Cours 2 : Cycle de vie « classiques »

Modèle en cascade

Cycle en V,

Cycle en Spirale

Intervenants clés

Cycle de vie nominal d'un logiciel

Loi de Brooker :
dix grammes d'abstraction valent des tonnes de bricolages.

Loi de Myers :
on passe la moitié de son temps à refaire ce que l'on n'a pas eu le temps de faire correctement.

Cycle de vie : introduction

- **Ces cycles prendront en compte toutes les étapes de la conception, de la réalisation d'un logiciel (pas la maintenance)**
  - « **Début** »
    - demande d'informatisation d'un processus
      - Demandeur : « Je veux que nous développions ça »
      - Cahier des charges
    - Souvent reprise d’un existant
      - Ancienne version d’un logiciel proche obsolète
      - Rétroingénieurie (reverse engineering)
  - « **fin** » (très provisoire)
      - l'écriture et la mise au point d'un logiciel
      - livraison au demandeur.

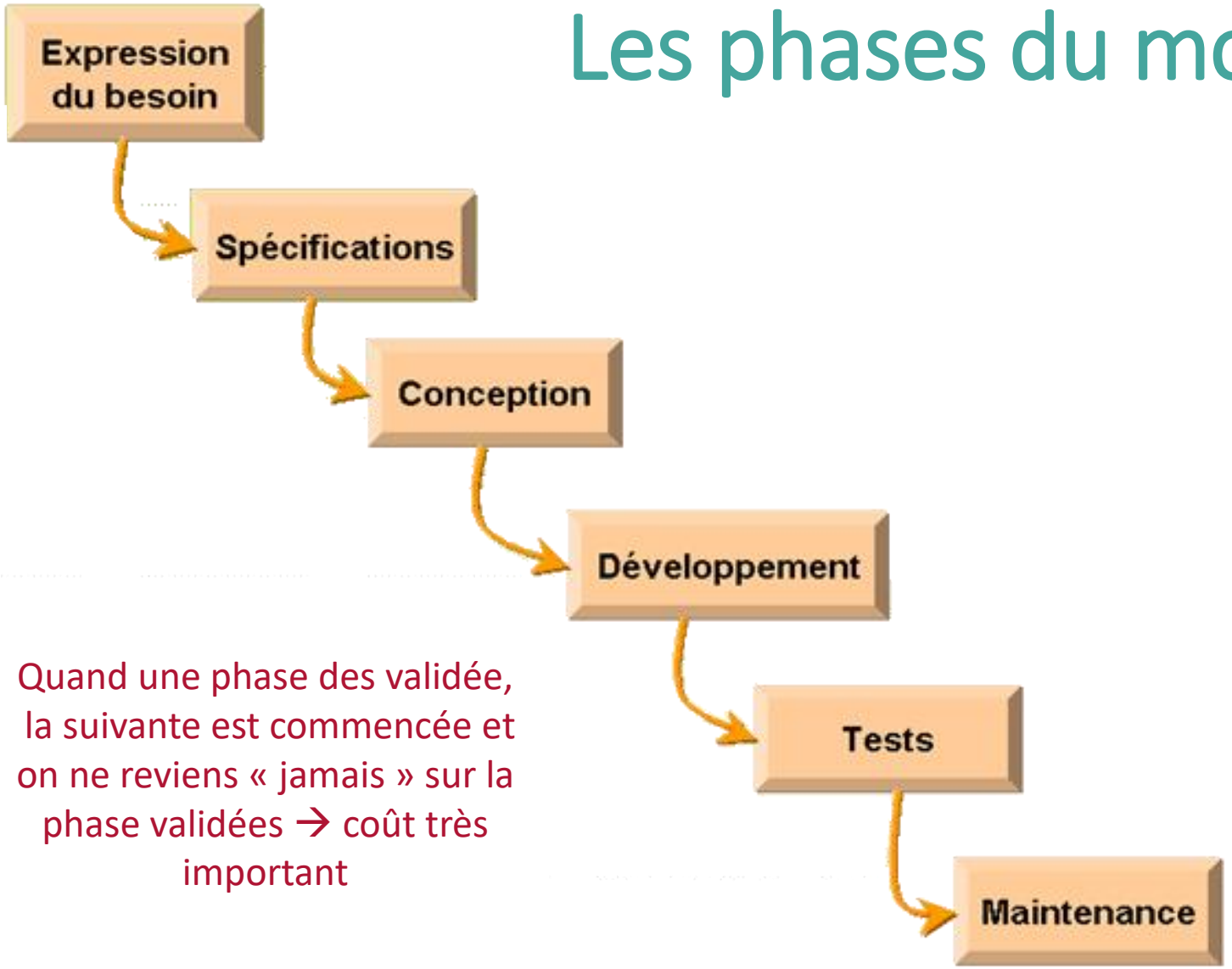
Cycle de vie

- **Décomposition en étapes à valider et à enchaîner chaque étape produit des résultats**
- **Type de cycle de vie**
  - Modèle en cascade (1970)
  - Cycle en V (1984)
  - Cycle en Y
  - Cycle en spirale
  - Méthode agile (2001) : Scrum
- **Aujourd'hui, les projets informatiques sont réalisés à :**
  - 40% à l'aide de méthodes classiques type cycle en V
  - 60 % à l'aide de méthodes Agile type Scrum

Modèle en cascade

- **Origine**
  - Industrie du **BTP**.
- **Hypothèses**
  - on construire la toiture après les fondations ;
    - des modifications tardives sur les premières phases ont des conséquences financières très importantes besoin de reprendre les phases
  - Si nous modifions les fondations alors que la toiture est déjà posée
    - Coût excessif
    - **PAS de RETOUR ARRIÈRE**

Les phases du modèle en cascade



Quand une phase des validée, la suivante est commencée et on ne reviens « jamais » sur la phase validées → coût très important

Exemple : Créer un logiciel de facturation pour un magasin de bricolage

1. **Expression**
   - Quelles sont les besoins du magasin (métier) ? (cahier des charges)
2. **Spécification**
   - Quelles sont les réponses informatiques ?
     - Architecture du SI : une seule ou plusieurs machines ?
     - Base de données ou fichier de données
     - Langage(s) de développement
     - Fonctions développées
3. **Conception**
   - Quelles classes implantées
   - Quelles sont les entrées, les sorties, les contraintes d'un des besoins ? (conception d'une fonction)
4. **Développement**
   - Développement des méthodes des classes, des requêtes d'accès aux bases de données, des processus de sauvegarde…
5. **Test**
   - Vérification que le SI répond bien aux besoins
6. **Maintenance**
   - Correction des anomalies
   - Évolution du SI pour intégrer de nouveaux besoins simples

Exercice : Créer un logiciel de facturation pour un magasin de bricolage

- Quelles sont les données qui doivent être gérer? (conception de la base de données ou des fichiers)
- Qui code (une personne, une équipe) avec quelle chartes de codage ?
- Comment tester le logiciel ?
- Comment livrer le logiciel, l'installer et le déployer ?
- Comment le maintenir (nouvelles fonctions, correction d'erreurs…) ?

Fin de chaque phase

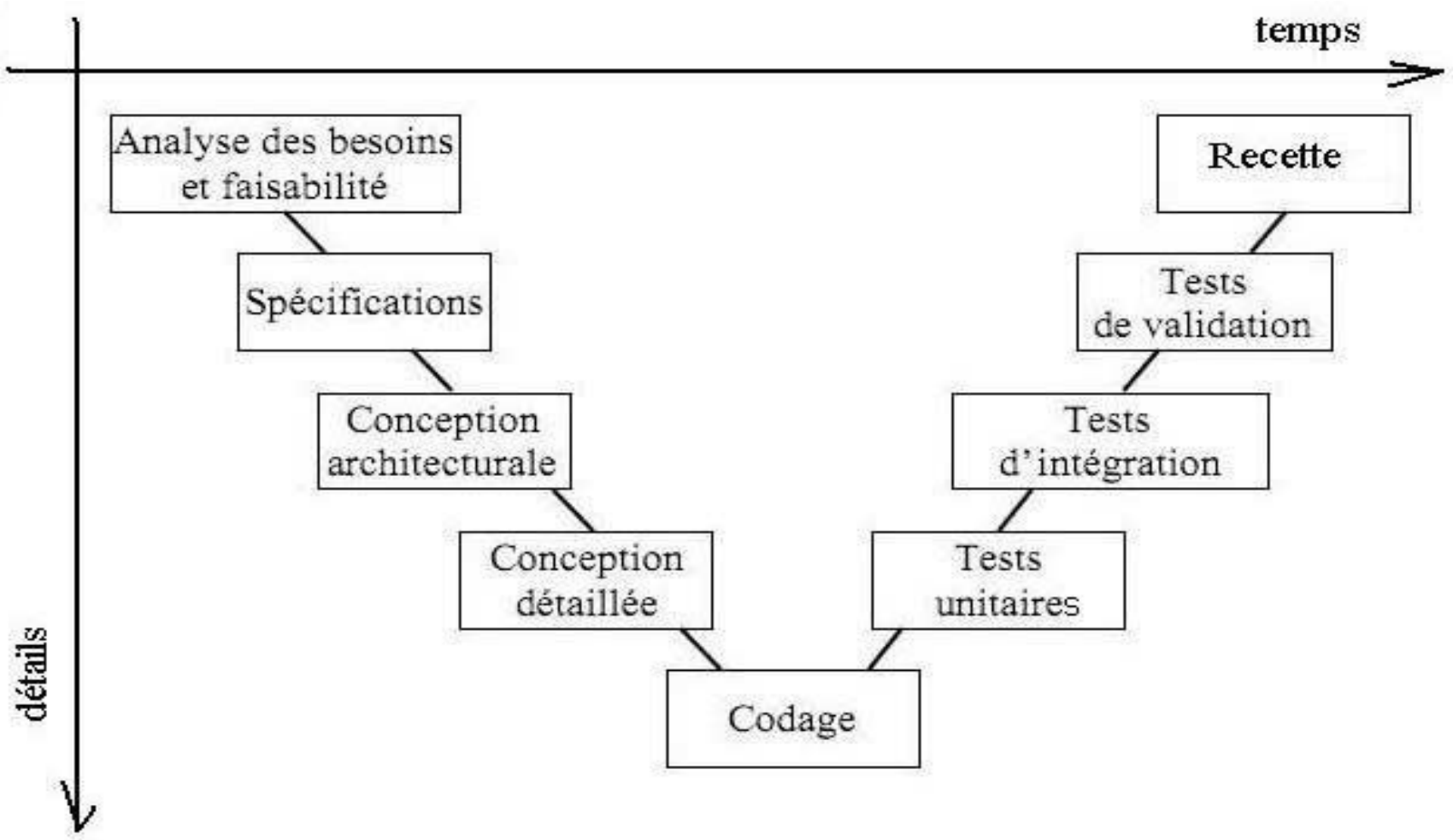
- **Production d'un livrable défini au préalable ;**
  - Document(s)
  - Logiciel(s)
  - Résultats de test(s)
- **Doit se terminer à une date précise ;**
- **Ne se terminer que lorsque les livrables sont jugés satisfaisants**
  - étape de validation-vérification à la fin de chaque tâche

Cycle en V

Cycle en V

- Le cycle de vie est décomposé en phases de développement:
  - Analyse des besoins et faisabilité
  - Spécification fonctionnelle
  - Conception architecturale
  - Conception détaillée
  - Codage
  - Test unitaire (TU)
  - Test d'intégration (TI)
  - Test de validation : recette usine, validation usine, VAU
  - Test d’acceptation : vérification d'aptitude au bon fonctionnement, VABF, recette

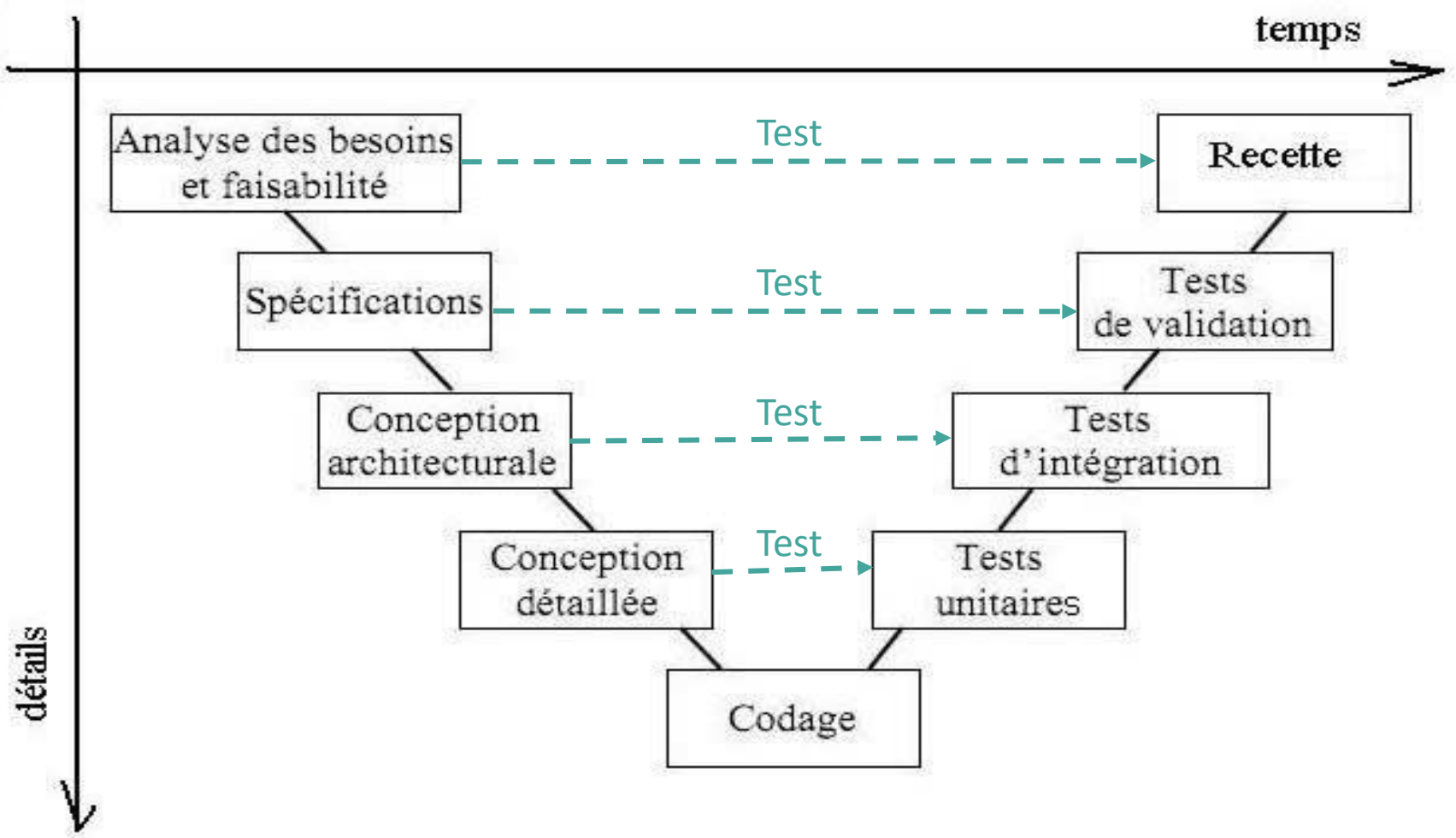
Cycle de vie en V



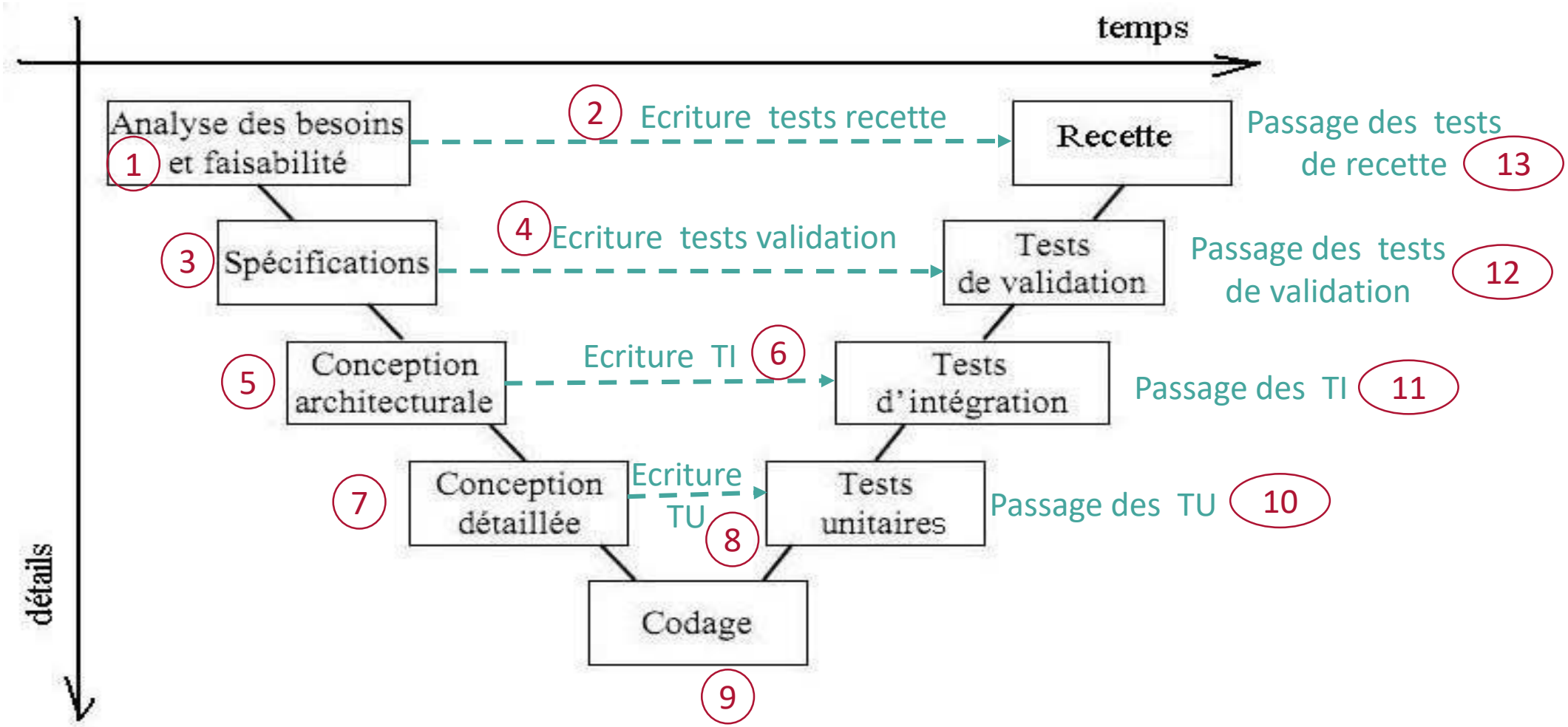
Cycle en V

- Ces phases sont échelonnées dans le temps.
- Une phase se termine
  - par la remise d'un (ou plusieurs) document(s) validé(s)
  - Par la revue de cette phase
  - Lorsque le propriétaire et le développeur sont d'accord.
- Une phase ne peut commencer que lorsque la précédente est terminée.
- La décomposition en phase de développement permet donc le suivi du projet par l'utilisateur.

Cycle de vie en V



Ordre des étapes (Ecriture et Réalisation des tests)



Les phases

- Les premières phases (descendantes) permettent de **décomposer** l'ensemble du projet pour simplifier la phase de codage.
- Les phases suivantes **recomposent** l'ensemble du logiciel en le testant du détail vers l'ensemble.

Décomposer pour gérer la complexité

- **Diviser pour résoudre**
  - Aspects techniques
    - en s'appuyant techniquement sur la modularité
    - Encapsulation et masquage d'information,
  - Aspects organisationnels
    - S’organiser au-delà de la réalisation : méthodes
      - Analyse, conception
      - Validation et vérification
    - problème de gestion de ressources (humaines, etc.) et de synchronisation entre tâches
    - Ingénierie de la conduite de projet = compromis entre
      - Respect des délais
      - Respect des coûts
      - Réponse aux besoins/assurance qualité

Analyse des besoins et faisabilité

Besoins

- Le cahier des charge regroupe les besoins métiers du client
- Les spécifications sont une réponse informatique
- A l ’issue de cette phase :
  - Client et fournisseur sont d ’accord sur le produit à réaliser (IHM comprise)
  - Prend en compte des contraintes de faisabilité

Faisabilité

- **Economique**
- **Technique**
  - Risques de développement
  - Disponibilité des ressources
  - Technologie nécessaire
- **Légale**

Spécification fonctionnelle

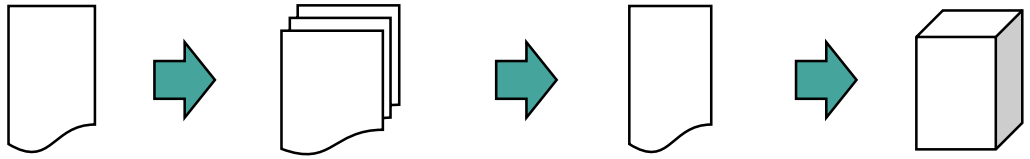
- **La spécification fonctionnelle est la description des fonctions (modules) d'un logiciel en vue de sa réalisation.**

- **La spécification fonctionnelle décrit dans le détail la façon dont les exigences seront prises en compte.**
  - Deux niveau
    - Spécification fonctionnelle générale ou architecture métier
    - Spécification fonctionnelle détaillée

Spécification fonctionnelle

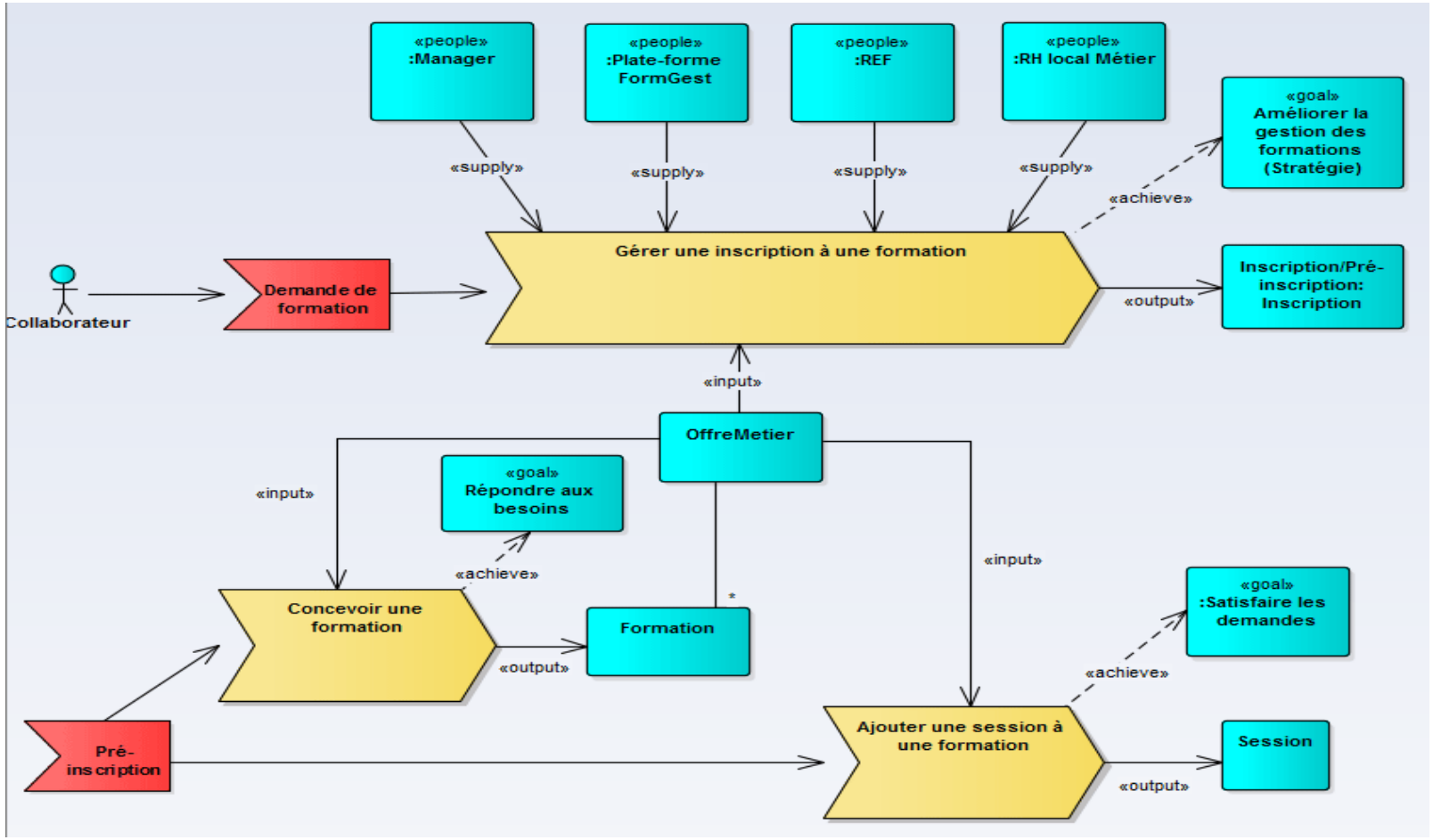
- Le point de vue statique
  - Le point de vue statique consiste à
    - découper le logiciel en modules si l'on utilise un langage classique
      - Pour notre exemple :
        - Un module gestion des clients
        - Un module gestion des articles
        - Un module création de facture
        - Un module bilan des ventes
    - de définir les classes d'objets du système si l'on utilise un langage orienté objet

Spécification fonctionnelle



- **But**
  - réduire les délais
  - fractionner de manière récursive le projet en éléments traités par plusieurs équipes
- **Objectifs**
  - Le maximum d'autonomie entre les sous-systèmes
    - regroupement en fonctions
      - des caractéristiques communes
      - de la capacité des équipes
- **Règles**
  - Définir les protocoles de communication entre les éléments
    - interfaces qui vont autoriser l'intégration
  - Le logiciel résultera de l'assemblage de ses parties de manière récursive
    ⇒ **Intégration**

Spécification fonctionnelle dynamique : exemple gestion de formation



http://www.urbanisation-si.com/agl-atelier-de-genies-logiciel-les-ministeres-recommandent-enterprise-architect-de-sparx-system

Spécification fonctionnelle détaillée

- **Exercice**
  - Pour le module création d'un client, il faut enchaîner plusieurs étapes pour :
    - Affecter un numéro au client
    - Saisir le nom, l'adresse et l'email
    - Vérifier que le client n'existe pas
    - Vérifier que l'email (non obligatoire) est valide
    - Vérifier que l'adresse est valide
  - Comment pouvez vous enchainer ces étapes pour cette fonction ?

Spécifications des fonctionnelle

- Besoin de choisir des solutions **évolutives**
  - Exemple infrastructure GSM Nokia
    - 50% des requirements (besoins numérotés) ont changé *après* le gel du cahier des charges
    - 60% de ceux-ci ont changé au moins 2 fois!
    - Hélas c'est classique
  - L'approche Orientée Objet (modèle et langage) est plus flexible

Spécifications des fonctionnelle

- Pistes de réponses
  - D’abord modéliser ce qui est stable dans un système
    - Entités et événements selon Jackson
  - Ensuite ajouter les fonctionnalités
    - C’est la partie visible, donc ce qui bouge le plus
    - Exemple : nouveaux rapports, …

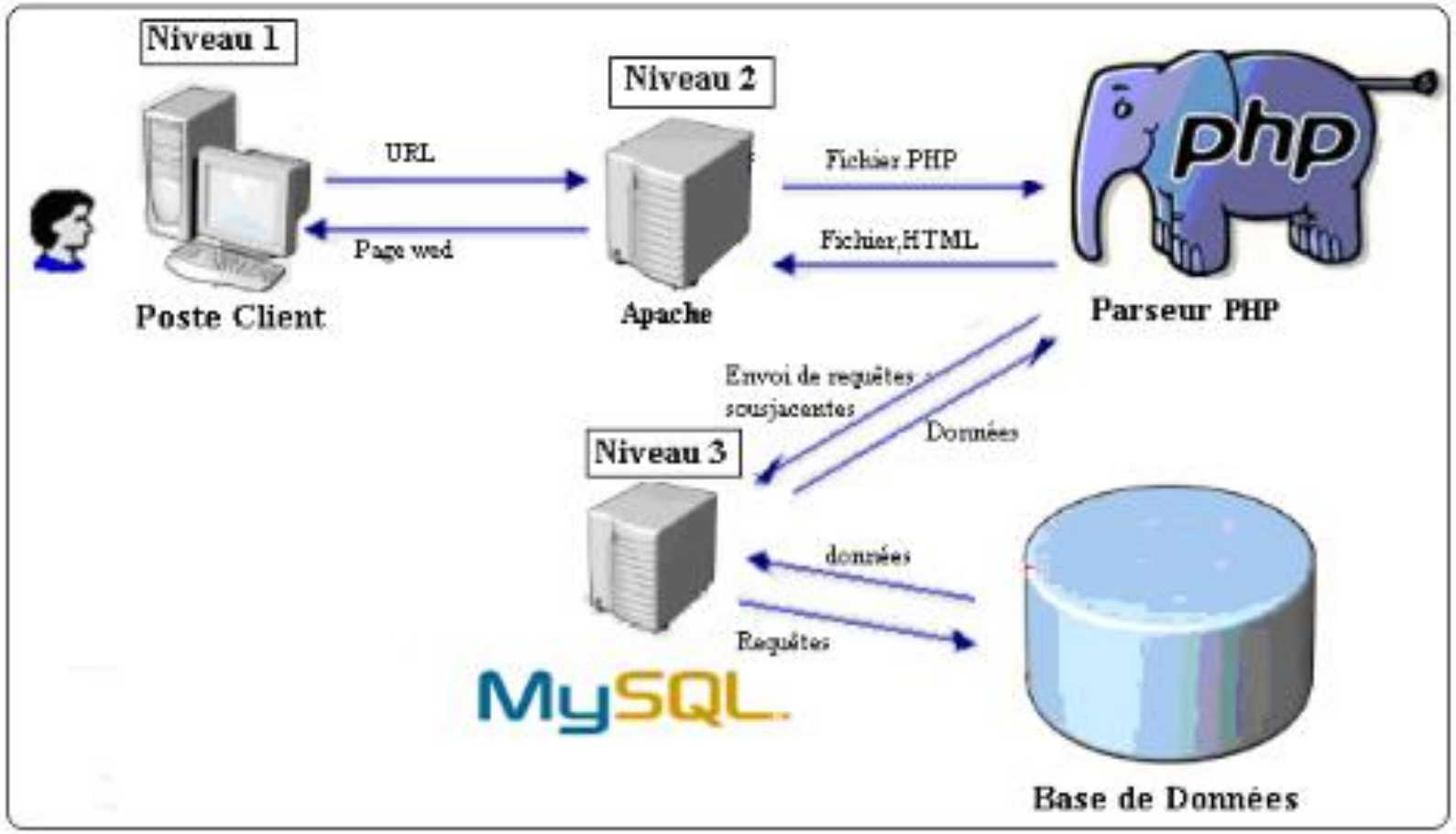
Spécifications fonctionnelle

- Pistes de réponses (suite)
  - Maintenance traitée comme le développement initial
  - Aspects organisationnels
    - Traçabilité
    - Gestion contrôlée des changements

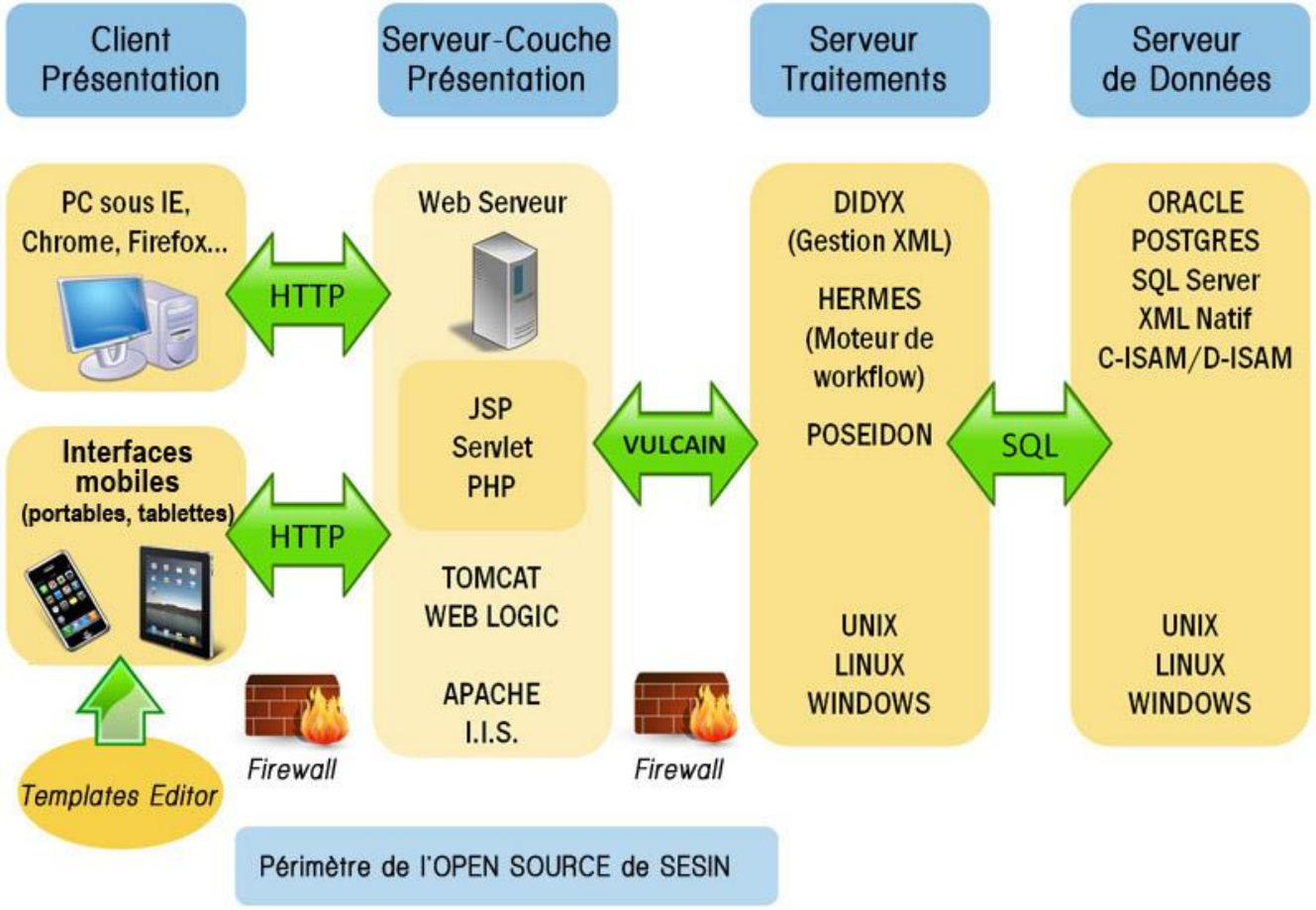
Conception architecturale

- L’architecture logicielle décrit d’une manière symbolique et schématique les différents éléments d’un ou de plusieurs systèmes informatiques, leurs interactions.
- ne décrit pas ce que doit réaliser un système informatique
- Décrit comment il doit être conçu de manière à répondre aux spécifications.

Exemple simple architecture 3-tiers



Exemple détaillée d'architecture logicielle



Codage

- Développement (rédaction) du (ou des) code source d'un logiciel.
- Doit répondre à des normes
- Doit inclure des commentaires
- Le code est compilé ou interprété

Test unitaire

- procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module »), en générale une fonction ou une méthode
- Les test à passer sont défini lors de la phase de conception détaillé

Test unitaire

- **Exercice**
  - Quelle valeur faut-il tester pour la création d'un client ?

Test d'intégration

- **chacun des modules indépendants du logiciel est assemblé et testé dans l'ensemble**
- **Exemple**
  - Tester que le module facture communique bien avec création de client pour une facture à un nouveau client
    - Si nouveau client
      - Appeler le module création de client (vérifie que l'appel se passe bien)
      - Une fois le client créé, retour au module facture avec les données du client

Test de validation

- **Vérifier si toutes les exigences client, décrites dans le document de spécification du logiciel, sont respectées.**
- **Vérification de scénarios prédéfini**
- **Validation performance et robustesse :**
  - Combien de temps faut-il pour obtenir la réponse ?
  - Combien de requêtes par seconde sont possibles ?
  - Combien d'utilisateurs en même temps ?

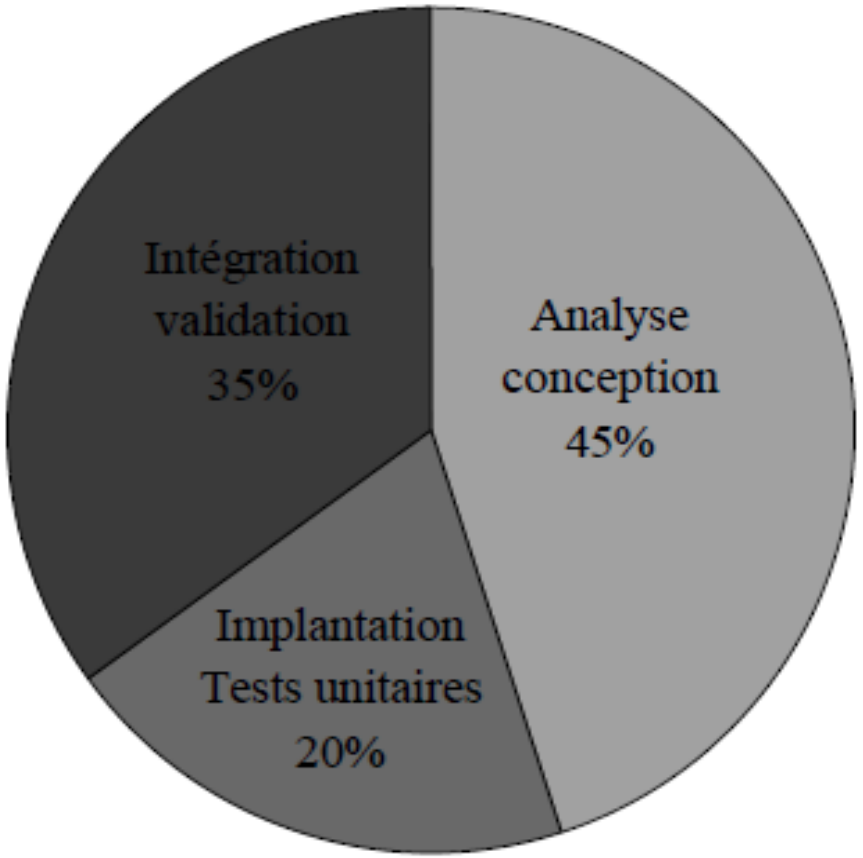
Test d'Acceptation : recette

- vérification d'aptitude au bon fonctionnement en conditions réelles
- se déroule dans les locaux et avec les infrastructures du client : matériel de production, données réelles
- la recette-usine correspond à une période de quelques jours durant laquelle le client procède lui-même à ses propres tests

Test d'Acceptation : recette

- **deux catégories de tests**
  - Recette fonctionnelle
    - Le logiciel doit répondre aux besoins exprimés dans le cahier des charges
  - Recette technique
    - Vérification des caractéristiques techniques (les tests de supervision, de sauvegarde, de performance... et en particulier les tests de respect des exigences d'architecture technique
- **Si le logiciel est validé, le client procède alors à la mise en service opérationnelle → mise en production**

Coûts du développement



Et la proportion
Analyse/Conception
+ Intégration/Validation
tend à augmenter avec
la taille du logiciel

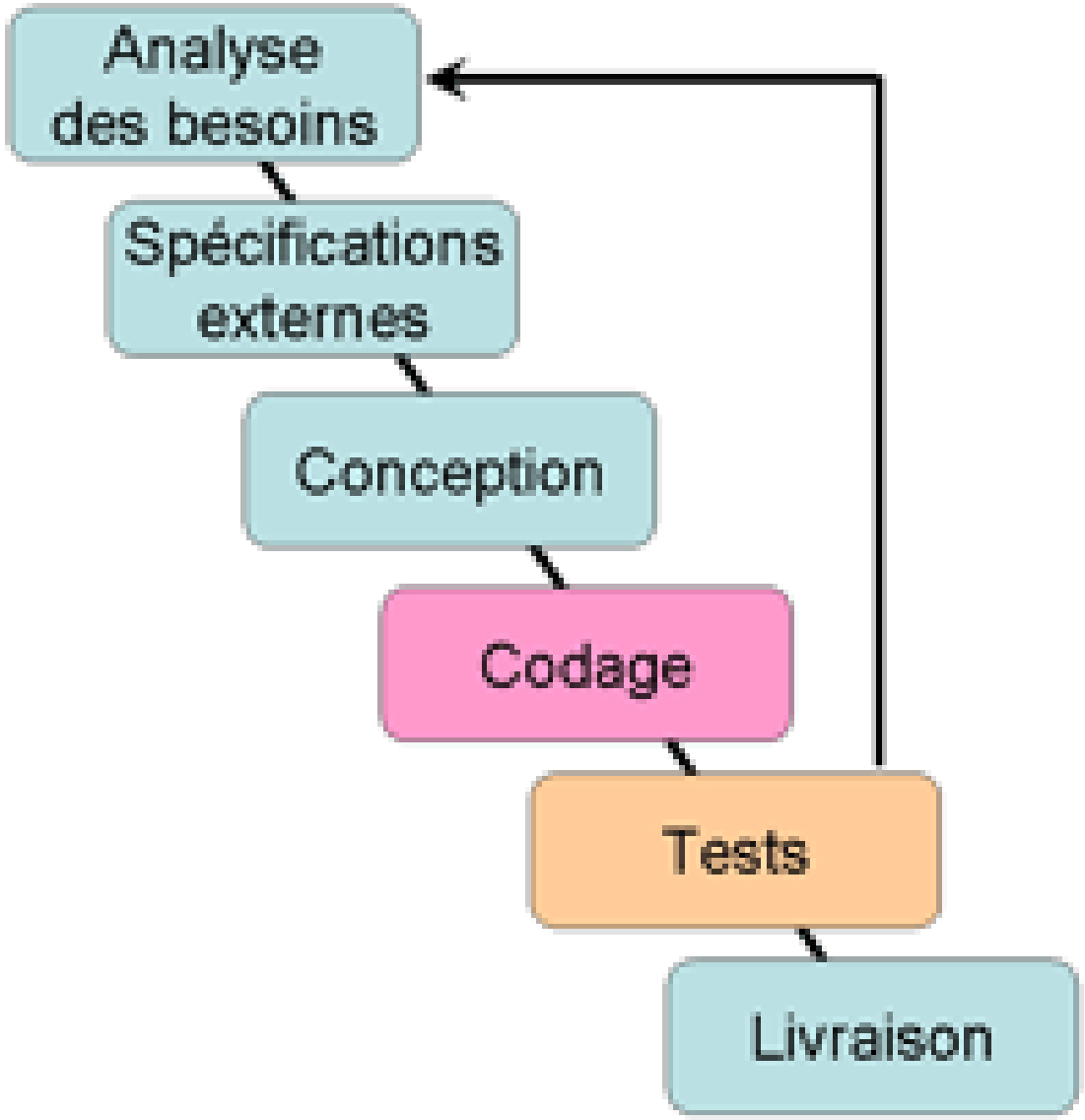
Cycle en spirale

Cycle en spirale

- **Cycle de vie itératif**
- **convient aux projets risqués**
  - consiste à
    - coder un peu
    - tester
    - Recommencer
- **Défini par**
  - Barry Boehm (1988)

Cycle en spirale

- **Il reprend les différentes étapes du cycle en V**
- **Par l'implémentation de versions successives,**
  - le cycle recommence en proposant un produit de plus en plus complet et dur.

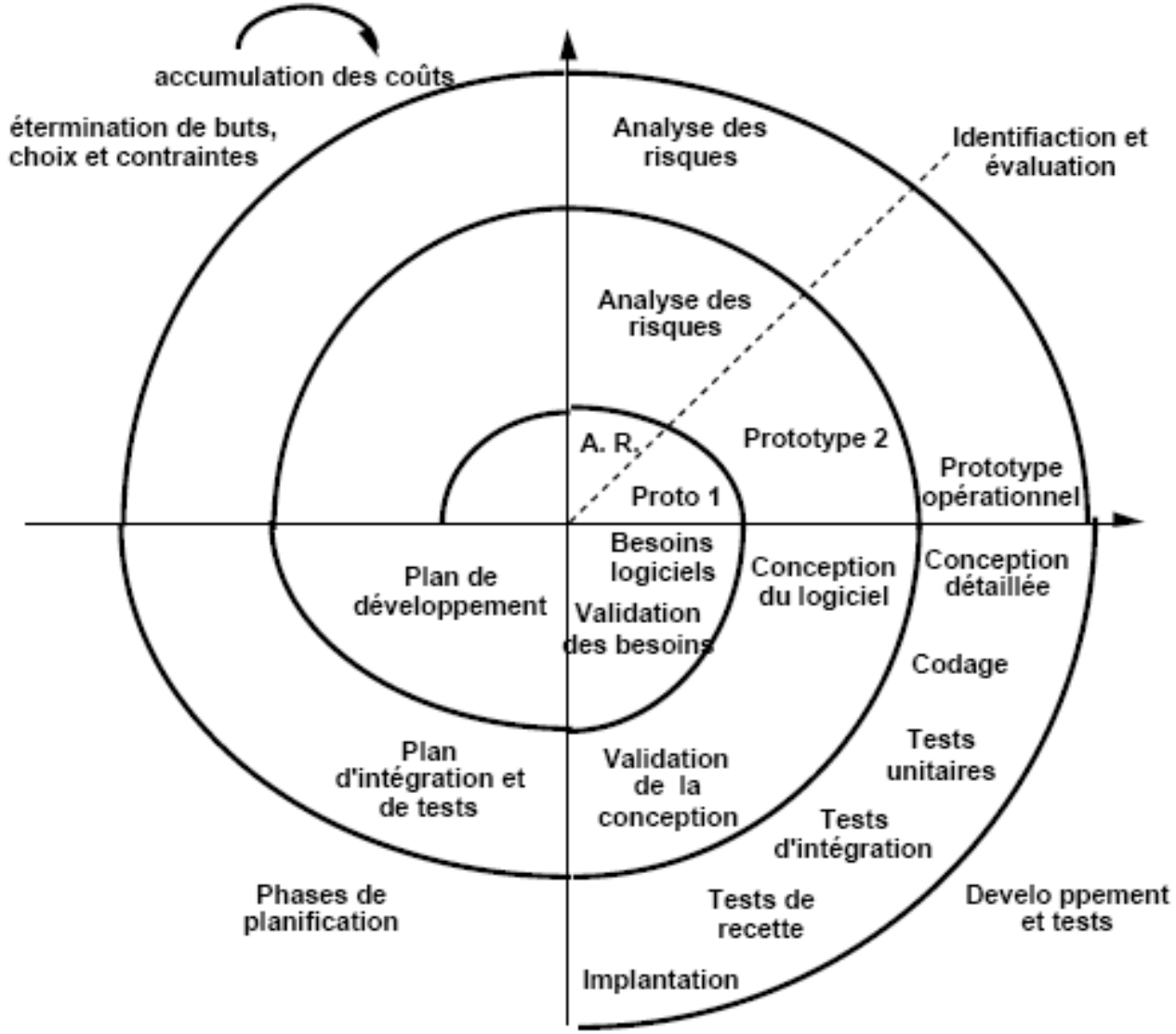


Cycle en spirale

- **On distingue 4 phases**

1. détermination
   - des objectifs,
   - des alternatives
   - des contraintes ;
2. analyse
   - des risques,
   - évaluation des alternatives ;
3. développement et vérification de la solution retenue ;
4. revue des résultats et vérification du cycle suivant.

d



Intervenants clés

Maître d’ouvrage

Maître d’œuvre

Génie Logiciel et humain

- Le Génie logiciel doit aussi gérer les aspects **organisationnels** dont
  - problèmes de gestion de ressources humaines et de synchronisation entre leurs tâches
    - Chaque intervenant à
      - un ou des rôles spécifiques
      - Des tâches
    - Nombreuses collaborations

La maîtrise d'ouvrage

- **Maître d'ouvrage (*Owner*) : MOA**
  - **Représente le propriétaire**
  - Expression de besoins
  - Expression des contraintes fortes (délais, coût)
  - Rédige le cahier des charges
- **Maître d'œuvre (*Engineer*) : MOE**
  - **Représente les développeurs**
  - Rédige les spécifications
  - Réalise l'ouvrage
- Ils sont liés par contrat
- Analogie au secteur du bâtiment (architecte et chef de chantier)

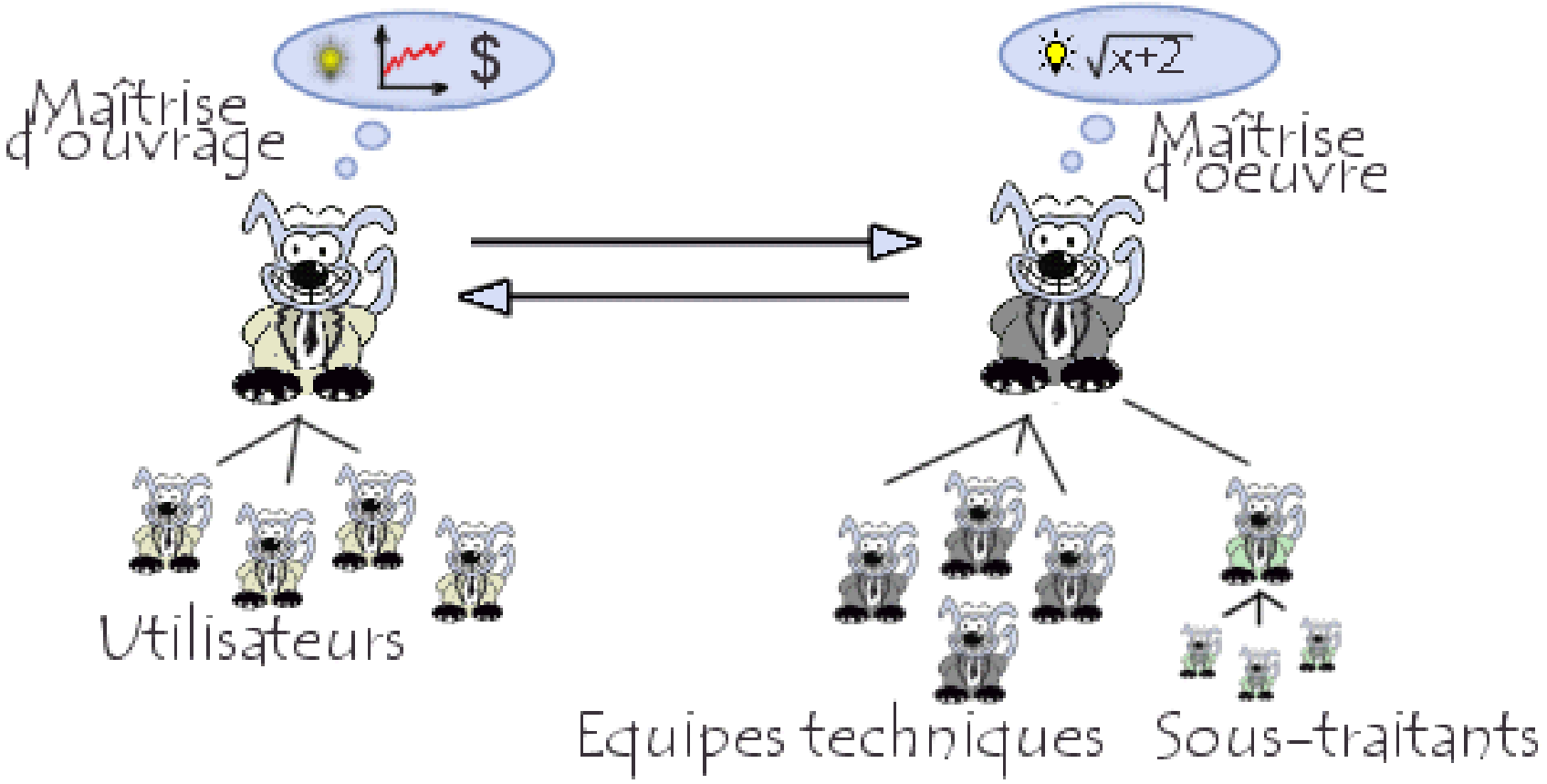
le maître d'ouvrage (MOA) pour un système d'information

- Il décrit les besoins, le cahier des charges,
- Il établit le financement et le planning général des projets.
- Il fournit au Maître d'œuvre les spécifications fonctionnelles (le « **modèle métier** ») et valide la recette fonctionnelle des produits.
- Il assure la responsabilité de pilotage du projet dans ses grandes lignes,
- Il adapte le périmètre fonctionnel en cas de retard dans les travaux, pour respecter la date de la livraison finale.

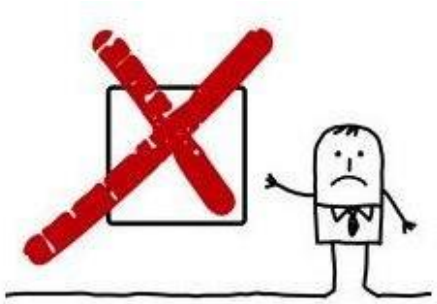
Le maître d'œuvre (MOE) pour un système d'information

- Il garantit la **bonne réalisation technique** des solutions.
- Il a un devoir de conseil vis-à-vis du maître d'ouvrage
- Assure la coordination des fournisseurs (prestations, sous-traitance).
- il veille à la cohérence des fournitures et à leur compatibilité.
- Il coordonne l'action des fournisseurs en contrôlant la qualité technique, en assurant le respect des délais fixés par le MOA et en minimisant les risques.
- Il est responsable de la qualité technique de la solution.
- Il doit, avant toute réception contractuelle, procéder aux vérifications nécessaires selon une méthodologie définie préalablement : recette « usine », pré-réception sur site, validation en service régulier...

La maîtrise d'ouvrage

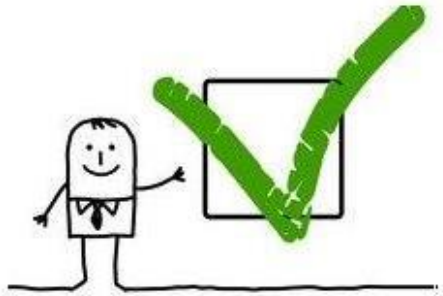


vision du chef



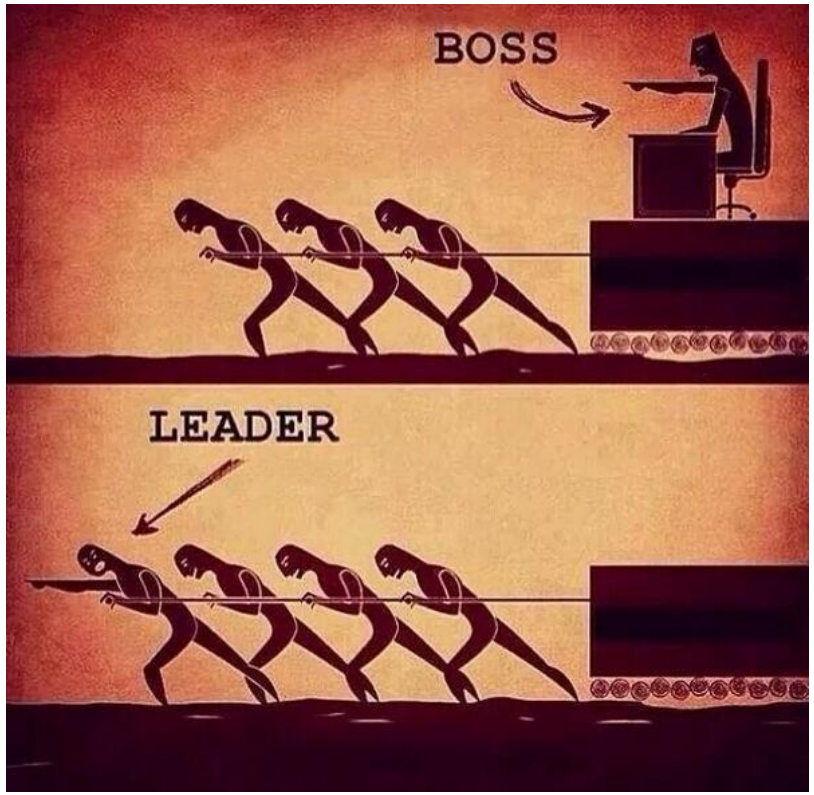
Petit Chef

Il dirige les employés
Il impose son autorité
Il inspire la peur
Il dit "Je"
Il appuie sur les erreurs
Il sait ce qui est fait
Il utilise les gens
Il récolte les lauriers
Il commande
Il dit "Allez-y !"



Manager

... Il les accompagne
... avec intelligence
... Il génère l'enthousiasme
... Il dit "Nous"
... Il assume et corrige les erreurs
... Il montre ce qui doit être fait
... Il fait grandir les gens
... Il distribue les récompenses
... Il demande
... Il dit "Allons-y"



Maintenance

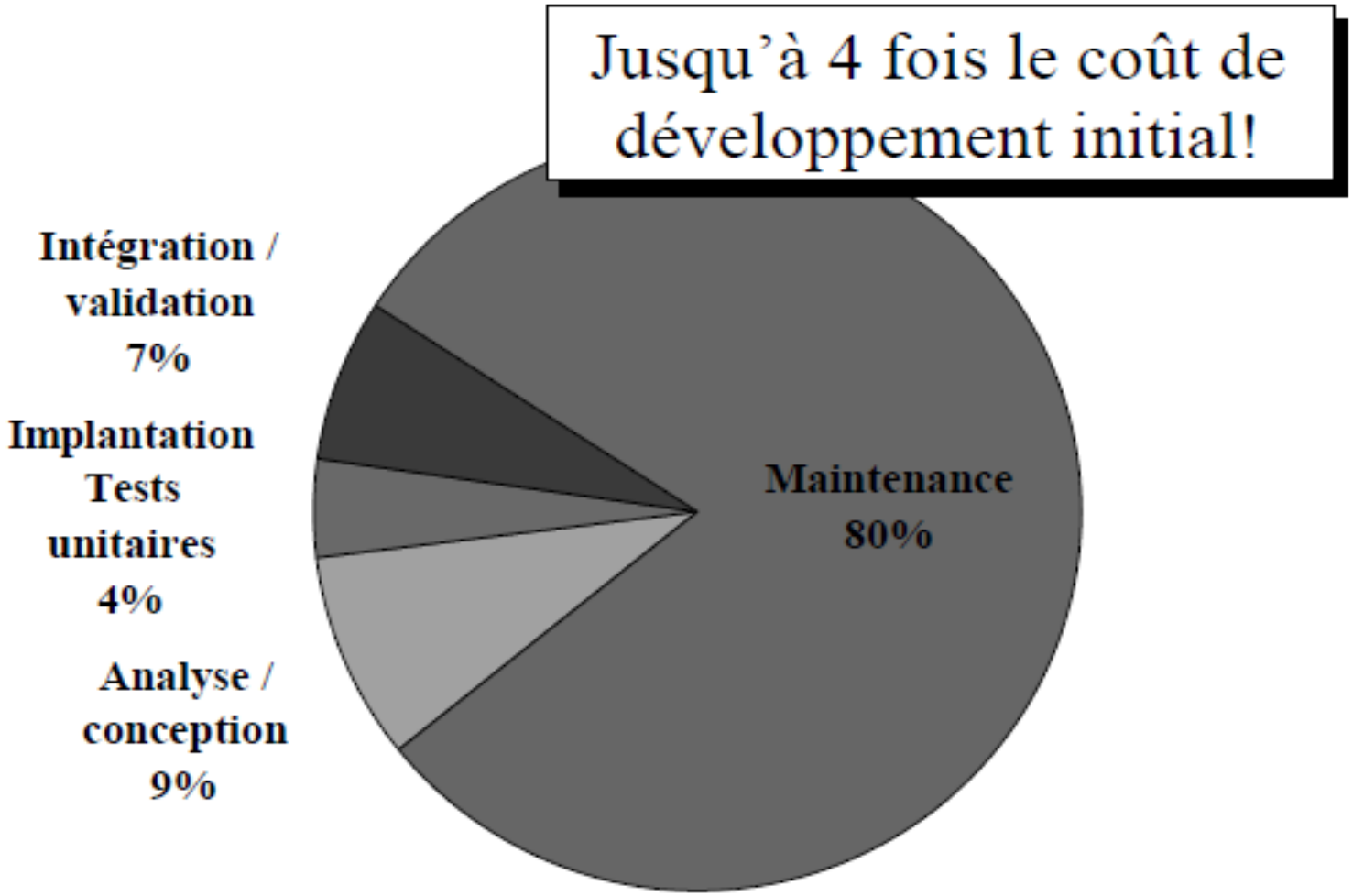
Maintenance

- Activité qui comprend la **formation** de l'utilisateur et l'assistance technique.
- La maintenance peut être :
  - **Corrective**: non conformité aux spécifications, d'où détection et correction des erreurs résiduelles.
  - **Adaptative**: modification de l'environnement (matériel, fournitures logicielles, outil, ...)
  - **Évolutive**: changement des spécifications fonctionnelles du logiciel.
- Les activités de maintenance couvrent les domaines suivants:
  - qualifications des nouvelles versions
  - suivies des modifications
  - archivage
  - mise à jour de la documentation
  - Livraison des nouvelles versions
  - exécution des modifications

Maintenance

- Gestion de l'évolution d'un système
- D'après Swanson & Beath (Maintaining Information Systems in Organizations, 1989)
  - Durée de vie moyenne d'un système : moins de 7 ans
    - Maintenant sûrement moins
  - 26% des systèmes sont âgés de plus de 10 ans
- Problème de la maintenance corrective et évolutive
  - Adaptation aux changements des besoins
  - Adaptation aux changements de l'environnement
    - Support nouvelles plates-formes,

Maintenance



Maintenance et Rétro-ingénierie

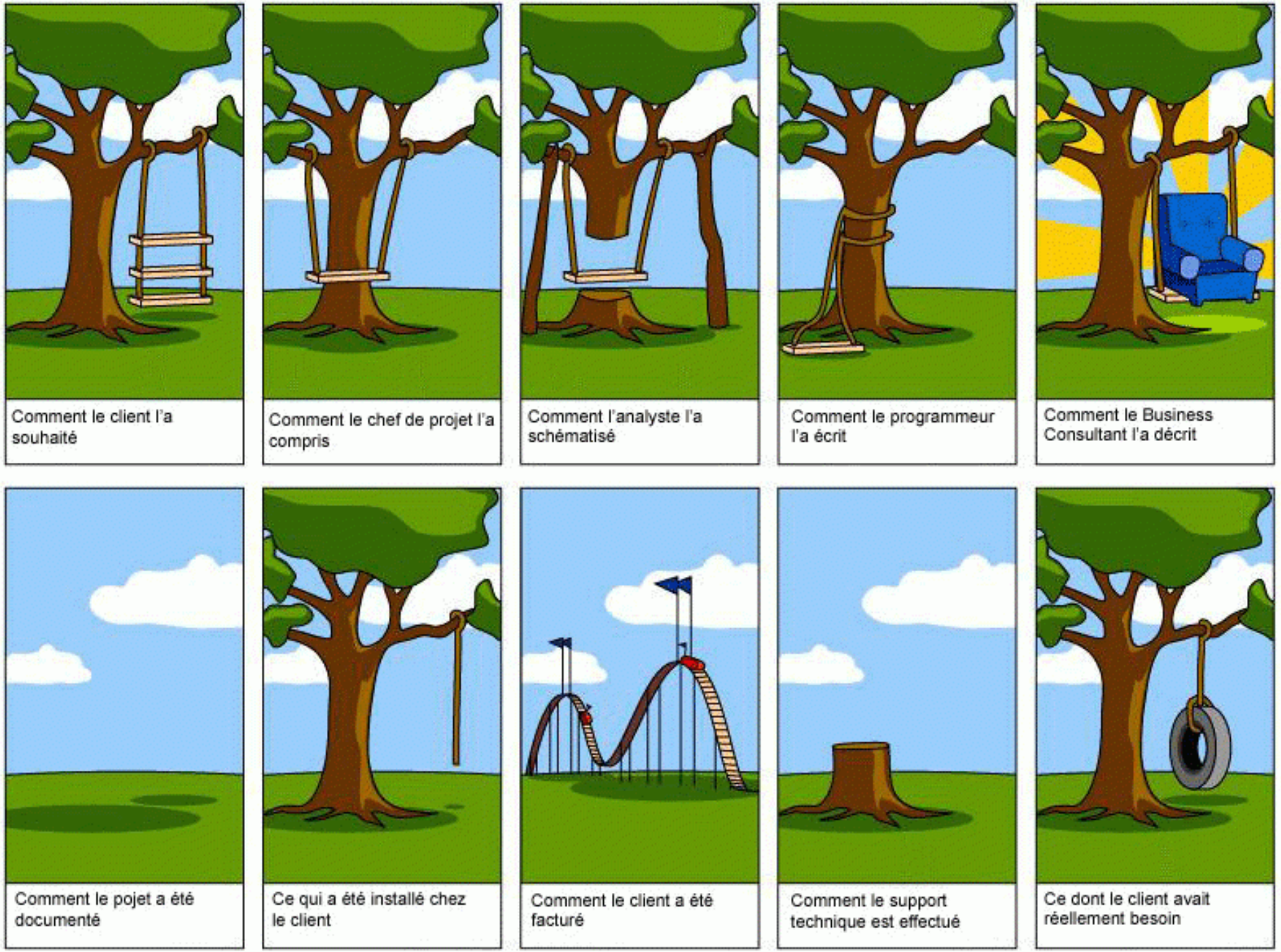
- **Coût moyen des modifications augmente avec le temps :**
  - Saturation de l'architecture et des interfaces
  - Documentation technique se désagrège
  - Tests ne sont plus pertinents
  - Équipes de développement et de maintenance changées plusieurs fois.
  - Le flux d'erreurs oscille (injection de nouvelles erreurs)
- **Solutions possibles :**
  - On recommence tout (80's)
  - Récupération partielle (Rétro-ingénierie) puis modernisation (90's)

Maintien en condition

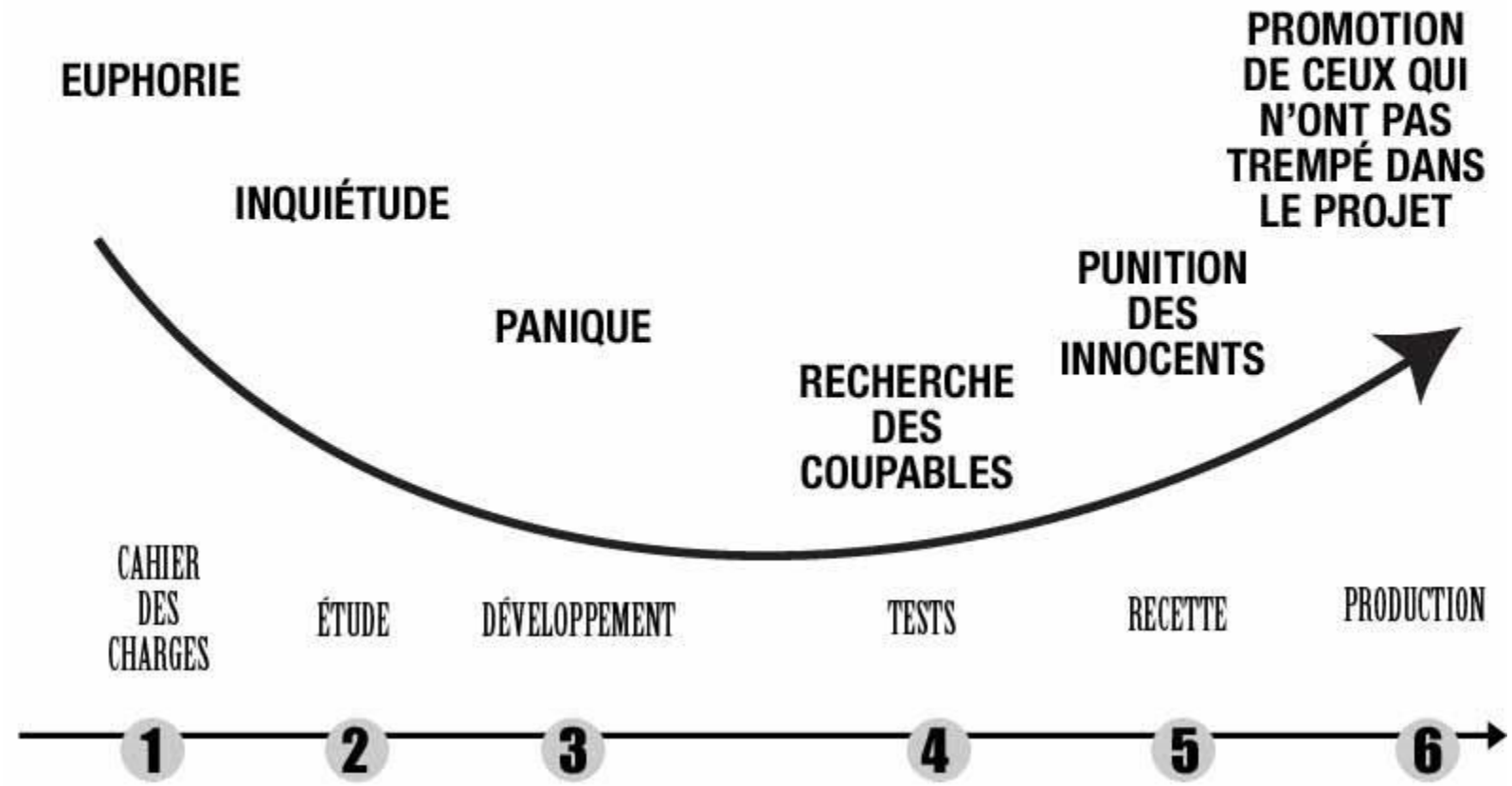
- **Beaucoup de projets informatiques**
  - manquent de suivi
  - Sont livrés sans une vraie réflexion sur son utilisation, sa maintenance
  - Or, un système d'information est un service critique pour une entreprise
  - Besoin d'aborder les sujets suivants :
    - Comment organiser un système d'information ?
    - Comment améliorer l'efficacité du système d'information ?
    - Comment réduire les risques ?
    - Comment augmenter la qualité des services informatiques ?

Conclusion

humour


Comment le client l'a souhaité
Comment le chef de projet l'a compris
Comment l'analyste l'a schématisé
Comment le programmeur l'a écrit
Comment le Business Consultant l'a décrit
Comment le pojet a été documenté
Ce qui a été installé chez le client
Comment le client a été facturé
Comment le support technique est effectué
Ce dont le client avait réellement besoin

CONDUITE DE PROJET



Conclusion

- **Les projets informatiques**
  - Echouent rarement pour des raisons techniques
  - Mais pour des raisons
    - De mauvaise conception
    - De défaut d'organisation
    - D'interface entre applications
    - De tests insuffisants
- **Facteurs humains très importants**
  - Besoin de compétences variées
    - pour des équipes différentes
    - À l'intérieur de la même équipe