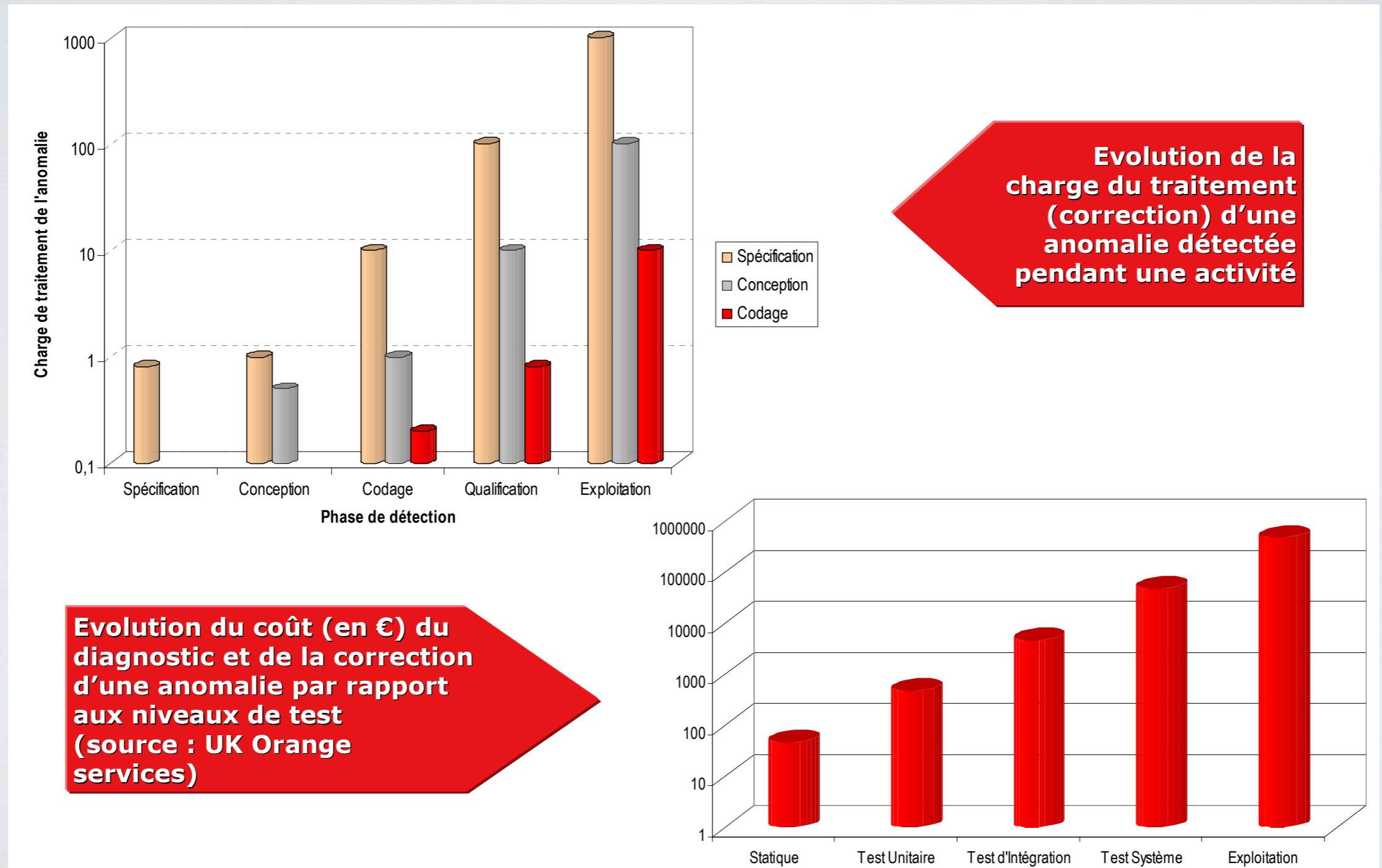
LA RÉALITÉ DES PROJETS

- Dérive dans le temps :
 - 51% des projets ne respectent pas leurs dates de livraison
 - 16 % des projets livrent dans les temps et avec leur budget prévu
- Surcoût du projet :
 - 43% des projets dépassent leur budget initial
 - dont 53% des projets coûtent 189% du budget initial estimé
- Taux d'anomalies trop important
- Turnover des équipes
- Exigences oubliées, fausses ou mal-comprise

DÉCALAGE ENTRE INJECTION ET DÉTECTION DES DÉFAUTS

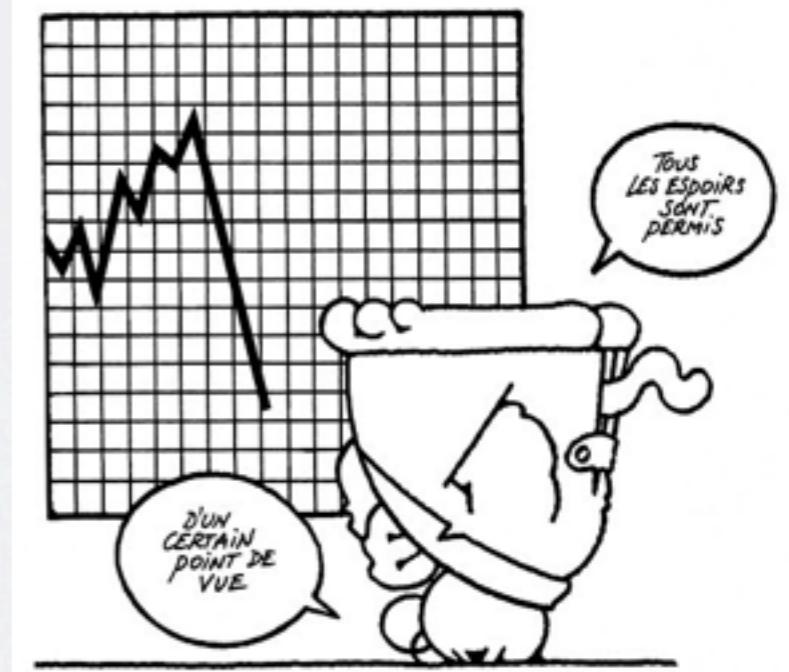


CHARGE ET COUT DES DÉFAUTS



CONSTAT

- L'impasse sur la qualification n'est jamais gagnante
- Coûts de corrections après livraison >>> Coûts de la qualification
- Coût de la non-qualité (plus tard, plus cher)
 - Le test réduit les coûts





PRINCIPES DE TESTS

40 années de tests offrent des *indications communes*

PRINCIPE I

« Les tests montrent la présence de défauts »

- Les tests peuvent prouver la présence de défauts, mais ne peuvent en prouver l'absence
- Les tests réduisent la probabilité que des défauts restent cachés dans le logiciel mais, même si aucun défaut n'est découvert, ce n'est pas une preuve d'exactitude

PRINCIPE 2

« Les tests exhaustifs sont impossibles »

Tout tester (toutes les combinaisons d'entrées et de pré-conditions) n'est pas faisable sauf pour des cas triviaux

Exemple du test de l'IHM suivante :

- 20 écrans
- 4 menus / 3 options
- 10 champs
- 2 types de données
- 100 valeurs possibles

Valeurs courante d'exécution :

- Expert = 10 s / Moyen = 1 mn / Junior = 10 mn

Total pour test « exhaustif »

$$20 \times 4 \times 3 \times 10 \times 2 \times 100 = 480000 \text{ cas de tests}$$

Expert (10 s par test !) : 18 jours !

Plutôt
que des tests exhaustifs, utilisons
les risques et les priorités pour focaliser
les efforts de tests !

PRINCIPE 3

« Tester tôt »

Les activités de tests devraient commencer aussi tôt que possible dans le cycle de développement du logiciel ou de la solution, et devraient être focalisés vers des objectifs définis

- Dès la conception des cas tests, des erreurs sont détectées
- Les fautes identifiées plus tôt coûtent moins cher à réparer
- Les fautes « critiques » sont trouvées en premier lieu
- Les fautes sont détectées avant qu'elles n'entrent dans le code
- Tester tôt ne nécessite pas d'effort supplémentaire, simplement une organisation des activités
- Les Tests Unitaires et d'Intégration composants sont nécessaires

PRINCIPE 4 ET 5

« Regroupement des défauts »

Un petit nombre de modules contient la majorité des défauts détectés lors des tests pré-livraison, ou affichent le plus de défaillances en opération

« Paradoxe du pesticide »

- Si les mêmes tests sont répétés de nombreuses fois, il arrive un moment où le même ensemble de cas de tests ne trouve plus de nouveaux défauts
- Pour prévenir ce “paradoxe du pesticide”, les cas de tests doivent être régulièrement revus et révisés, et de nouveaux tests, différents, doivent être écrits pour couvrir d’autres chemins dans le logiciel ou le système /solution de façon à permettre la découverte de nouveaux défauts

PRINCIPE 6 ET 7

« Les tests dépendent du contexte »

Par exemple un logiciel de sécurité sera testé différemment d'un site e-commerce

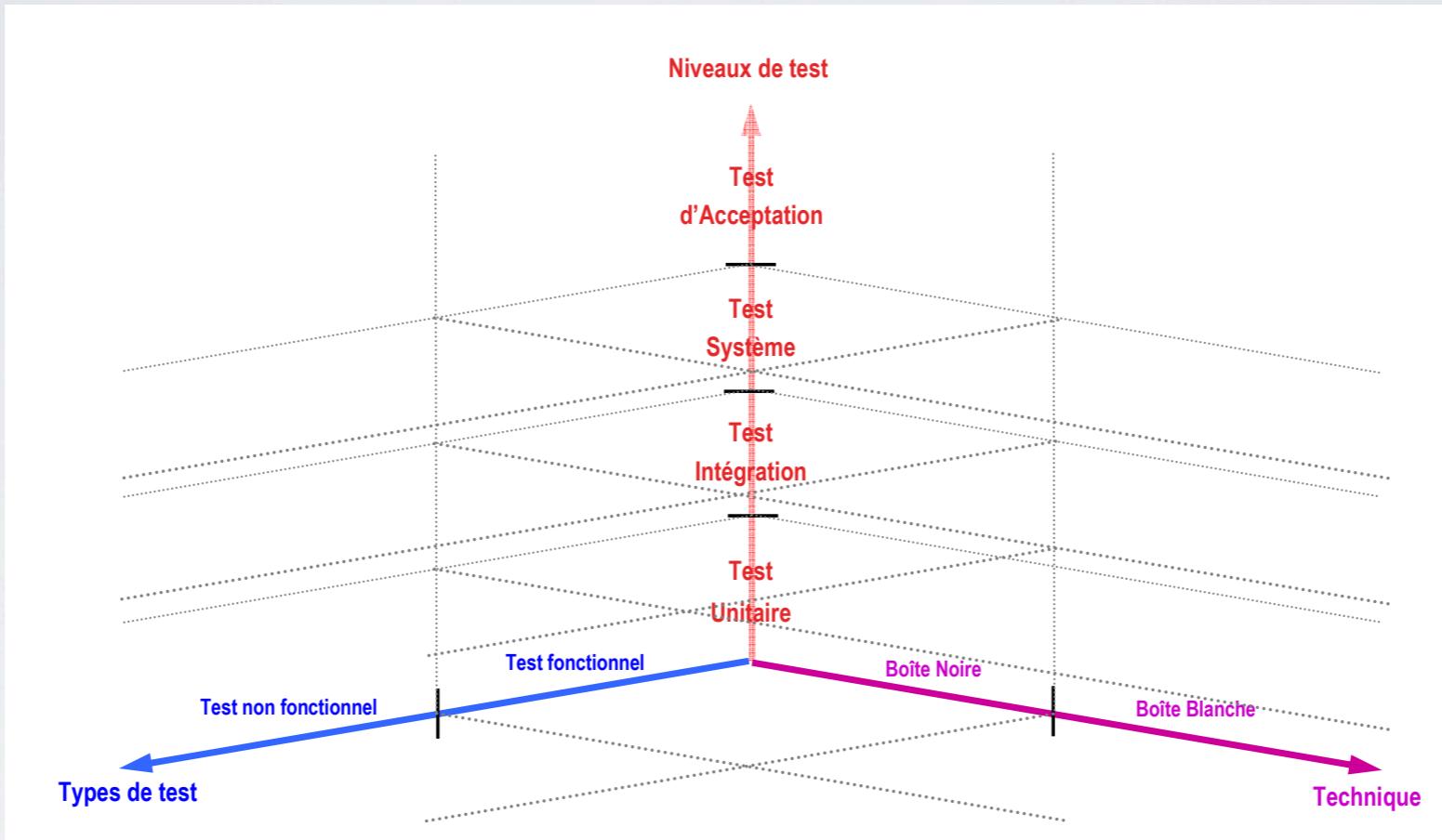
« L'illusion de l'absence d'erreurs »

Trouver et corriger des défauts n'aide pas si le système conçu est inutilisable et ne comble pas les besoins et les attentes des utilisateurs

PSYCHOLOGIE DE LA QUALIFICATION

La meilleure façon d'établir la confiance dans la qualité d'une solution est d'essayer de la détruire !

- Le succès de la qualification est influencé par des facteurs psychologiques
- Le paradoxe : Confiance ?
 - But de l'activité : trouver des fautes
 - Cependant le but est bien d'établir la confiance dans la qualité de ce qui est produit ... plus précisément de démontrer sa fiabilité
- Des problèmes de communication peuvent survenir, particulièrement si les testeurs sont vus uniquement comme messagers porteurs de mauvaises nouvelles concernant des défauts (Développeur = créateur / Testeur = Destructeur)



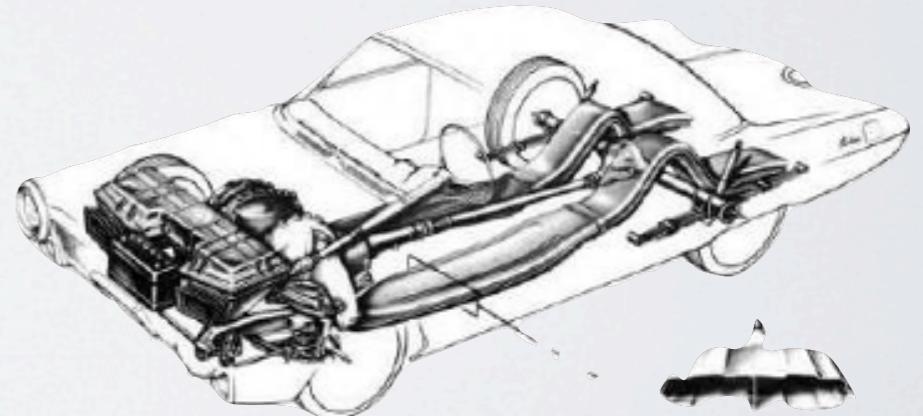
TYPOLOGIE DE TESTS

TYPE DE TEST

- **Test fonctionnel** => test des exigences fonctionnelles
- **Test non-fonctionnel** => test des exigences non-fonctionnelles
 - Interfaces : Utilisateur, Matérielles, Logicielles, Communication / Interopérabilité
 - Performance
 - Charge / Stress / Robustesse
 - Maintenance
 - Exploitation
 - Disponibilité
 - Sécurité
 - Fiabilité
 - Portabilité
 - Installation
 - Utilisabilité
 - Ergonomie
 - ...

TECHNIQUES DE TEST

- Boite noire : test bâti sans référence au contenu interne de l'objet testé
- Boite blanche : test bâti sur le contenu interne de l'objet testé



NIVEAUX DE TEST

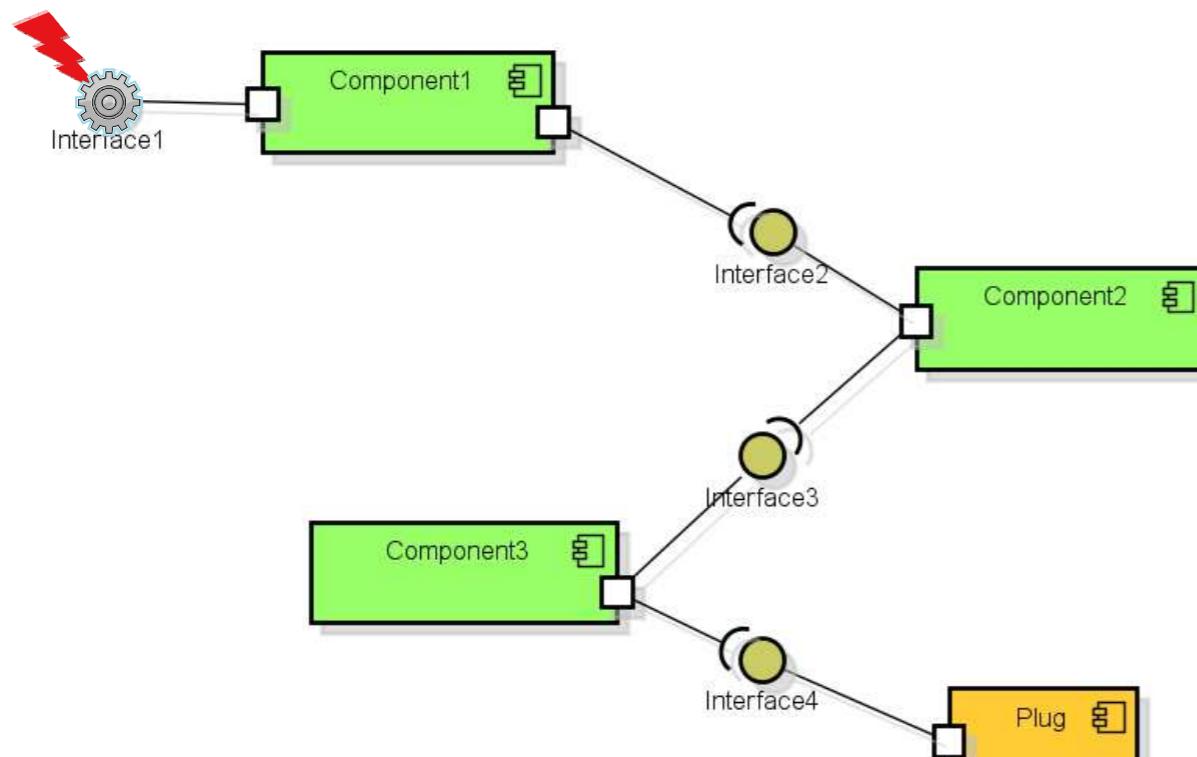
- Test Unitaire - Unit Testing
 - Test d'un élément logiciel
 - Autres appellations : TU / Test Composant / Test Composant Logiciel
 - Tests fonctionnels basés sur Solution Service Specification :
 - Contrat de services
 - Opérations
 - Tests non fonctionnels (exemples)
 - Performances
 - Robustesse
 - ...
 - Réalisés par l'Equipe de développement

NIVEAUX DE TEST

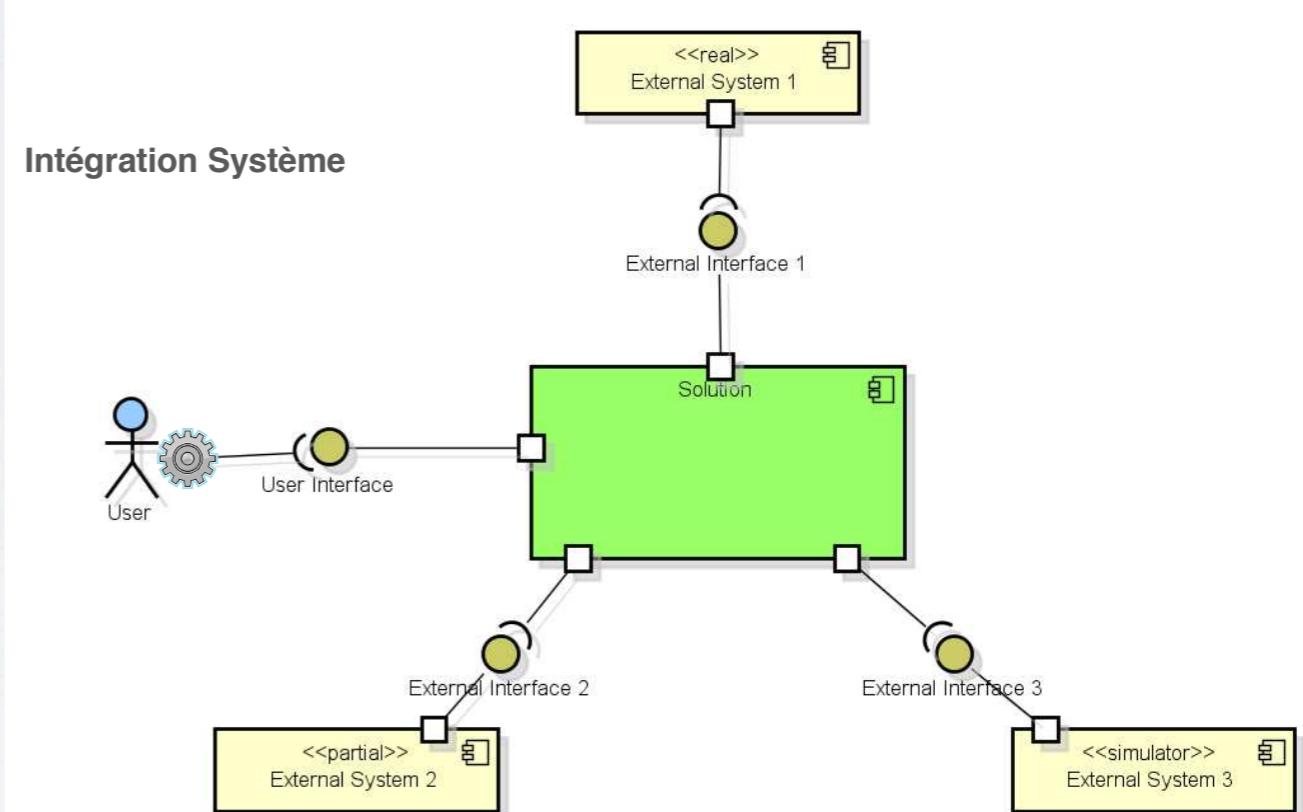
- Tests d'Intégration - Integration Testing
 - Tests d'Intégration des Composants
 - Tests des interfaces et des interactions entre les composants
 - Tests d'Intégration Système
 - Tests des interactions avec les autres systèmes ou sous-systèmes
 - Autres appellations : TI
 - Tests fonctionnels basés sur Solution Service Specification
 - Contrat de services
 - Opérations
 - Tests non fonctionnels (exemples)
 - Interfaces
 - Performances
 - Charge
 - Stress
 - ...
 - Réalisés par l'Equipe de développement

NIVEAUX DE TEST

- Tests d'Intégration - Integration Testing



Intégration de composants



Intégration Système

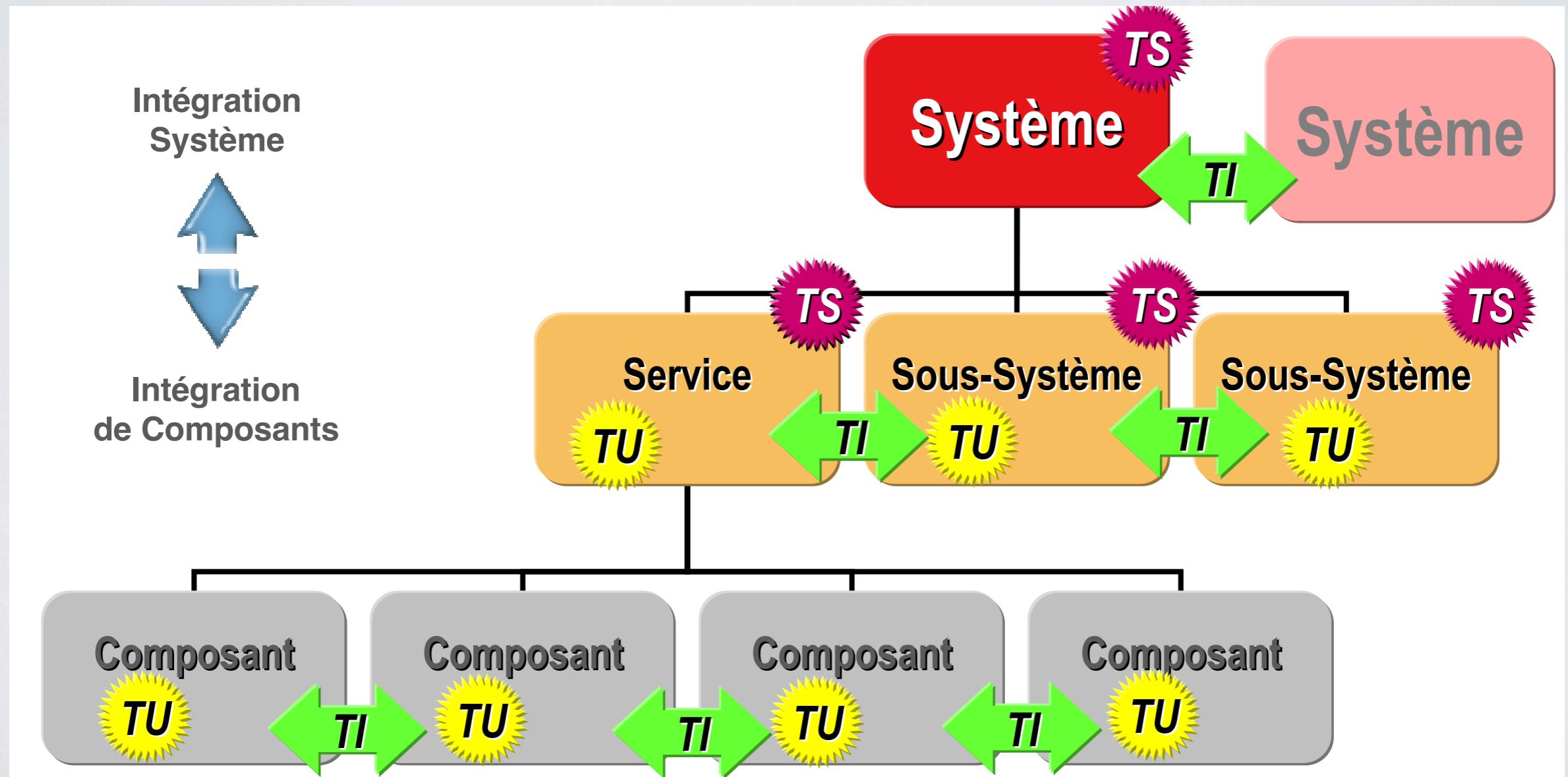
NIVEAUX DE TEST

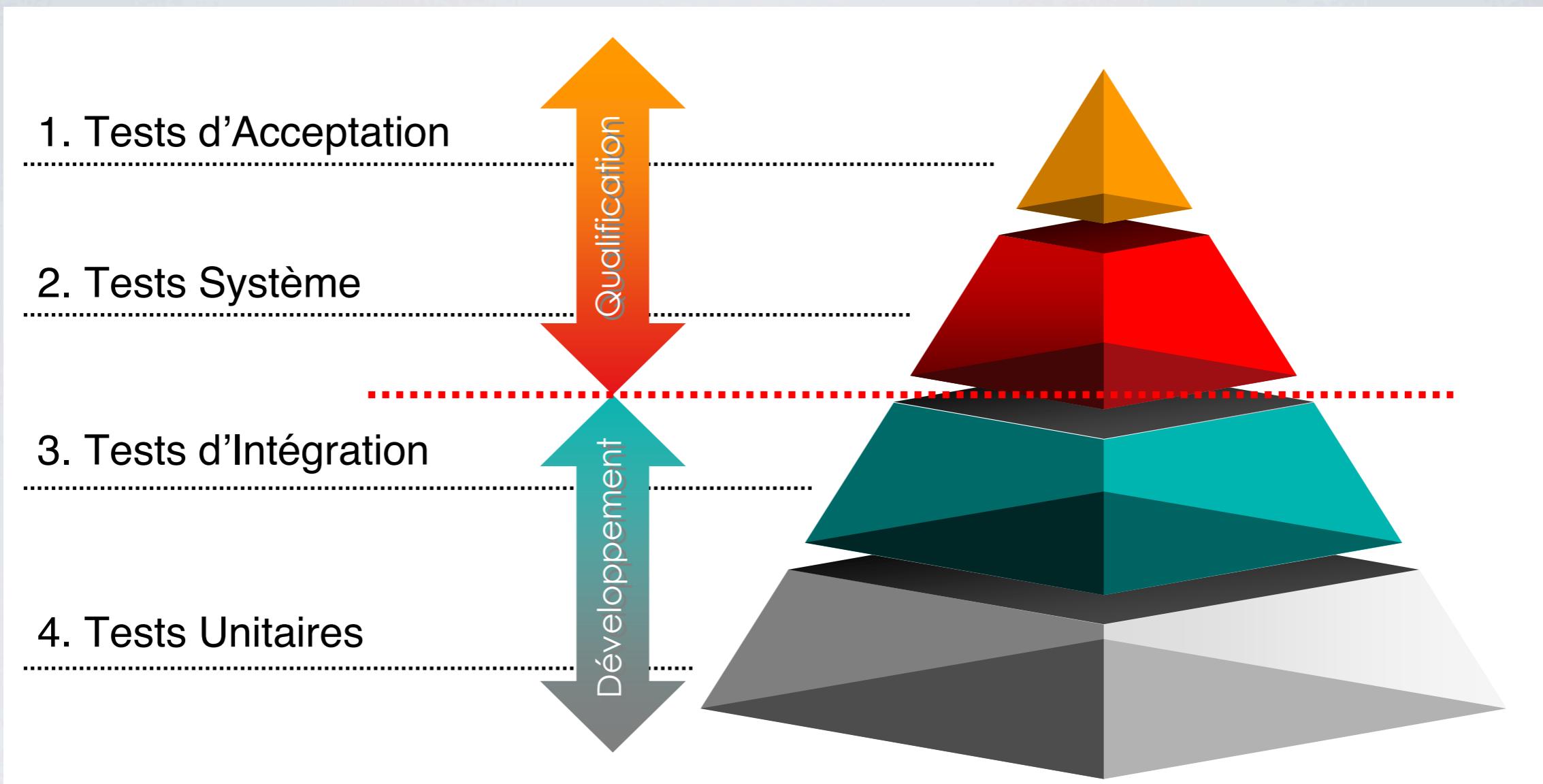
- Tests Système - System Testing
 - Tester que les composantes de la solution respectent les exigences spécifiées qui les concernent [VER (CMMI)]
 - Démontrer que la solution satisfait à l'utilisation prévue lorsqu'elle est placée dans l'environnement cible [VAL (CMMI)]
 - Autres appellations : TS, Solution Testing, Qualification interne, Qualification Fournisseur, Validation interne Recette interne
 - Tests fonctionnels basés sur
 - Cas d'Utilisation & Activités et Interactions « automated »
 - Process et Interactions métiers
 - Tests non fonctionnels basés sur
 - Non-functional Requirements
 - Réalisés par l'Equipe de test indépendante de l'équipe de réalisation

NIVEAUX DE TEST

- Tests d'Acceptation - Acceptance Testing
 - Déterminer que la solution est prête à être déployée au travers de son test par les utilisateurs finaux ou leur représentants
 - i.e. Démontrer que la solution satisfait à l'utilisation prévue lorsqu'elle est placée dans l'environnement cible VAL (CMMI)
 - Autres appellations : TA, Qualification Client, Qualification externe, Recette / UAT (User Acceptance Testing)
 - Tests fonctionnels et non fonctionnels
 - Réalisés par l'Equipe de test client
 - Comprend les Tests Alpha et Beta dans le monde de l'édition
 - Les tests Alpha (α) sont exécutés sur le site de l'organisation effectuant le développement
 - Les tests Beta (β) ou tests sur le terrain sont exécutés par des personnes sur leurs sites propres.
 - Les deux sont exécutés par des clients potentiels et non par des développeurs du produit

LES NIVEAUX DE TEST TU ET TI DÉPENDENT DE LA COMPOSANTE DE LA SOLUTION OBJET DU TEST !





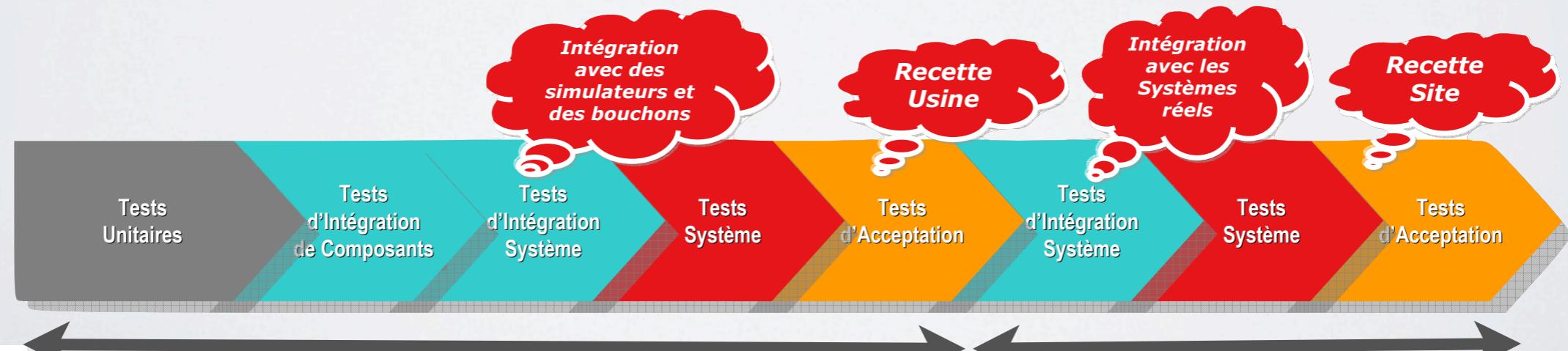
PYRAMIDE VERTUEUSE : ON OPTIMISE L'EFFORT DE TESTS DE
SORTE DE NE PAS RETESTER À CHAQUE NIVEAU CE QUI A
ÉTÉ TESTÉ DANS LES NIVEAUX INFÉRIEURS !

CHRONOLOGIQUEMENT À ADAPTER EN FONCTION DU PROJET

- De la version «tout usine» du cycle en V ...

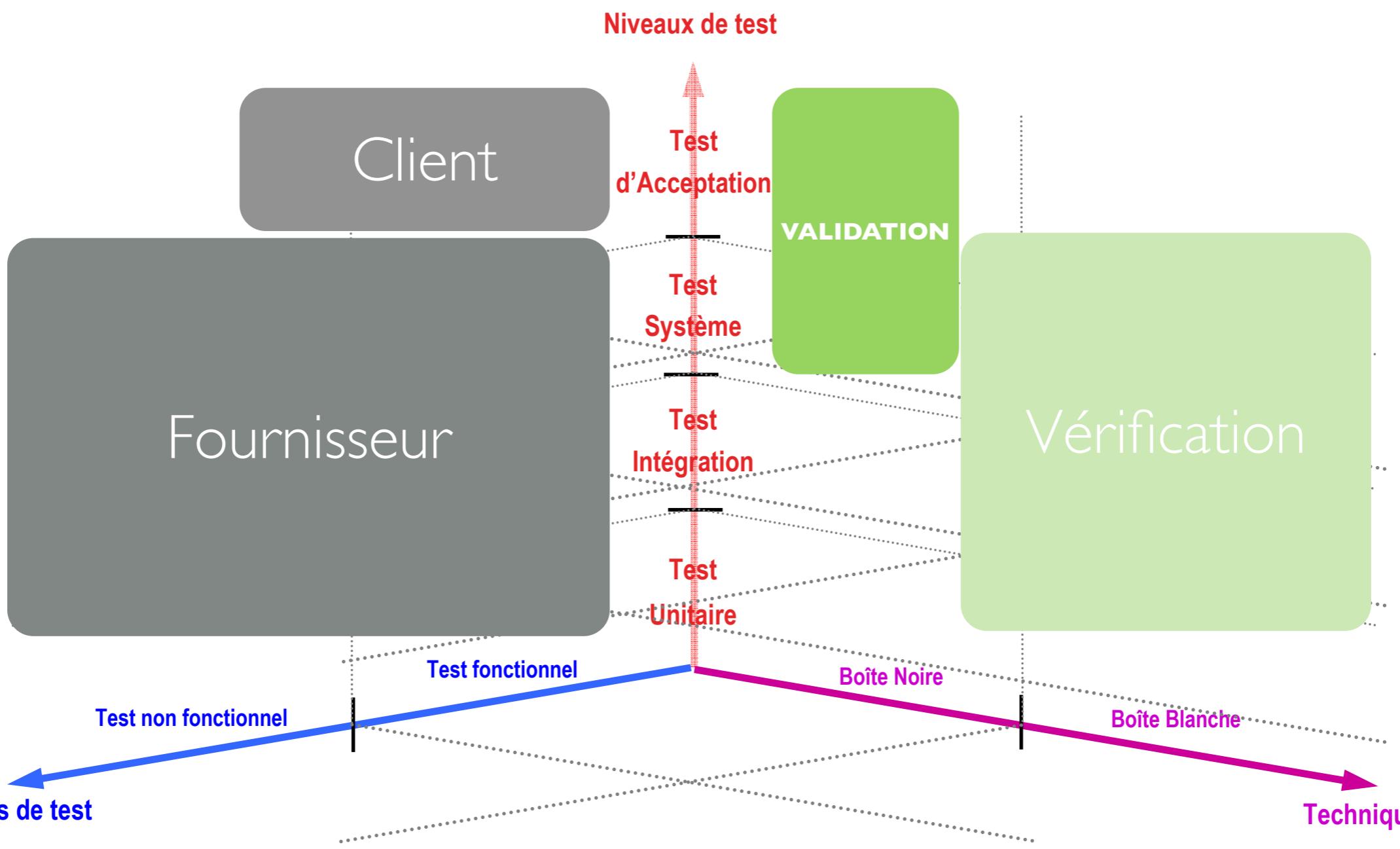


- ... A une version «mixte»



OBJECTIFS DE LA DISCIPLINE

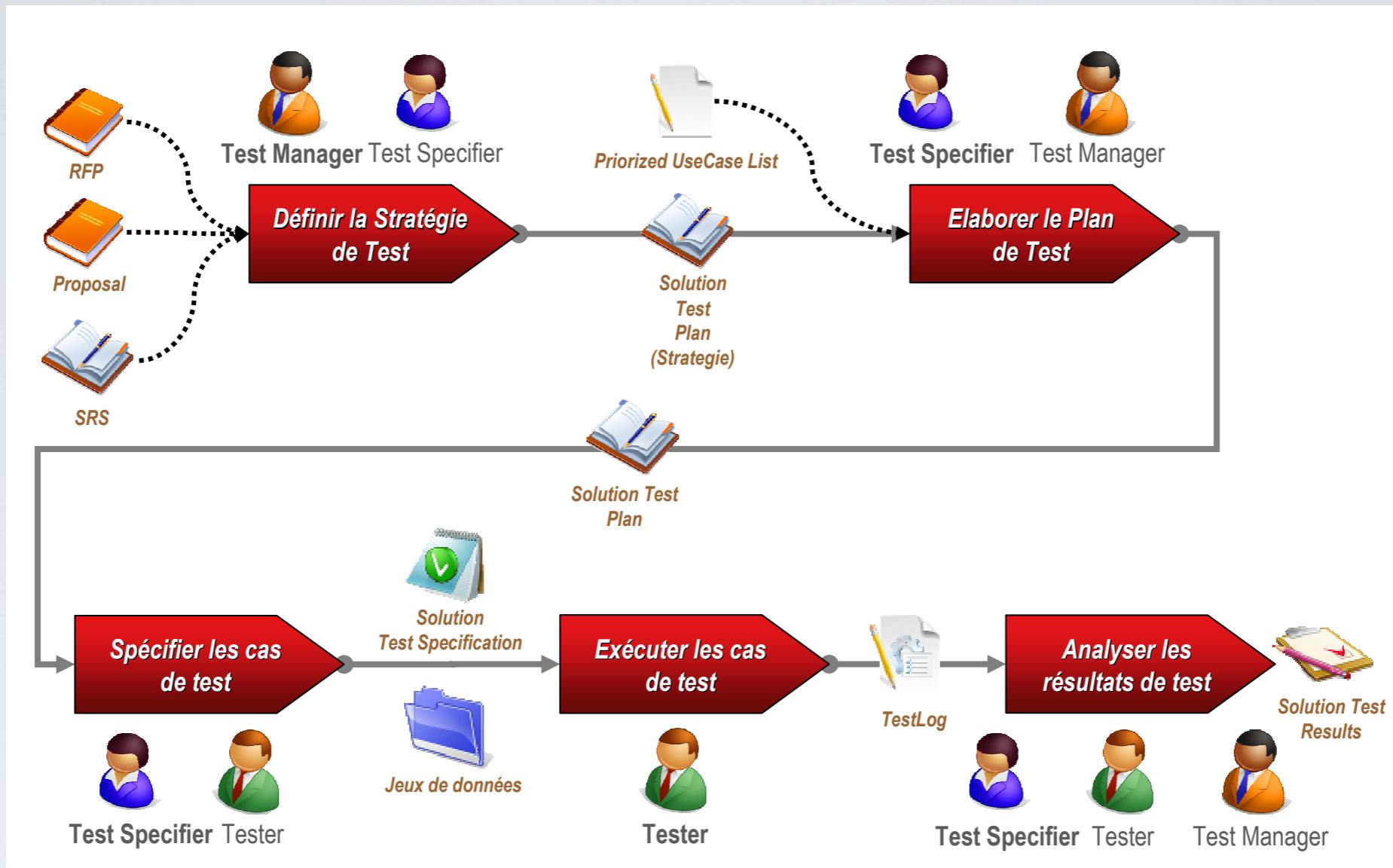
- Vérifier la conformité de la Solution aux exigences fonctionnelles et non fonctionnelles (Vérification) de la Solution
- Trouver les défauts
- Evaluer la qualité de la Solution produite en réponse à un besoin (Validation)



TESTS AUTOMATISÉS

- Critères : criticité, volumétrie, fréquence et durée
 - Test de performance, de charge, de volume, de stress
 - Ex : un test automatique = 8 tests manuels (équivalence en charge)
- Automatisable, approche itérative
 - Tests d'upgrade mineure de non-régression (fonctions de base)
 - Tests demandant des informations détaillées internes à la Solution
- Non automatisables
 - Tests ponctuels
 - Tests qui vérifient la facilité d'utilisation de la Solution
 - Tests n'ayant pas des résultats vérifiables de façon automatique
 - Tests des impressions
 - Tests de fonctions encore instables ou sujettes à des évolutions





DISCIPLINES DE TEST

LA STRATÉGIE DE TEST

Objectifs

- Valider la conformité des exigences
- Définir la stratégie de qualification de la version
- Elaborer le planning détaillé de la qualification de la version
- Identifier les moyens spécifiques nécessaires à la qualification de la version (humain, matériel, logiciel)



Activités

- **Conception de la stratégie de qualification**
 - Prendre en compte les exigences de la version et valider leur conformité par rapport aux critères qualité attendus sur ces exigences
 - Proposer un plan de mise en conformité des exigences, le cas échéant
 - Identifier les priorités des exigences
 - Analyser les risques métiers sur le périmètre de la version (méthode rapide MoSCoW et analytique)
 - Déterminer la profondeur de test associée à chaque élément constitutif de la version
 - Définir les conditions d'arrêt des tests de chaque élément constitutif de la version
 - Rédiger le plan stratégique de qualification
- **Road map de qualification de la version**
 - Identifier les besoins spécifiques à la version
 - ressources humaines
 - environnements de qualification
 - Etablir la macro planification de la qualification

Pré-requis et entrants

- Fourniture par le Client du cahier des exigences validées pour la version (*exigences fonctionnelles et non fonctionnelles*)
- Réunion de présentation des exigences par le Client
- Fourniture par le Client des jalons clés de la version (fin spécifications, livraison Editeur/Réalisateur, Mise en production)



Acteurs

Client

- Chef de Projet



Fournisseur

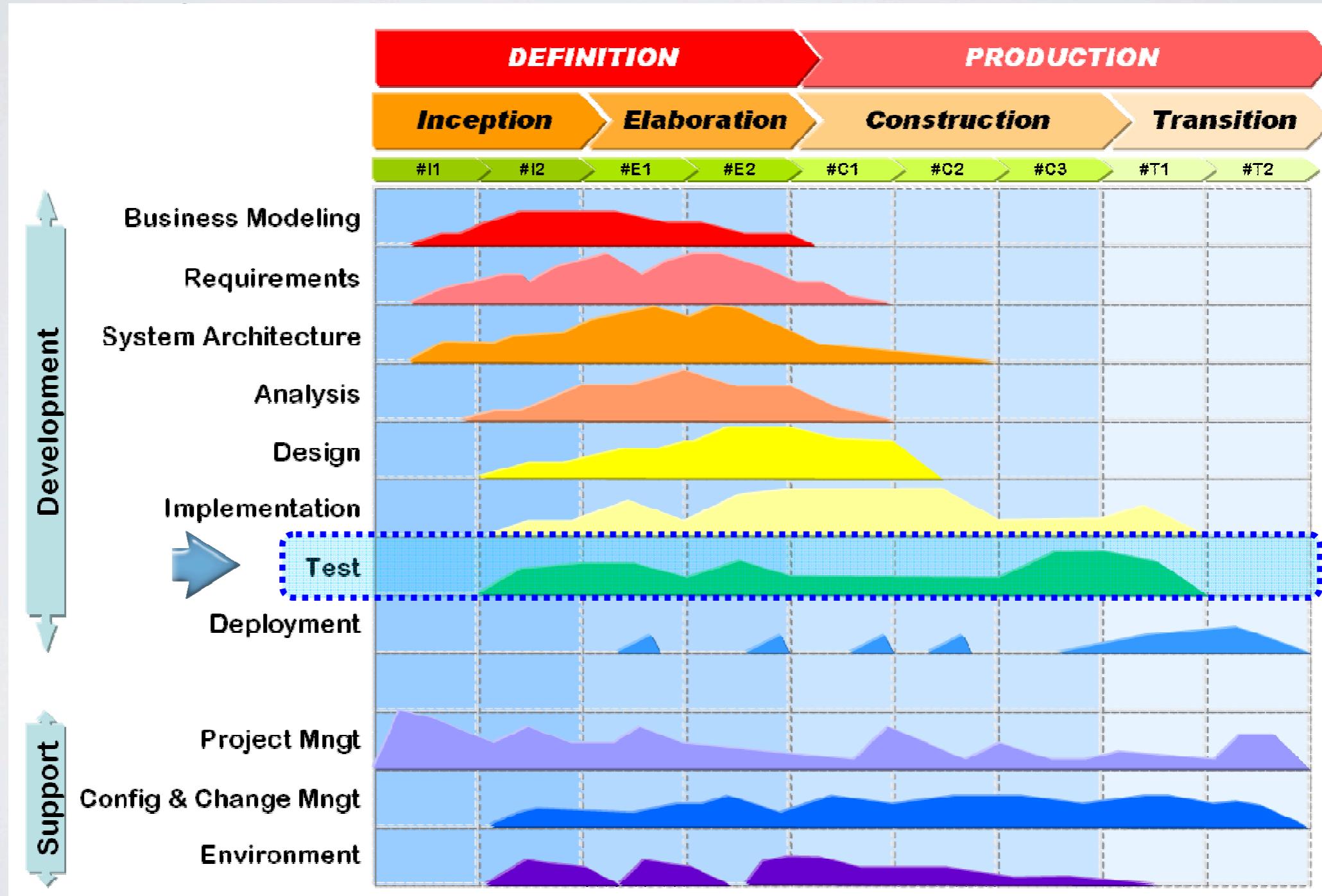
- Chef de Projet

Livrables

- Dossier de stratégie de qualification de la version
- Road map de qualification de la version

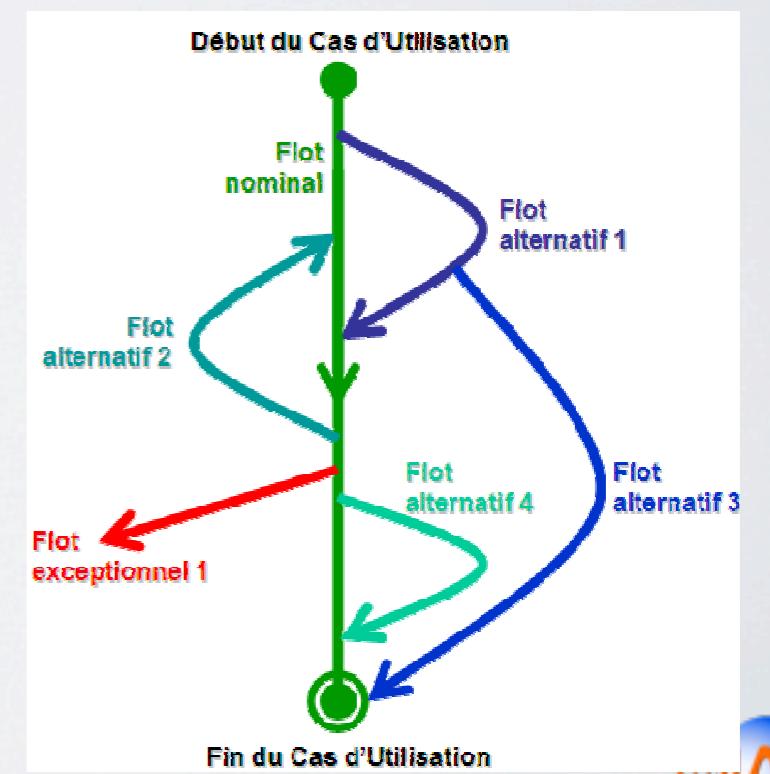


DIAGRAMME DE BOSES

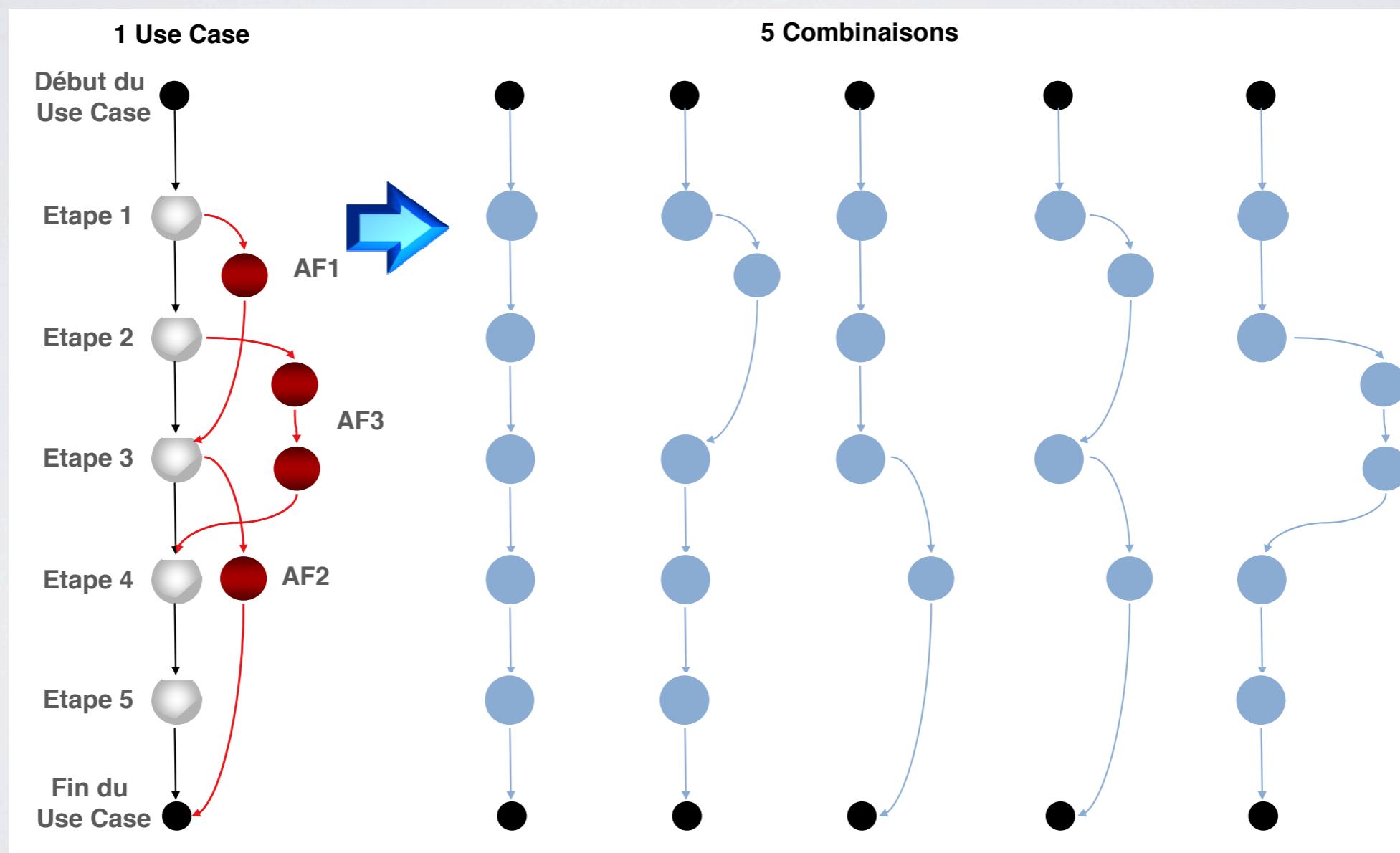


LES FLOWS D'UC

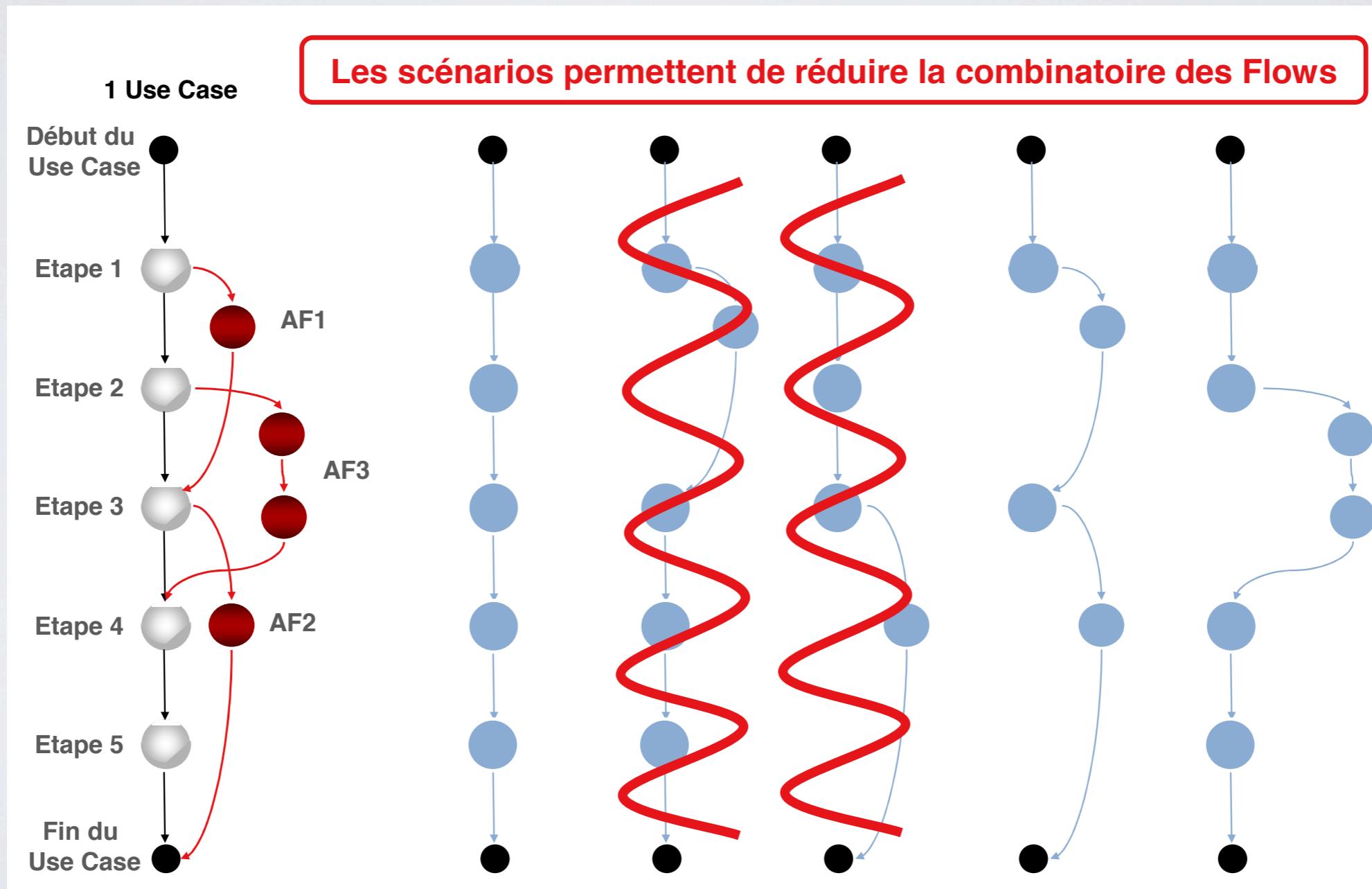
- Un Use-Case raconte, sous forme d'histoires, la manière dont l'acteur utilise la Solution, et le Système lui répond
- Les interactions entre l'acteur et le Système sont organisées selon des Flows. 3 types de Flows :
 - Les flows nominaux => normalement, l'objectif est atteint
 - Les flows alternatifs => **variations** par rapport au flot nominal permettant d'atteindre l'objectif (total ou partiel)
 - Les flows exceptionnels => **variations** qui correspondent à une situation anormale qui ne permet pas d'atteindre l'objectif



SCÉNARIOS



SCÉNARIOS



SCÉNARIOS ET CAS DE TEST

- Scénario de Test
 - Pour 1 Use-Case donné
 - Scénarios de Use-Case : nombre de combinaisons très élevé dont certaines n'ont pas de sens => on en retient qu'un sous-ensemble
 - 1 Scénario de Use-Case retenu => 1 Scénario de Test
 - Pour 1 Scénario de Test donné, il y a plusieurs Cas de Test qui dépendent
 - Des valorisations (données)
 - Des conditions d'exécution Cas de test
- Cas de Test
 - Un ensemble de valeurs d'entrée, de pré-conditions d'exécution, de résultats attendus et de post-conditions d'exécution, développées pour un objectif ou une condition de tests particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique [ISTQB/IEEE 610]

UN CAS DE TEST COMPREND DES ÉLÉMENTS BASIQUES

- La valorisation du Use-Case ou des interactions / activités « automated »
 - Cela répond à la question « De quoi ai-je besoin ? »
- Les étapes : les étapes du Use-Case (Use-Case Steps ou Transaction) ou le workflow des interactions / activités « automated » permettant d'exécuter le test
 - Cela répond à la question « Que dois-je faire ? »
- Le ou les résultats attendus
 - Cela répond aux questions
 - Qu'est-ce que j'attends comme résultat(s) ?
 - Comment je sais que le test est passé et est ok

ÉTAPES DE TEST

- **Une étape de test est constituée d'au moins**
 - Une référence d'ordonnancement
 - Etape 10
 - Une description
 - décrit la (ou les) action(s) à réaliser par le testeur
 - Les résultats attendus pour l'étape
 - décrit les résultats que l'on doit constater suite à l'exécution des actions présentées dans la partie description
 - décrit les moyens de contrôle pour vérifier les résultats attendus
- **Si une étape échoue ou n'est pas exécutée**
 - Le Cas de Test et donc les exigences qu'il couvre sont « non validés »

SCÉNARIO VS CAS DE TEST

- **Scénario de test**

- L'Utilisateur insère sa carte bleue dans le lecteur
- Le Système lit les informations de la carte
- Le Système demande à l'Utilisateur son code de carte bleue
- L'Utilisateur saisit son code
- Le Système demande à la banque si les informations de la carte et le code sont cohérents
- La Banque confirme la validité des informations

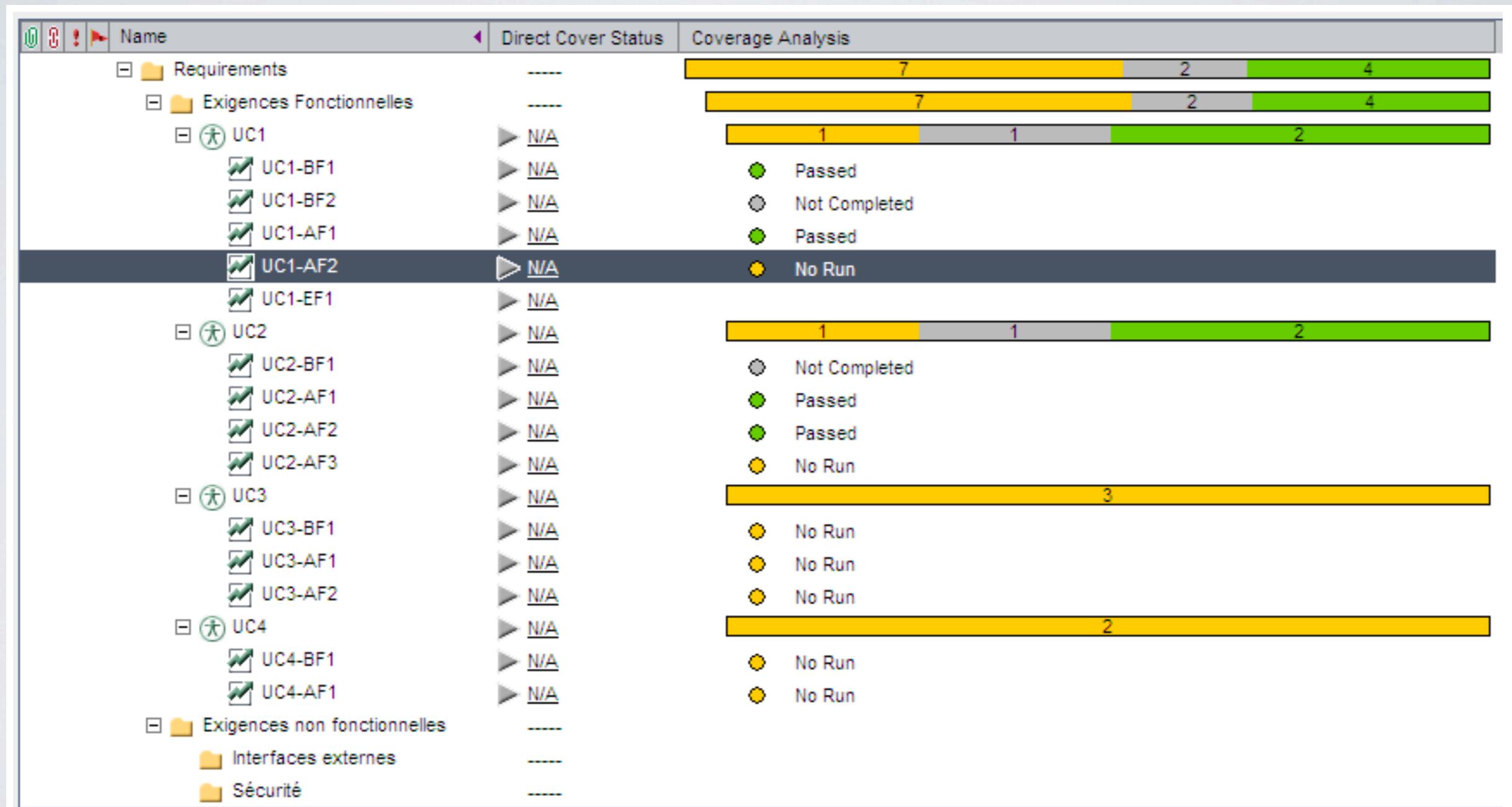
Générique

- **Cas de test**

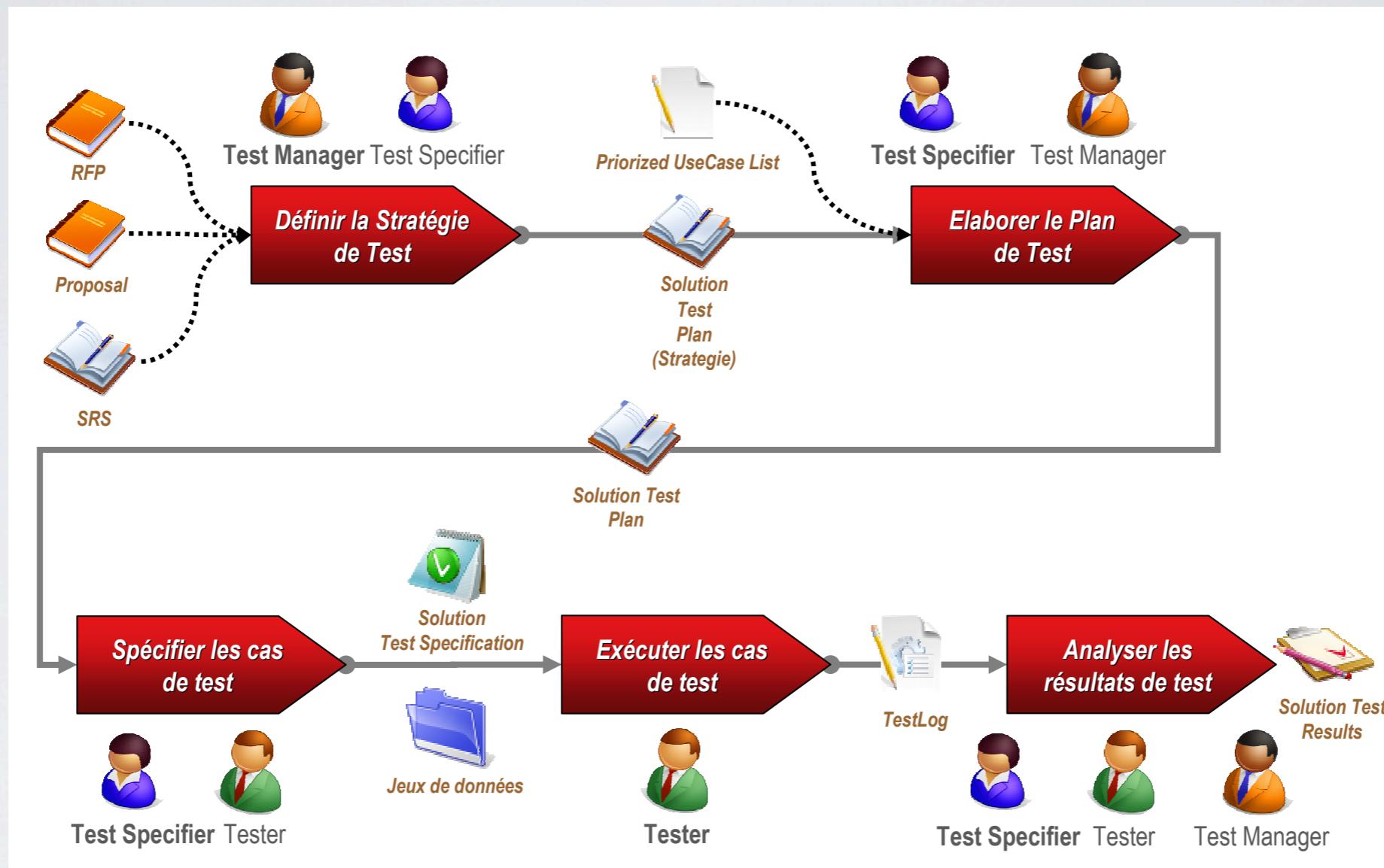
- L'Utilisateur insère la carte bleue **VISA 4978 1234 1234 1234** dans le lecteur
- Le Système lit les informations de la carte
- Le Système demande à l'Utilisateur son code de carte bleue
- L'Utilisateur saisit son code : **1234**
- Le Système demande à la banque « **Caisse d'Epargne** » si les informations de la carte et le code sont cohérents
- La Banque confirme la validité des informations

Valorisé

PENSEZ À LA TRAÇABILITÉ



VUE D'ENSEMBLE



A RETENIR

- Prévoir une couverture de test (matrice de traçabilité).
- Identifier et tester en premier les composants critiques.
- Couvrir tous les types de tests.
- Prévoir les moyens nécessaires à l'avance
- Donner de la visibilité sur l'avancement et le résultat des tests.
- Établir les critères d'arrêt des tests.
- Commencer le plus tôt possible les tests d'intégration (intégration en continue).
- Se conformer au plan de tests et dossiers de tests.
- Trouver des erreurs avant la livraison.
- Requalifier la version avant chaque livraison (scénario de non régression express).