Cours Gestion de projet

Méthodes de conduite de projet

Version	V2.2
Date	Janvier 2022
Auteur	Jean-Pierre ROBERTI

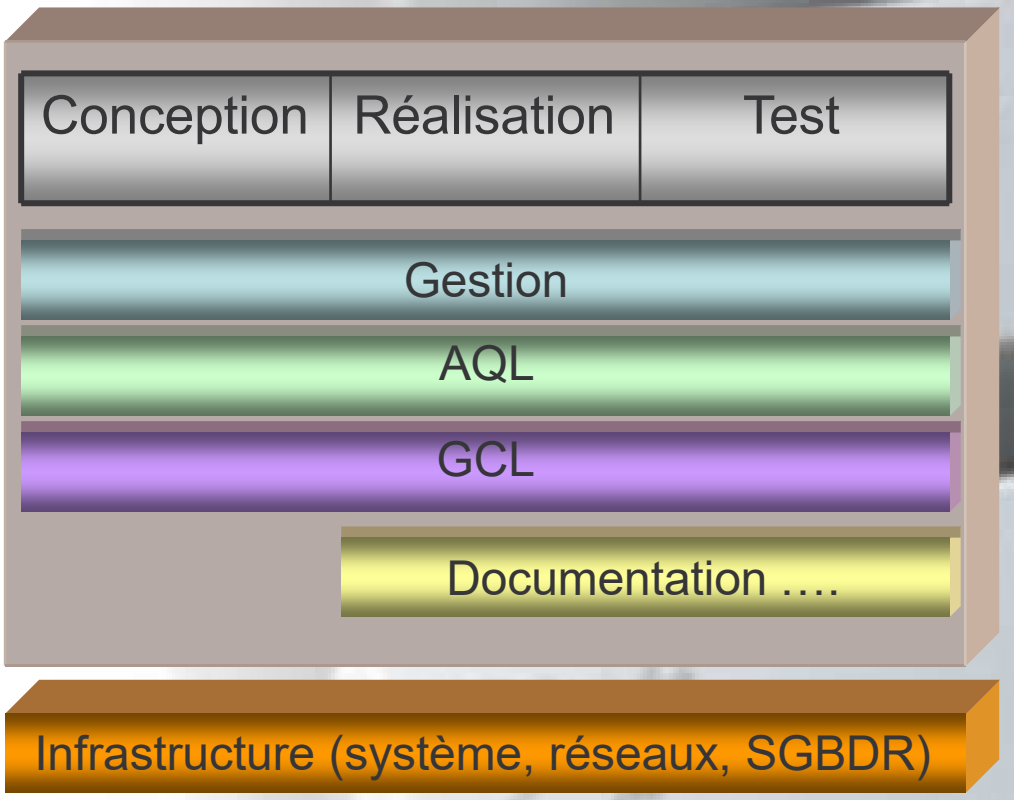
Méthodes de conduite de projet

Méthodes de conduite de projet

- Sommaire
  - Rappel sur les activités de développement
  - Plan de conduite
  - Cycle de vie du logiciel
  - Les principales méthodes
  - Synthèse
  - Recommandations

Rappel sur les activités d'un projet informatique de développement

- Activités d'ingénierie logiciel :
  - Conception
  - Développement
  - Test
- Activités de gestion :
  - Gestion de projet
  - Gestion des tests
- Activités support :
  - Assurance de la Qualité
  - Gestion de configuration logiciel
  - Documentation, traduction, formation
  - Système, réseaux, base de données



Le plan de conduite = scénario du projet

- Le scénario du projet dépend :
  - **De la méthode de conduite de projet** utilisée qui définit les activités d'ingénierie logicielle :
    - Découpage Projet en Phases/étapes/tâches
      - Pour chaque étape : entrée, activités (acteurs/responsabilité), sorties (produits), V&V
    - Le cycle de vie
  - **De la Méthode de conception (modélisation)**

Précisions sur le vocabulaire

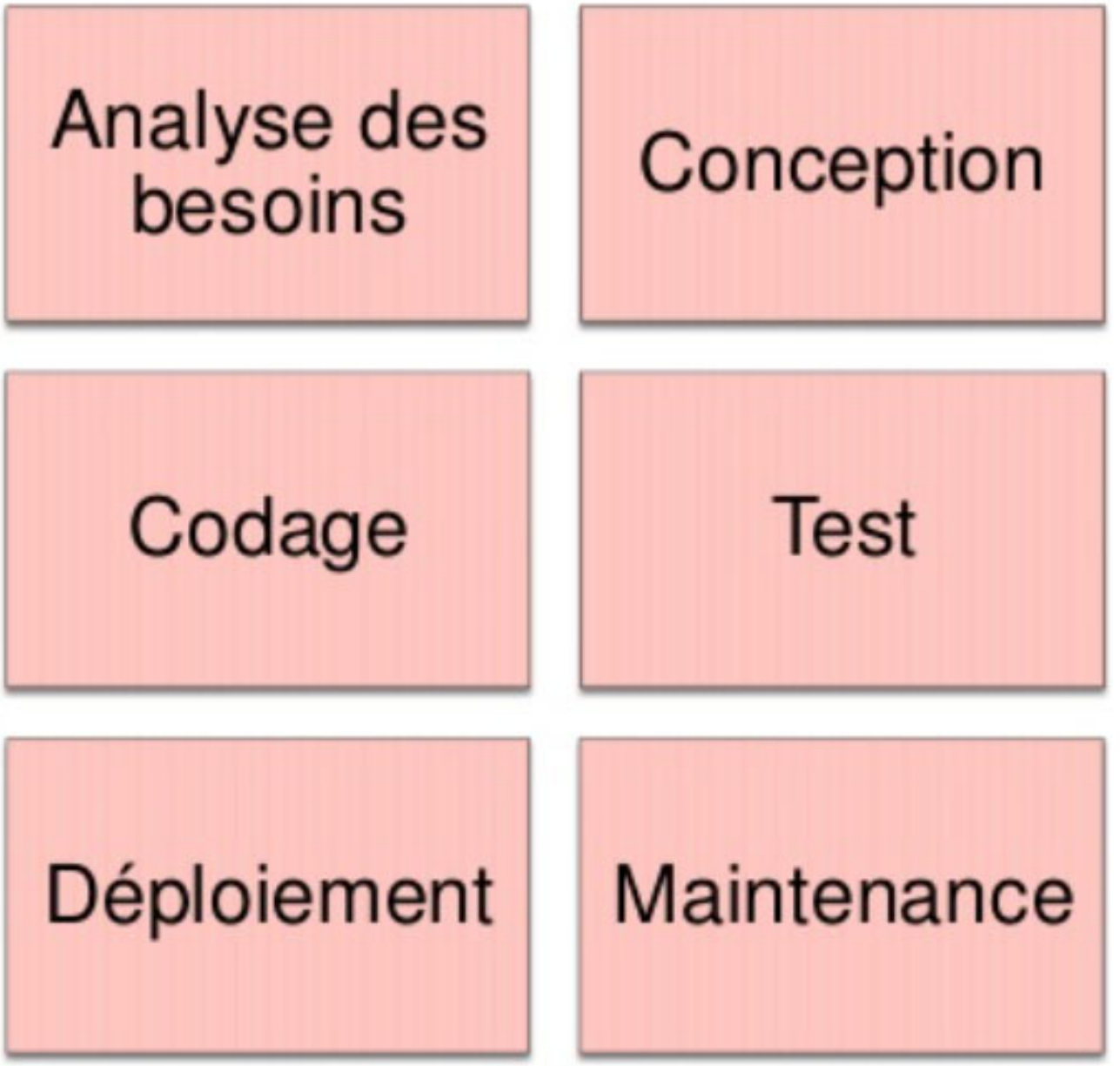
- Les méthodes de conception (modélisation) permettent de décrire un système d'information et ses évolutions :
  - UML
- Les méthodes de conduite de projet décrivent un processus de développement qui s'appuie sur une méthode de modélisation :
  - UP
  - RAD
- Des adhérences entre conception et conduite de projet :
  - Merise/Merise, UML/(UP et ses déclinaisons)

Les objectifs du cours

- Il s'agit de :
  - Comprendre les grands concepts des principales méthodologies de conduite de projet et leurs cas d'utilisation
  - De les appliquer à bon escient

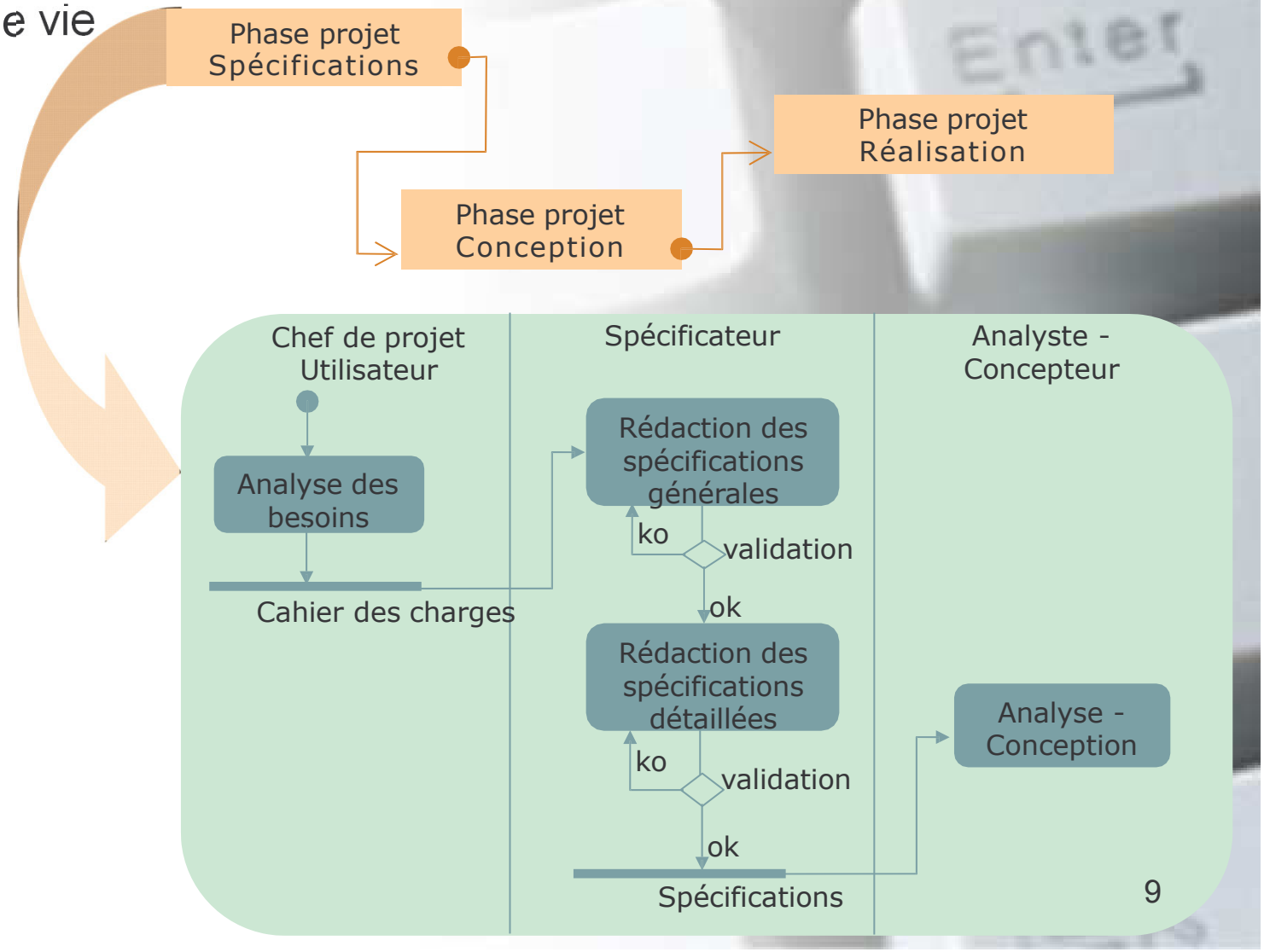
- Il ne s'agit pas de :
  - De maîtriser ces méthodologies

Les étapes principales de création d'un logiciel



Le processus de développement

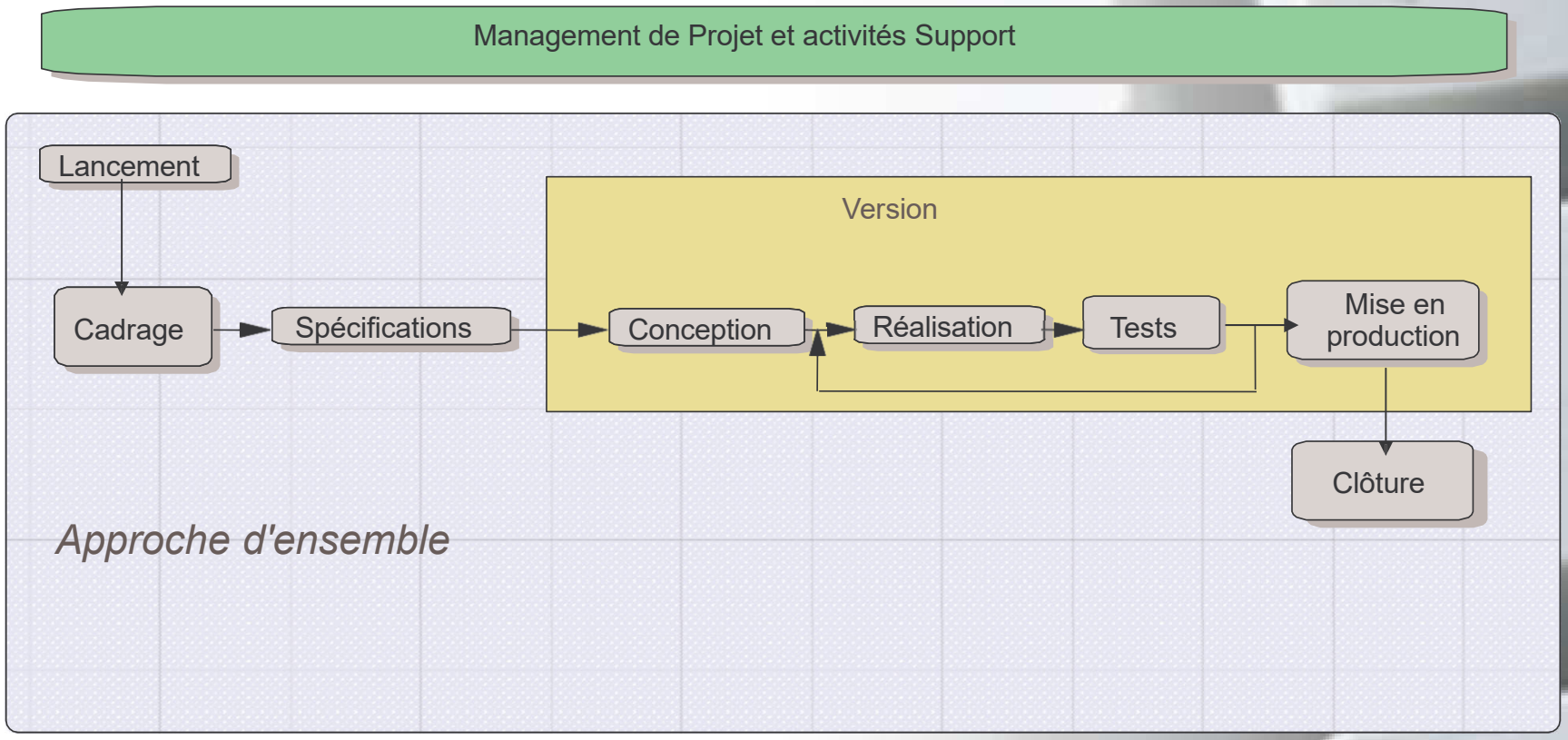
- Phases, étapes, activités de spécification, de conception, réalisation et test avec des activités de V&V
  - Le cycle de vie



Le processus de développement

- Est constitué de Phases/étapes/activités de spécification, de conception, réalisation et test avec des activités de V&V

  Leur enchaînement = Le cycle de vie



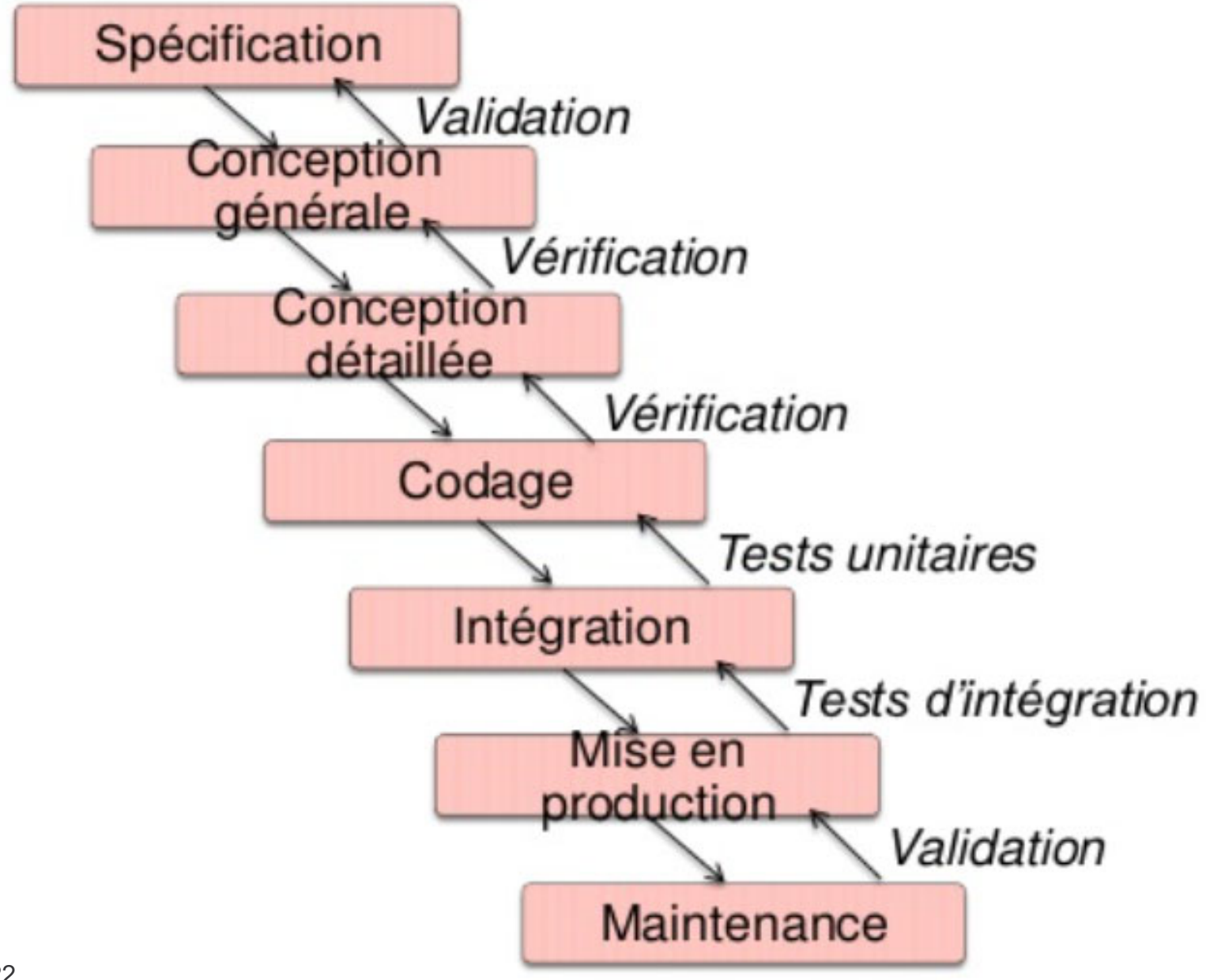
Cycle de vie d'un logiciel

- Il existe deux types de méthodes :

Méthodes classiques	Méthodologies Agiles

- Modèle en cascade
- Modèle en V
- Modèle incrémental
- Modèle en spirale

Cycles de vie classiques, modèles en CASCADE



Cycles de vie classiques, modèles en CASCADE

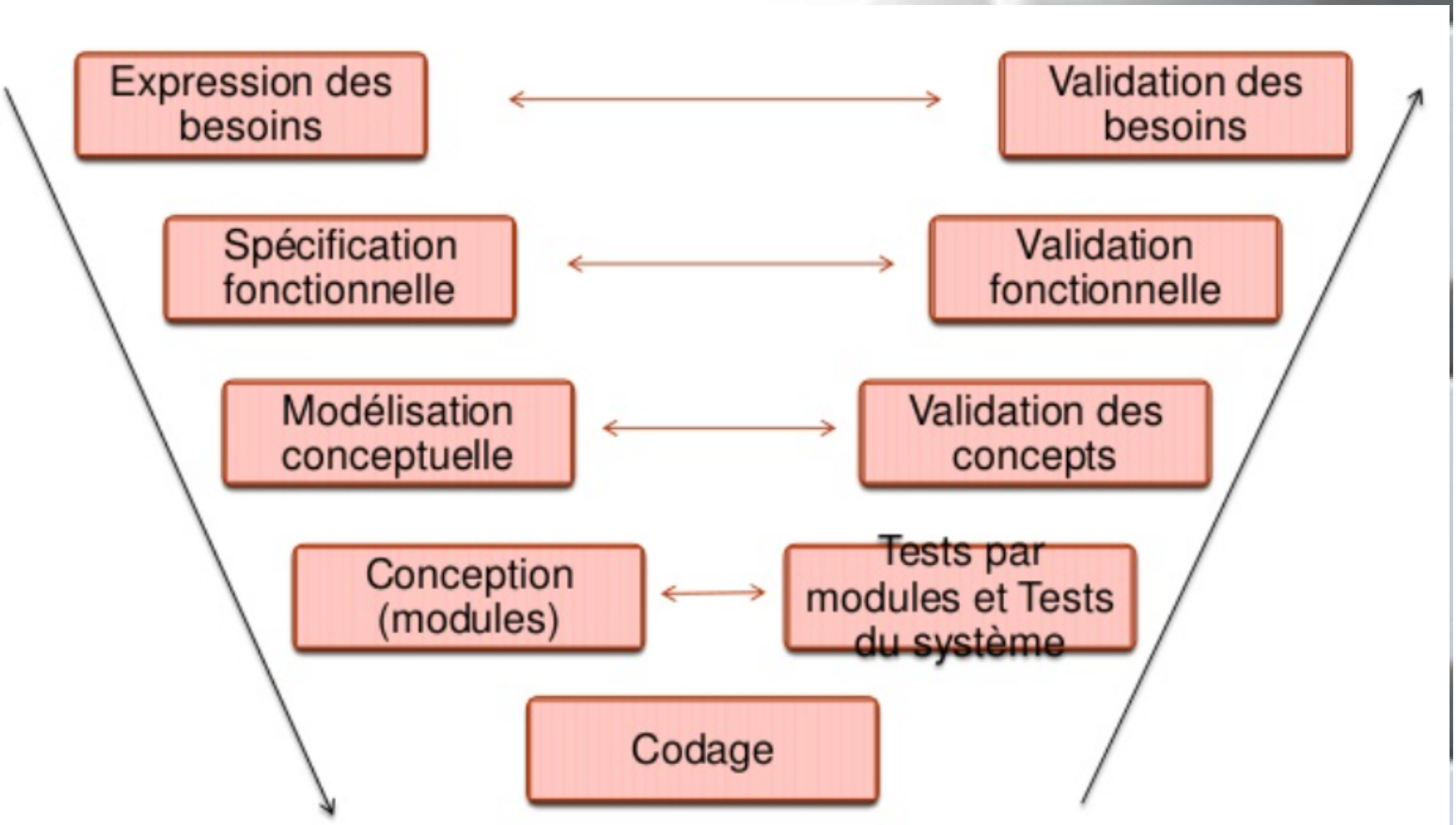
Avantages

- Facile à utiliser et à comprendre
- Structure simple pour une équipe inexpérimentée
- Fonctionne bien quand la qualité est beaucoup plus importante que les couts et le temps

Inconvénients

- Sensibilité aux nouveaux besoins : refaire tout le procédé
- Une phase ne peut démarrer que si l'étape précédente est finie
- Le produit n'est visible qu'à la fin
- Les risques se décalent vers la fin
- Très faible implication du client

Cycles de vie classiques, modèles en V



Cycles de vie classiques, modèles en V

Avantages

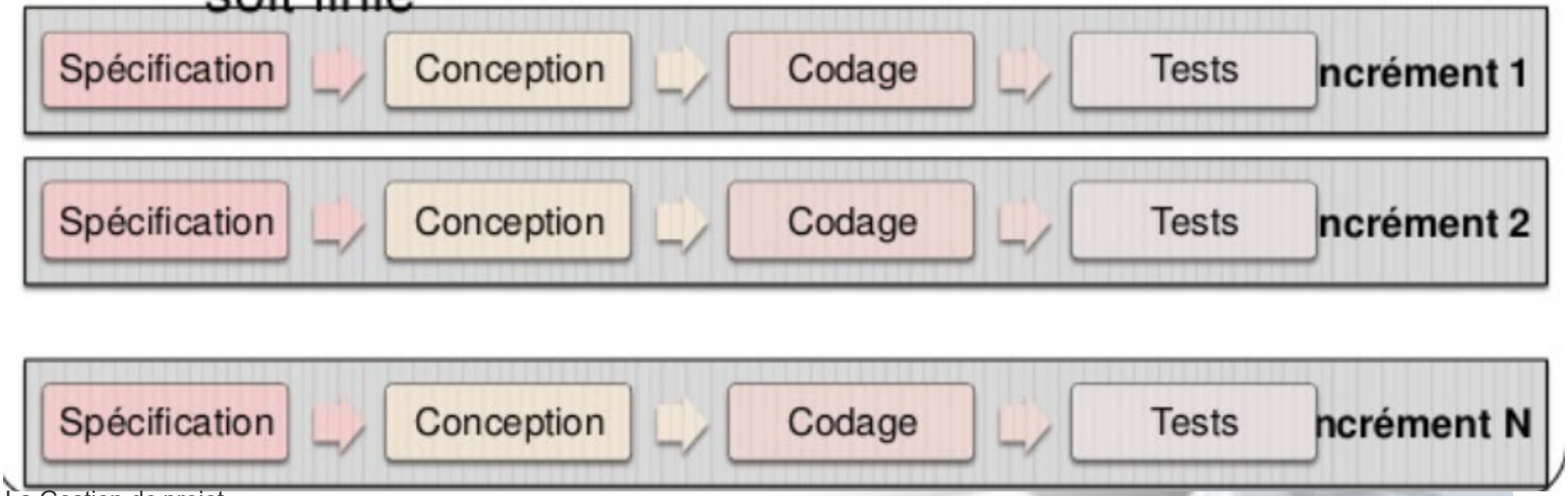
- Met l'accent sur les tests et la validation et donc accroît la qualité
- Chaque livrable doit être testable
- Facile à utiliser et planifier

Inconvénients

- Ne gère pas les activités parallèles
- Ne gère pas les changements des spécifications
- Ne contient pas d'activités d'analyse de risque

Cycles de vie classiques, modèle INCREMENTAL

- Chaque incrément est une construction partiel le du logiciel
- Trie les spécifications par priorités
- Chaque incrément implémente un ou plusieurs spécifications jusqu'à ce que la totalité du produit soit finie



Cycles de vie classiques, modèle INCREMENTAL

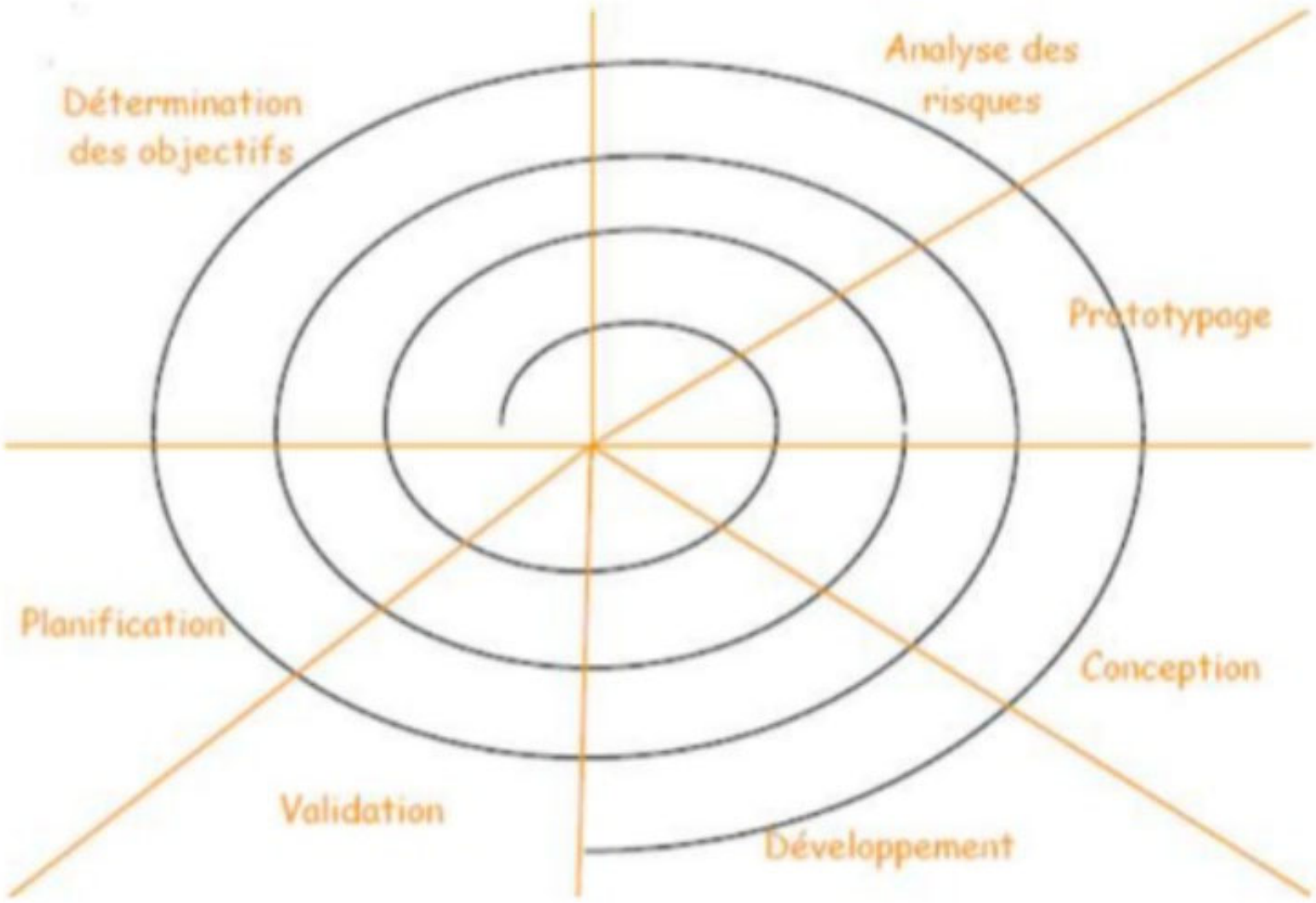
Avantages

- Développement de fonctionnalités à risque en premier
- Chaque incrément donne un produit fonctionnel
- Le client intervient à la fin de chaque incrément
- Utiliser l'approche « diviser pour régner »
- Le client entre en relation avec le produit très tôt

Inconvénients

- Exige une bonne planification et une bonne conception
- Exige une vision sur le produit fini pour pouvoir bien le diviser en incréments
- Le coût total du système peut être cher

Cycles de vie classiques, modèle en SPIRALE



Cycles de vie classiques, modèle en SPIRALE

Avantages

- Inclut l'analyse de risque et le prototypage
- Fonctions critiques développées en premier
- Feedback rapide du cl ient
- Une évaluation continue du procédé
- Chaque cycle est composé des même activités que du modèle en cascade

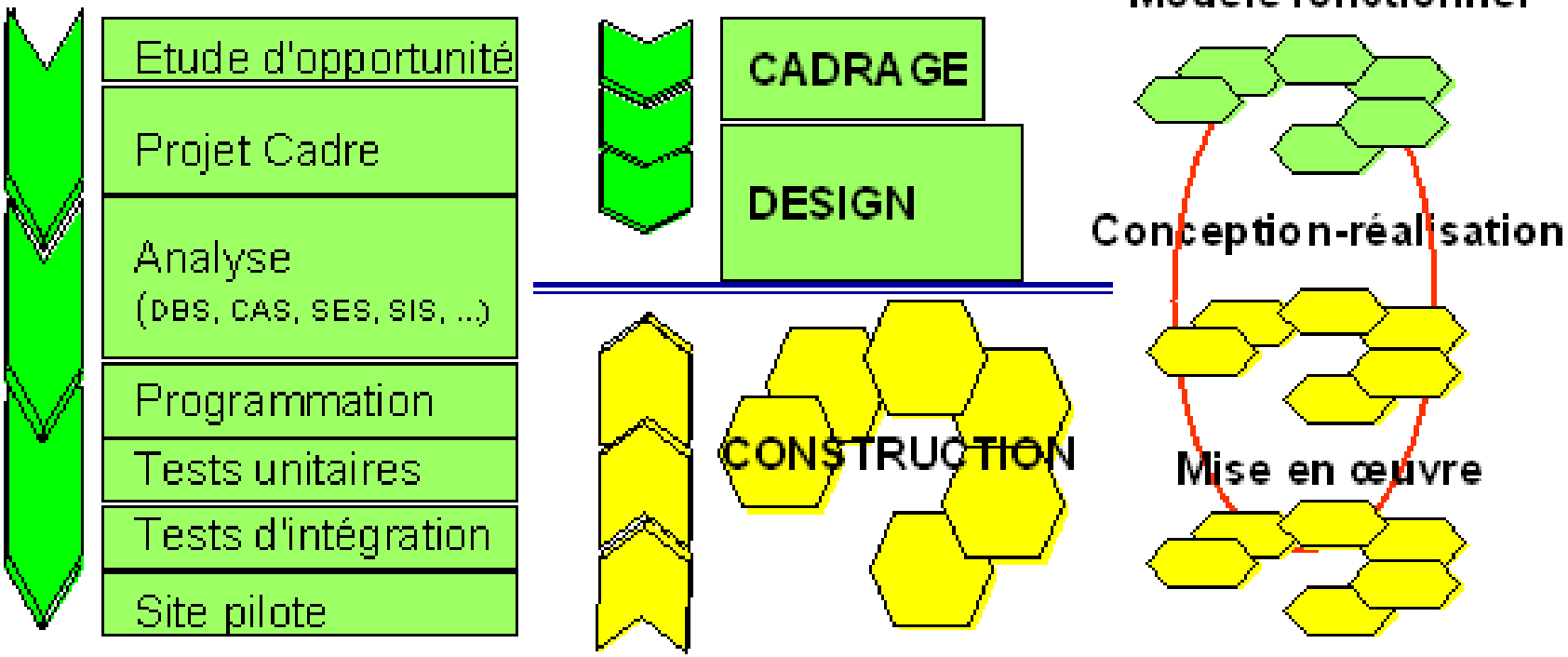
Inconvénients

- L'évaluation des risques peut prendre beaucoup de temps
- Le modèle est très complexe

Cycle de vie et méthodes de conduite

Cascade (Merise, SDM/S) **Semi-itératif (RAD)** **Itératif (UP, XP)**

Par la structure : cohérence systémique



Par le besoin : adéquation fonctionnelle

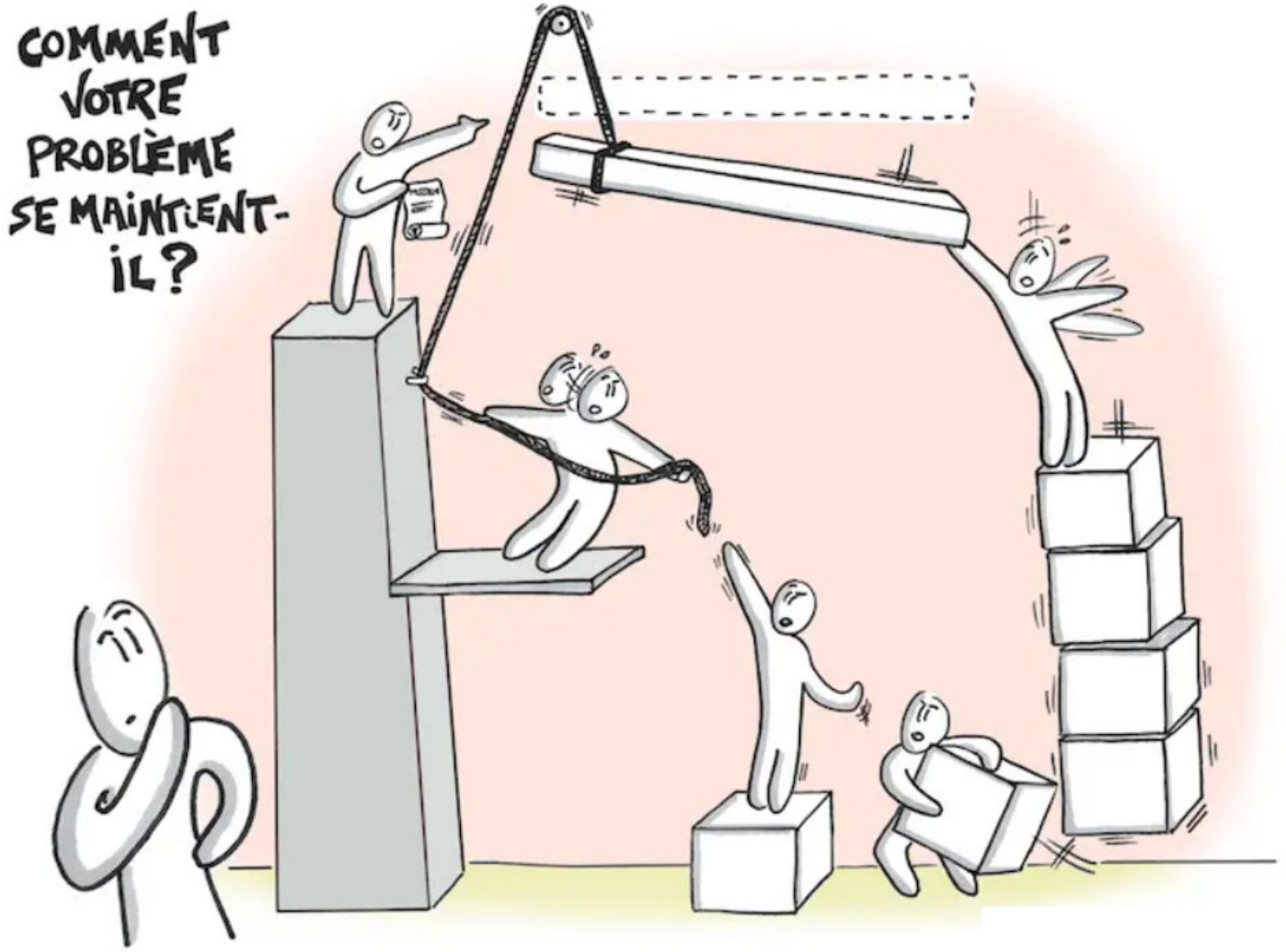
La méthode systémique

-) La **méthode systémique** est axée sur un ensemble des organes cohérents s'influençant les uns les autres, dépendant les uns des autres, et agissant les uns sur les autres

-) Pour appliquer l'approche systémique il faut **penser circulairement et non linéairement**.

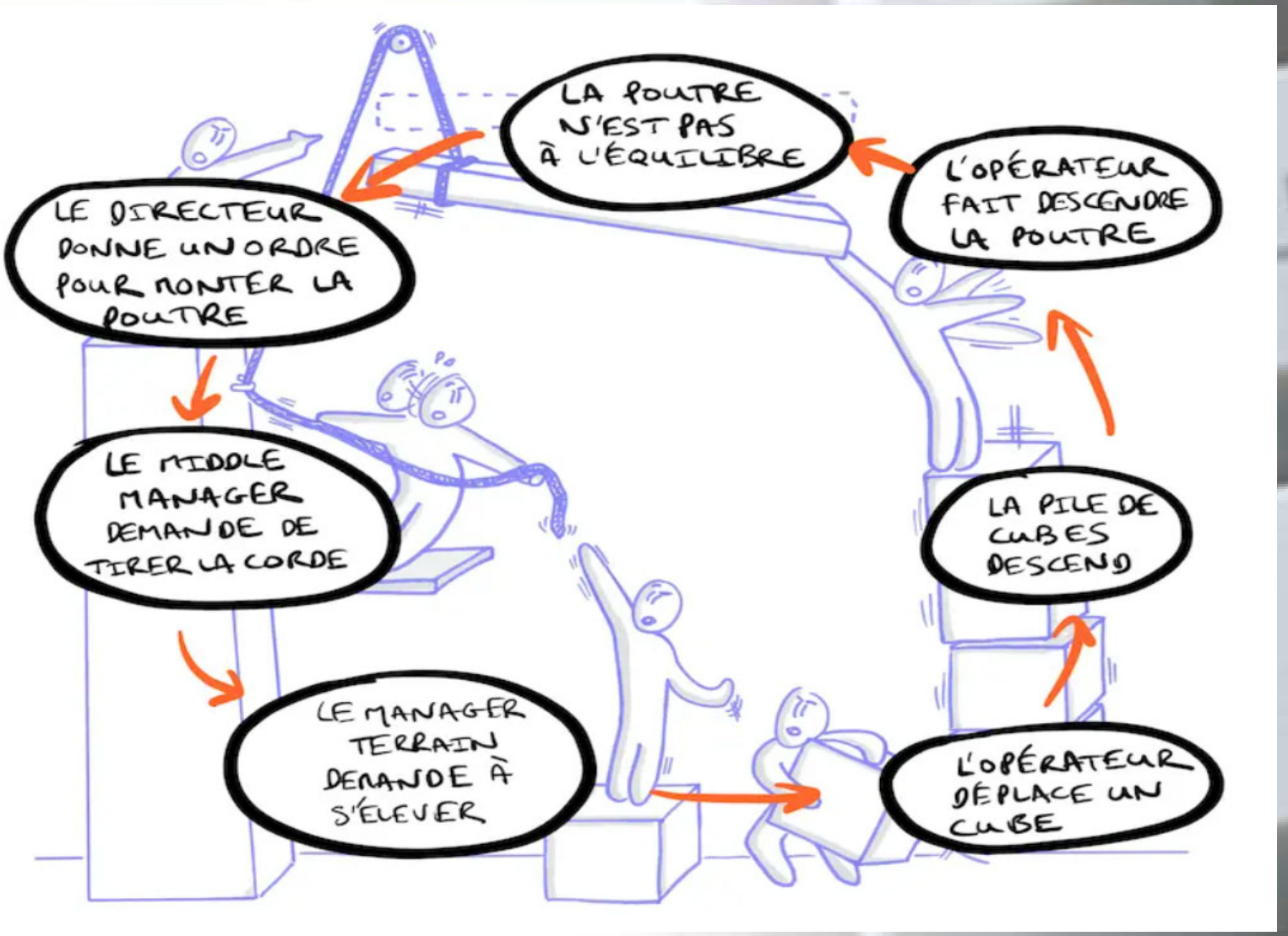
C'est-à-dire qu'il faut réaliser que tout est interconnecté.
(Imaginons que nous arrosions régulièrement une plante et que malgré toute notre bonne volonté celle-ci flétrisse.)

L’approche systémique



Et si causes et solutions de nos problèmes se confondaient ?

L’approche systémique



-) Chaque élément se retrouve à la fois cause et effet

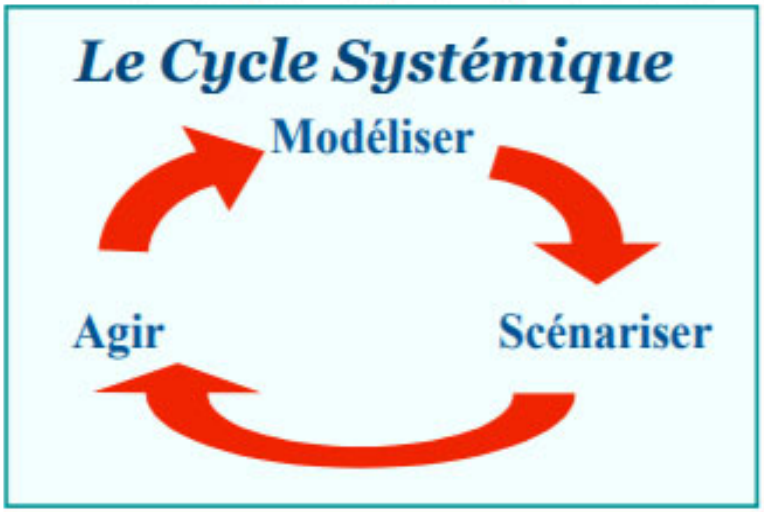
-) Prendre du recul pour apprécier la situation dans son ensemble. Ne pas chercher LA cause originelle….

Approche systémique

C’est l'étude d'objets dans leur complexité.
Pour tenter d'appréhender cet objet d'étude **dans son environnement, dans son fonctionnement, dans ses mécanismes, dans ce qui n'apparait pas en faisant la somme de ses parties**, cette démarche vise par exemple à identifier :

-) la « finalité » du système ,
-) les niveaux d'organisation,
-) les états stables possibles,
-) les échanges entre les parties,
-) les facteurs d'équilibre et de déséquilibre
-) les boucles logiques et leur dynamique, etc.

Approche systémique

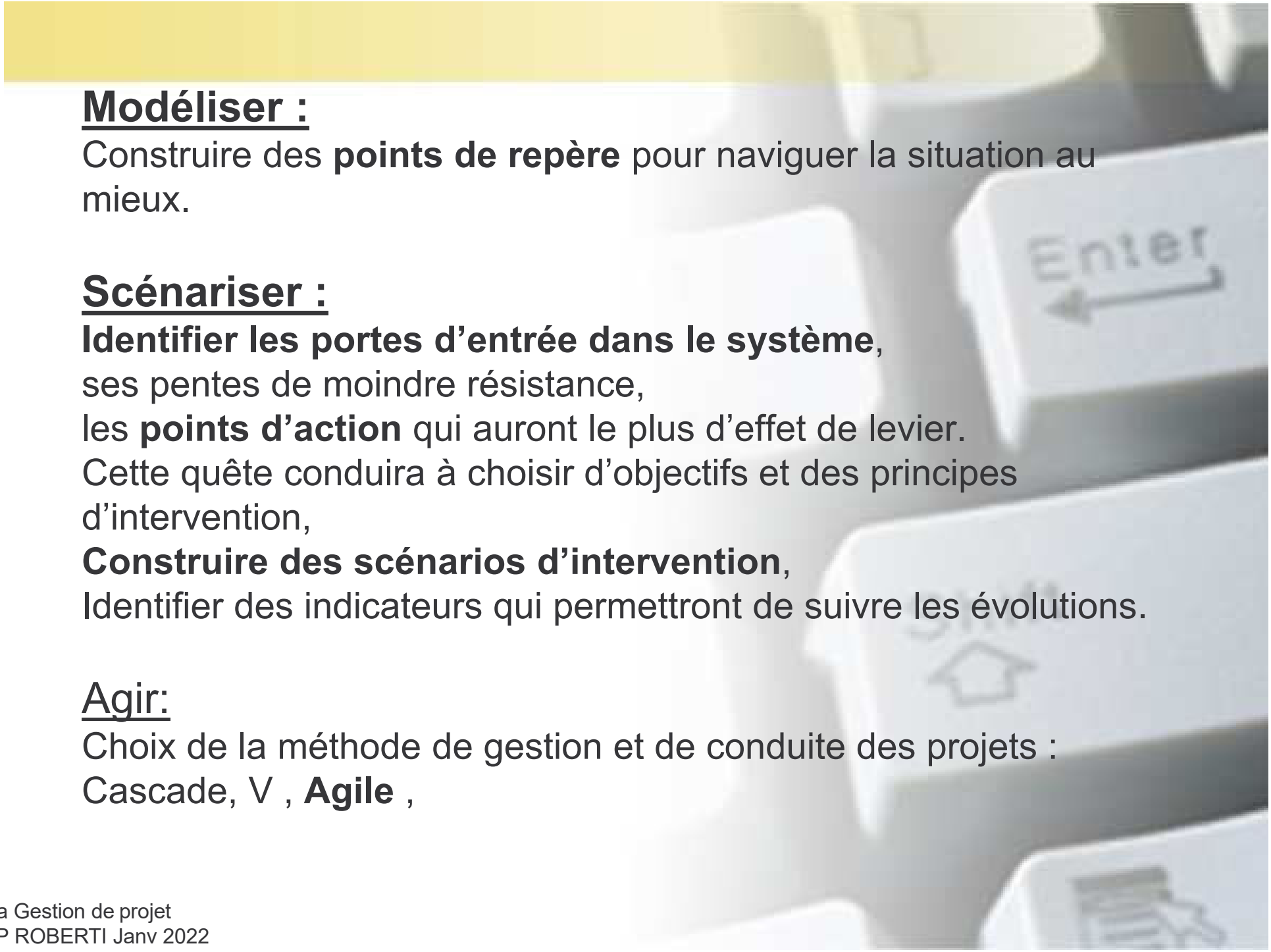


L'approche systémique s'applique en premier lieu aux situations confuses
Qui renvoient à de multiples perceptions contradictoires ;
Qui résistent à l'action et qu'on désespère de faire évoluer.

En conséquence, on préférera les méthodologies de la qualité pour les problèmes clairs et bien identifiés. Et l'approche systémique pour les situations floues, confuses et ambiguës.

Situations que les organisations doivent d'abord apprivoiser avant de réussir à trouver les bons modes d'action.

L'approche systémique s'appuie sur son propre cycle : **Model–Scenarise–Act**, c'est-à-dire **modéliser, scénariser, agir.**



Modéliser :

Construire des **points de repère** pour naviguer la situation au mieux.

Scénariser :

Identifier les portes d'entrée dans le système,
ses pentes de moindre résistance,
les **points d'action** qui auront le plus d'effet de levier.
Cette quête conduira à choisir d'objectifs et des principes d'intervention,
Construire des scénarios d'intervention,
Identifier des indicateurs qui permettront de suivre les évolutions.

Agir:

Choix de la méthode de gestion et de conduite des projets :
Cascade, V , **Agile** ,

Approche fonctionnelle

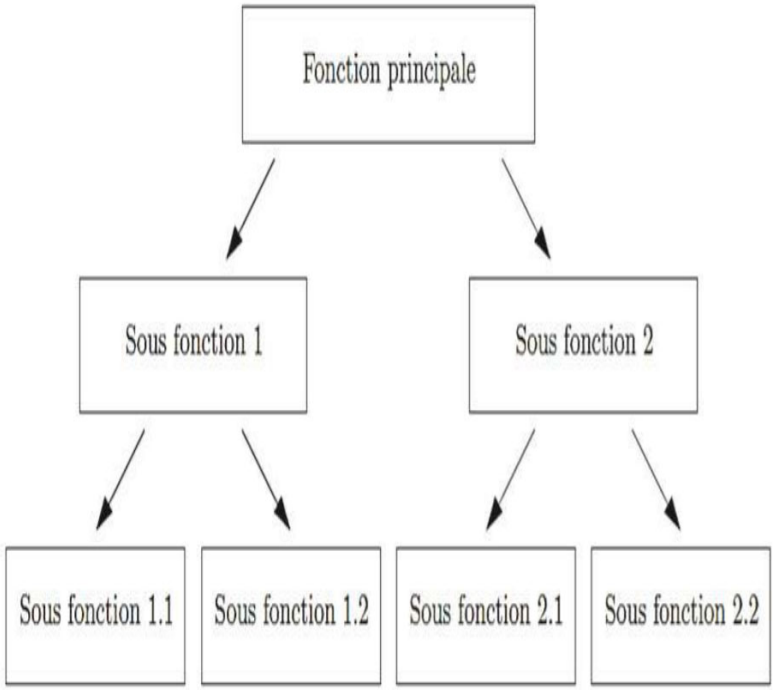


Schéma de la décomposition fonctionnelle

Les méthodes fonctionnelles ont pour origine la **programmation structurée**.
Elles consistent à **décomposer une fonctionnalité (ou fonction) du logiciel en plusieurs sous fonctions plus simples**.
Il s'agit d'une conception « **top-down** », basée sur le principe « **diviser pour mieux régner**».

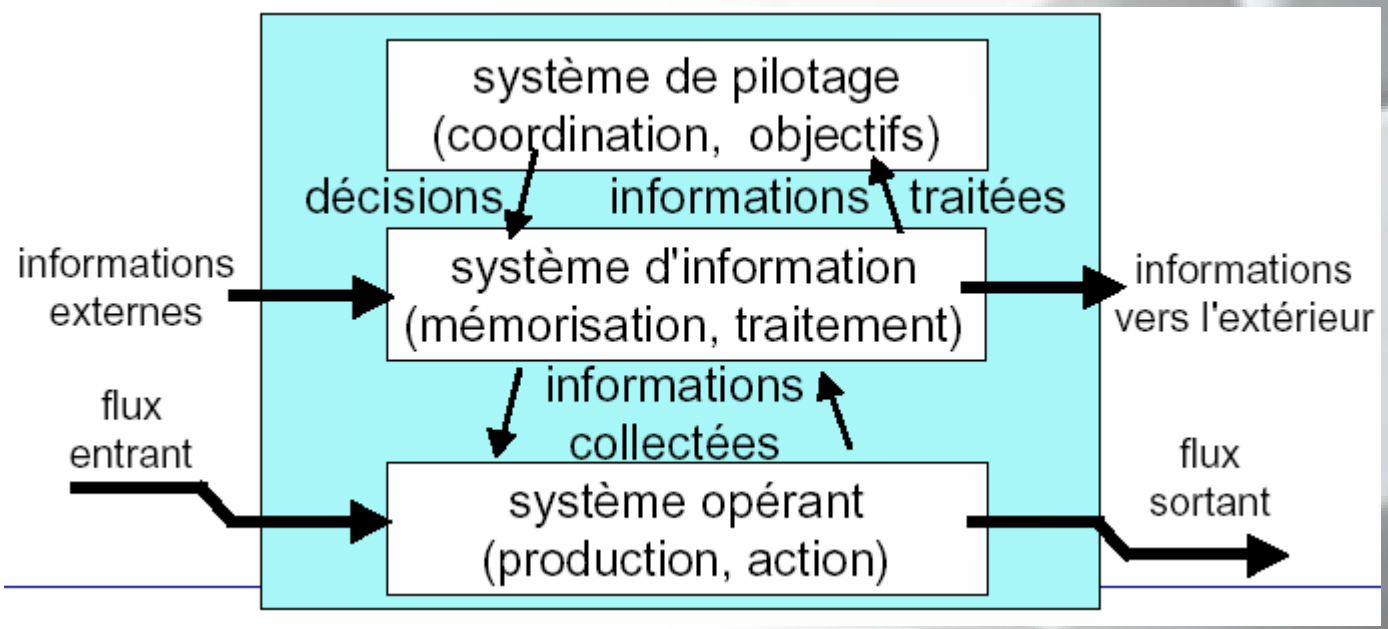
L'architecture du système est le reflet de cette décomposition fonctionnelle.
La programmation peut ensuite être réalisée soit à partir des fonctions de haut niveau (développement « top-down »), soit à partir des fonctions de bas niveau (développement « bottom-up »).

Merise/Historique

- **M**éthode d'**É**tude et de **R**éalisation **I**nformatique pour les **S**ystèmes d'**E**ntreprise
- En 1977/1978, demande du Ministère de l'Industrie : choix de sociétés de conseil en informatique pour la constitution d'une méthode de conception des systèmes d'information
  - Équipe de J.-L. Lemoigne (Univ. d'Aix / Marseille)
  - CTI (Centre Technique d'Informatique)
  - CETE (Centre d'Études Techniques de l'Équipement)

Merise/Principes de base

- Principes de base de la méthode Merise
  - Vision globale : la mise en place d'un S.I. est liée à la refonte de l'organisation
  - Vision systémique de l'entreprise : travaux de J.-L. Lemoigne, J. Melese, J. de Rosnay



Merise/Approche

1. Approche par niveaux et par étapes
   - Niveaux :
     - en contribuant à la stratégie de l'entreprise en mettant en œuvre les règles de gestion
     - en tenant compte des aspects organisationnels
     - en tenant compte des aspect techniques

     => formalise le système futur
   - Etapes :
     - conception
     - développement
     - mise en œuvre
     - généralisation de l'emploi du S.I. futur
     - évolution du S.I. futur

     => hiérarchise les décisions au cours de la vie du projet
2. Séparation des données et des traitements

Merise/Principes de base

1. Passage par différents étapes : "le cycle d'abstraction pour la conception des S.I."

   >**système d'information manuel**

   >expression des besoins

   >modèle conceptuel (quoi)

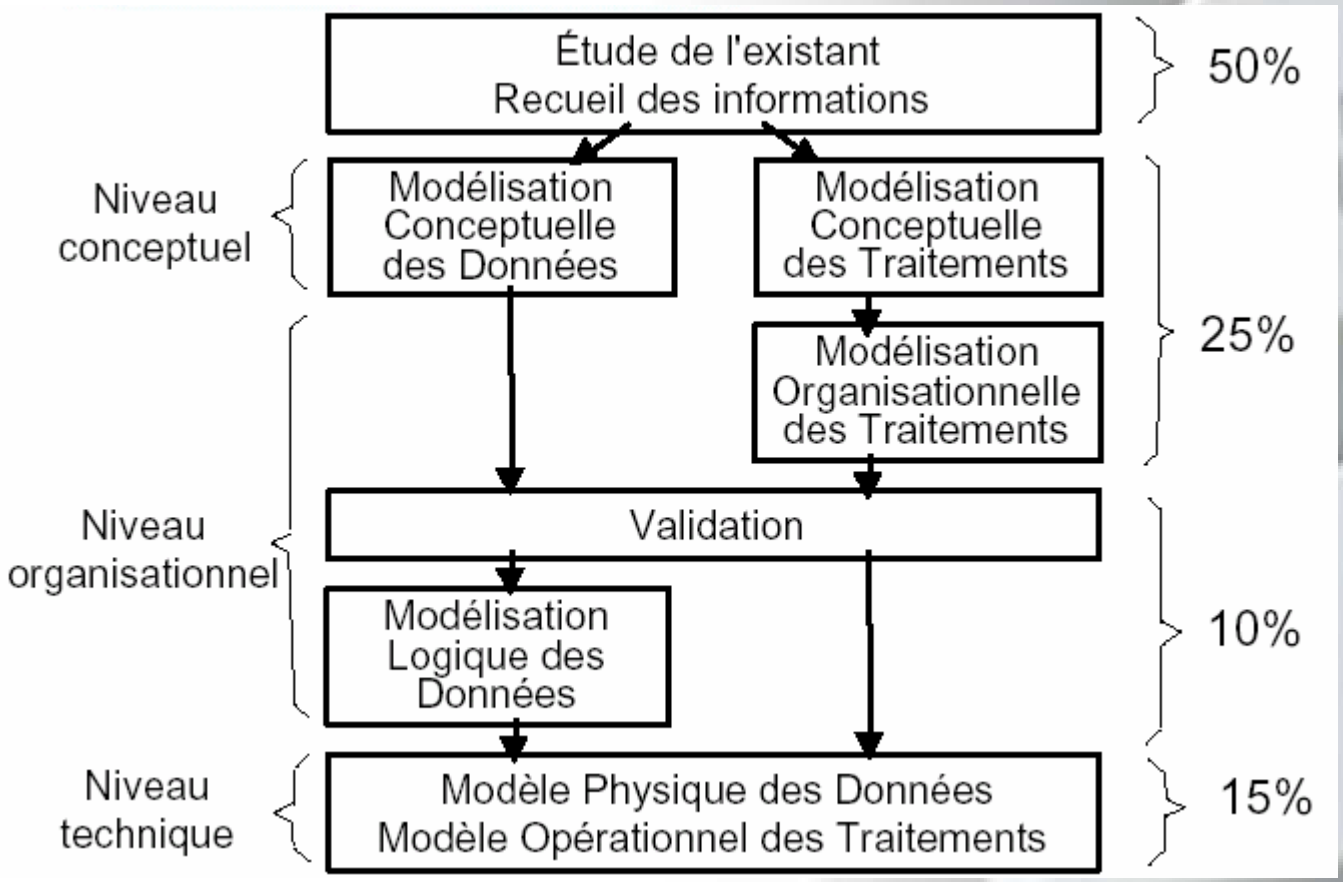
   >modèle logique (qui, quand, ou)

   >modèle physique (comment)

   >**système d'information automatisé**

2. La séparation des données et des traitements
   - l'agencement des données est plutôt stable
   - les traitements sont fréquemment remaniés
   - la séparation des données et des traitements
   - assure une longévité au modèle

Merise/Chronologie des étapes



Merise/cycle de vie

- Le cycle de vie se décompose en phases :
  - Spécifications des besoins,
  - Conception générale,
  - Conception détaillée,
  - Codage et tests unitaires,
  - Tests intégrés (intégration des modules),
  - Tests système (intégration du logiciel),
  - Recette.

Ces phases sont échelonnées dans le temps.

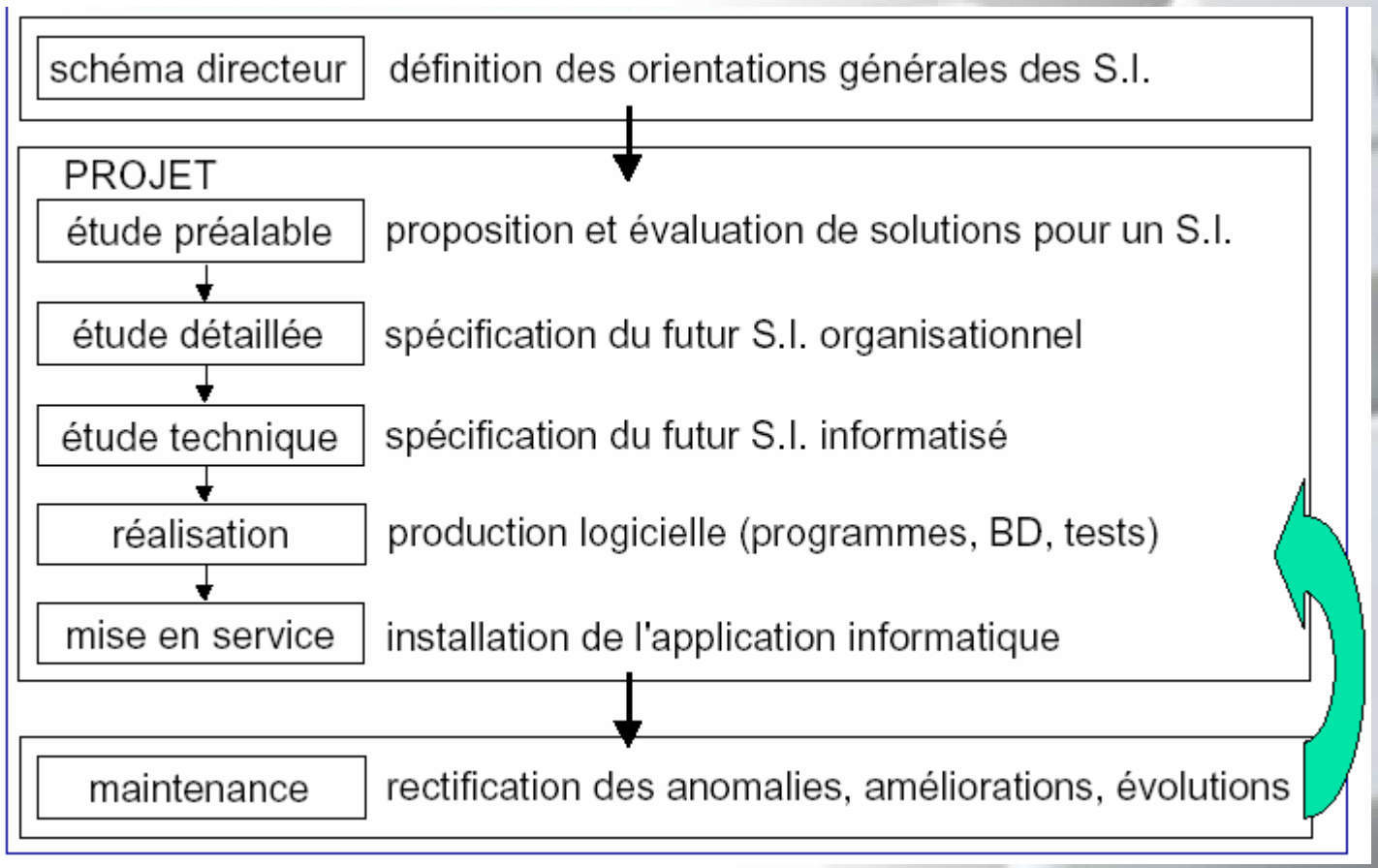
Une phase se termine par la remise de Livrables(s) validé(s).
Une phase se termine lorsque la revue de cette phase est faite.

Une phase ne peut commencer que lorsque la précédente est terminée.

=>La décomposition en phases permet donc le suivi du projet.

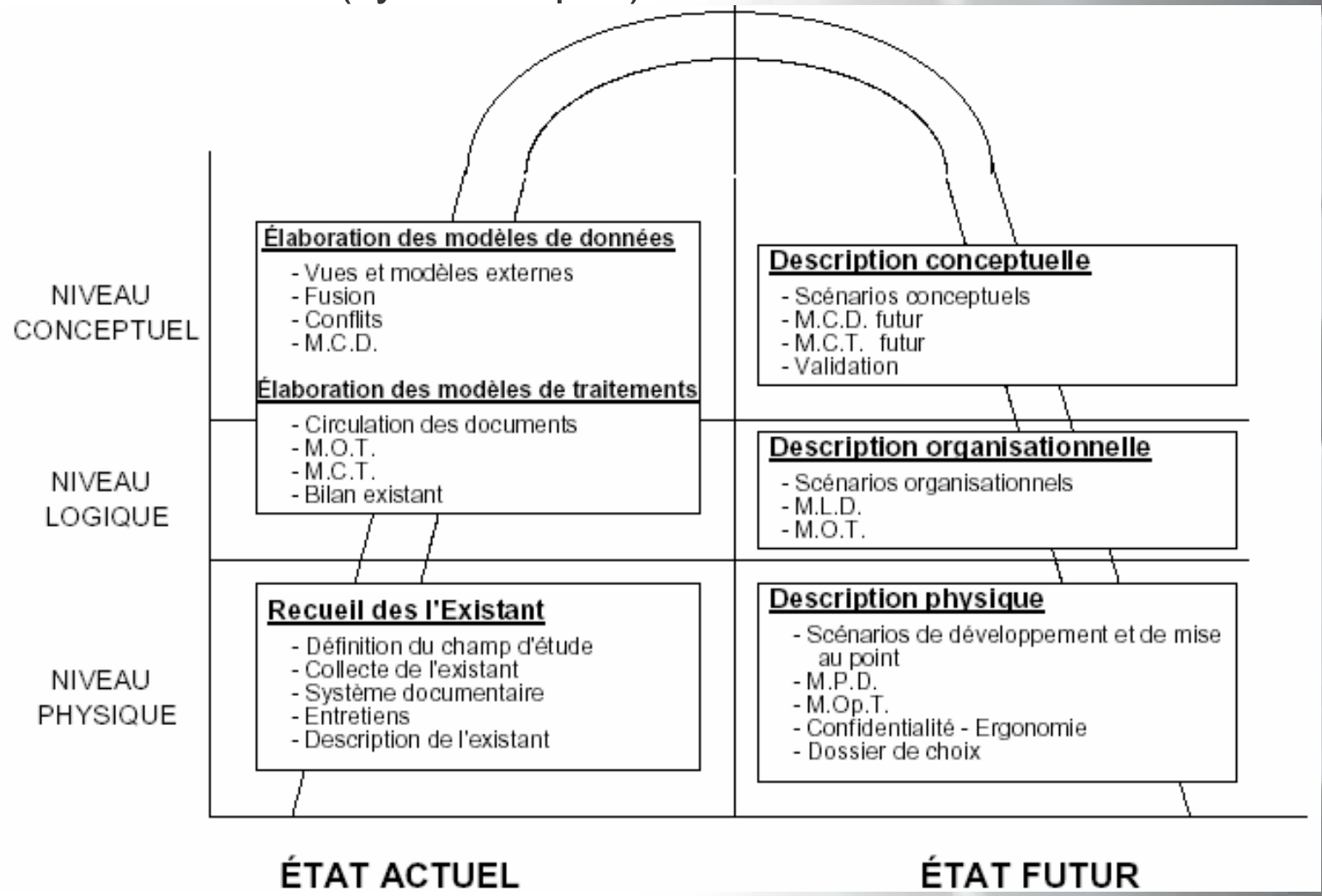
Cycle de vie

- Processus de développement d'un logiciel classique :



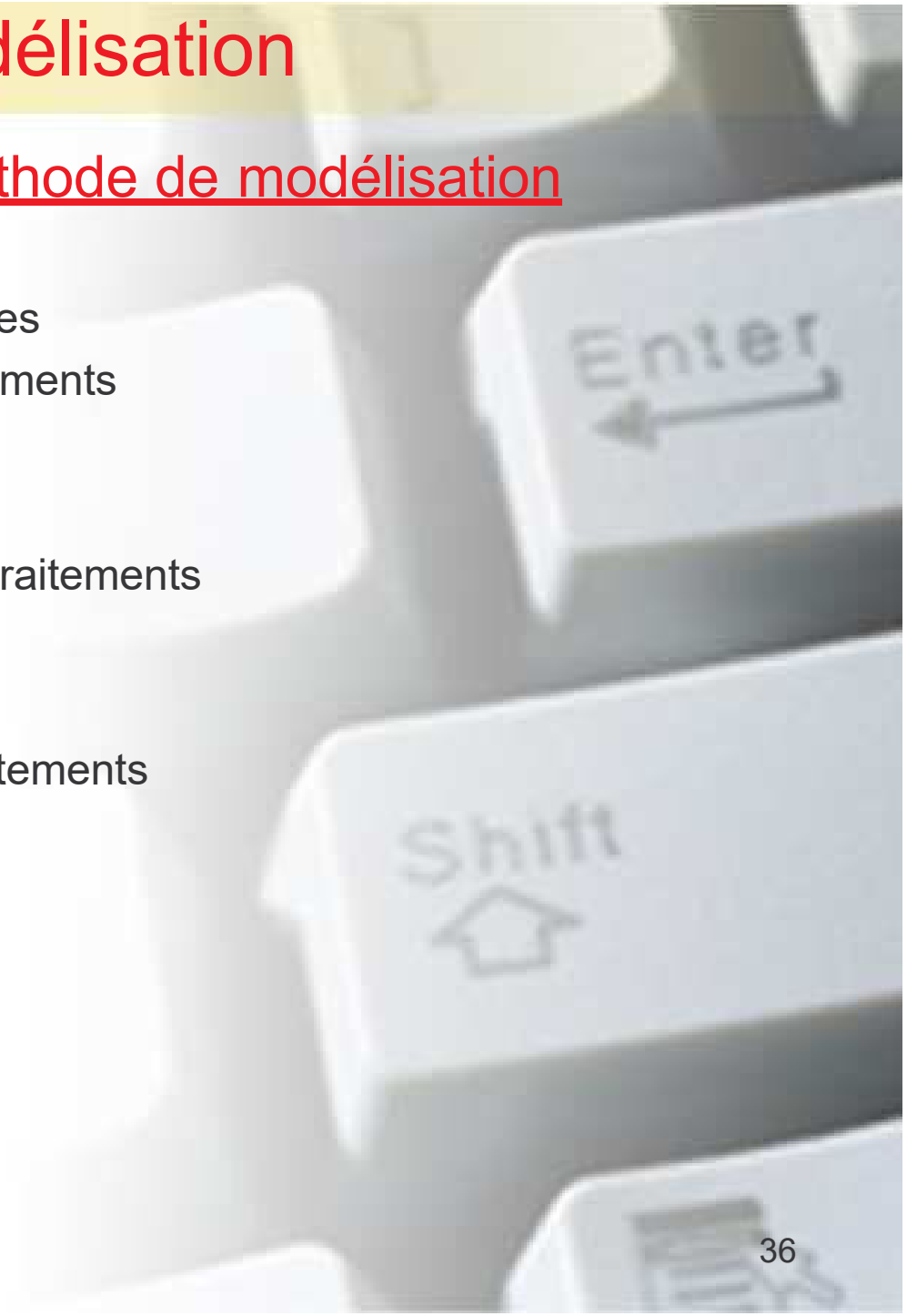
Merise/Processus de conception

- La courbe du soleil (cycle complet)



Merise/Modélisation

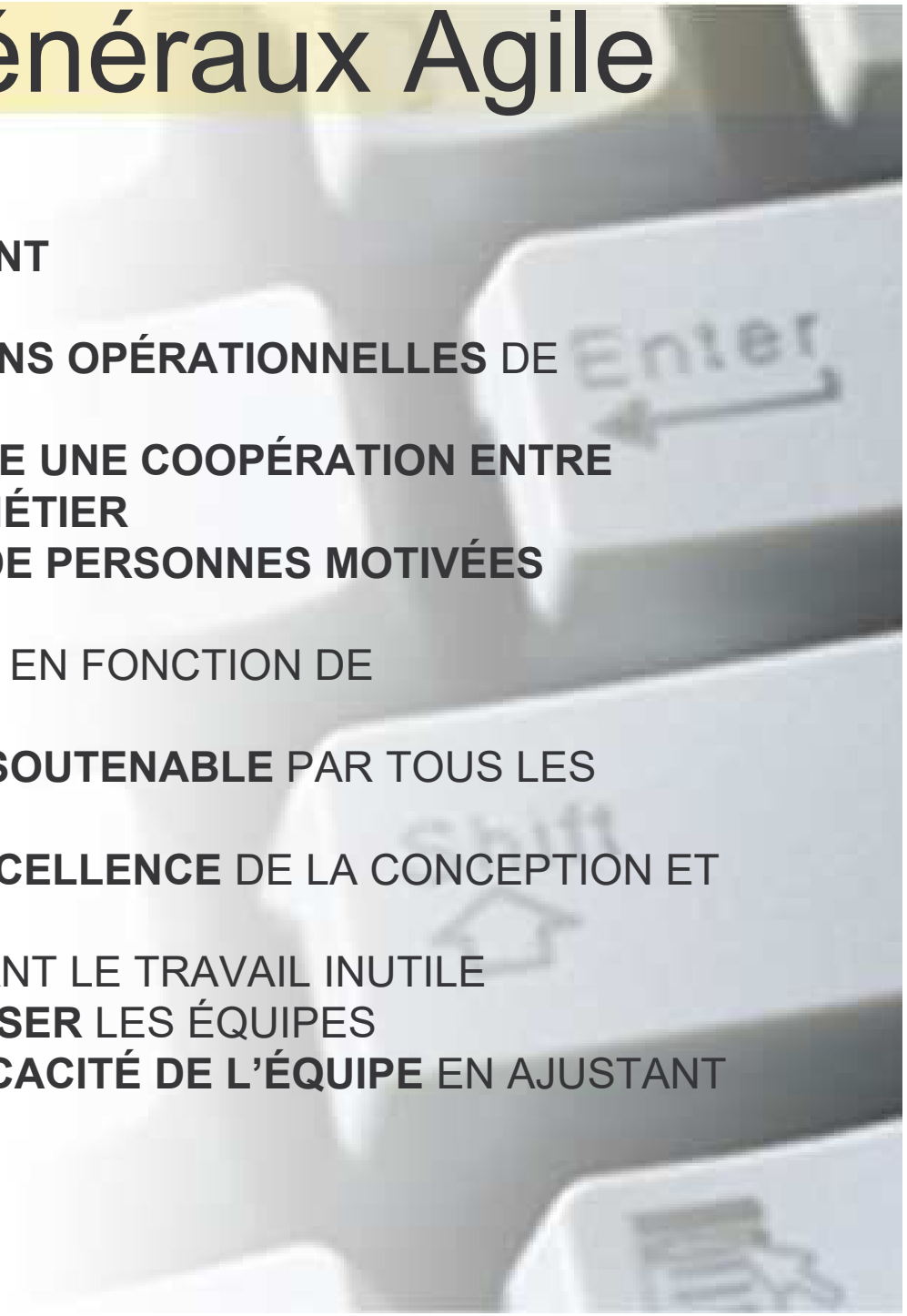
- Merise intègre sa propre méthode de modélisation
  - Niveau conceptuel
    - **M**odèle **C**onceptuel de **D**onnées
    - **M**odèle **C**onceptuel des **T**raitements
  - Niveau Organisationnel
    - **M**odèle **L**ogique de **D**onnées
    - **M**odèle **O**rganisationnel des **T**raitements
  - Niveau Physique
    - **M**odèle **P**hysique de **D**onnées
    - **M**odèle **Op**érationnel des **T**raitements



Les Méthodes Agiles (AM = Agile Modeling)

- "Méthodes itératives à planification souple qui permettent de s'adapter aux changements de contexte et de spécifications du projet" (adaptabilité,rapidité)
- En 2001, les instigateurs des principales méthodes agiles se sont réunis pour former l'Agile Alliance (http://www.agilealliance.org/home).
- Leur travail a abouti au "Manifeste pour le développement agile d'applications"… basé sur 4 valeurs principales :
  - **priorité aux personnes et aux interactions** sur les procédures et les outils,
  - **priorité aux applications fonctionnelles** sur une documentation pléthorique,
  - **priorité à la collaboration avec le client** sur la négociation de contrat,
  - **priorité à l'acceptation du changement** sur la planification.
- Ces 4 valeurs sont déclinées sur 12 principes plus généraux qui caractérisent en détail les méthodes agiles.

12 principes généraux Agile



1. **PRIORISER LA SATISFACTION DU CLIENT**
2. **ACCEPTER LES CHANGEMENTS**
3. **LIVRER EN PERMANENCE DES VERSIONS OPÉRATIONNELLES** DE L'APPLICATION
4. **ASSURER LE PLUS SOUVENT POSSIBLE UNE COOPÉRATION ENTRE L'ÉQUIPE DU PROJET ET LES GENS DU MÉTIER**
5. **CONSTRUIRE LES PROJETS AUTOUR DE PERSONNES MOTIVÉES**
6. **FAVORISER LE DIALOGUE** DIRECT
7. **MESURER L'AVANCEMENT** DU PROJET EN FONCTION DE L'OPÉRATIONNALITÉ DU PRODUIT
8. **ADOPTER UN RYTHME CONSTANT ET SOUTENABLE** PAR TOUS LES INTERVENANTS DU PROJET
9. **CONTRÔLER CONTINUELLEMENT L'EXCELLENCE** DE LA CONCEPTION ET LA BONNE QUALITÉ TECHNIQUE
10. **PRIVILÉGIER LA SIMPLICITÉ** EN ÉVITANT LE TRAVAIL INUTILE
11. **AUTO-ORGANISER ET RESPONSABILISER** LES ÉQUIPES
12. **AMÉLIORER RÉGULIÈREMENT L'EFFICACITÉ DE L'ÉQUIPE** EN AJUSTANT SON COMPORTEMENT

Les Méthodes Agiles

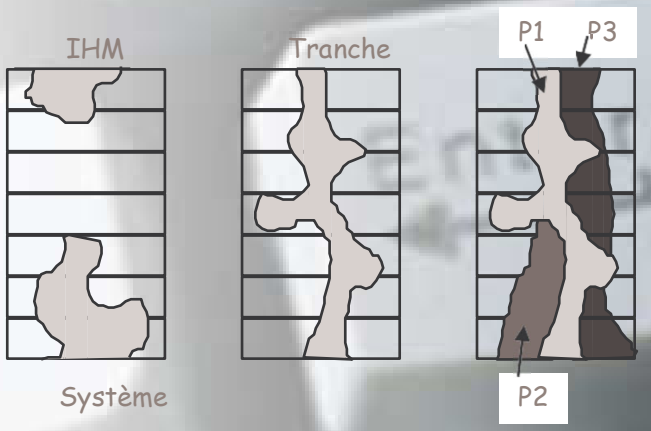
- Nombreuses
- S'appliquent en fonction de la taille et la typologie des projets

UP (Unified Process)

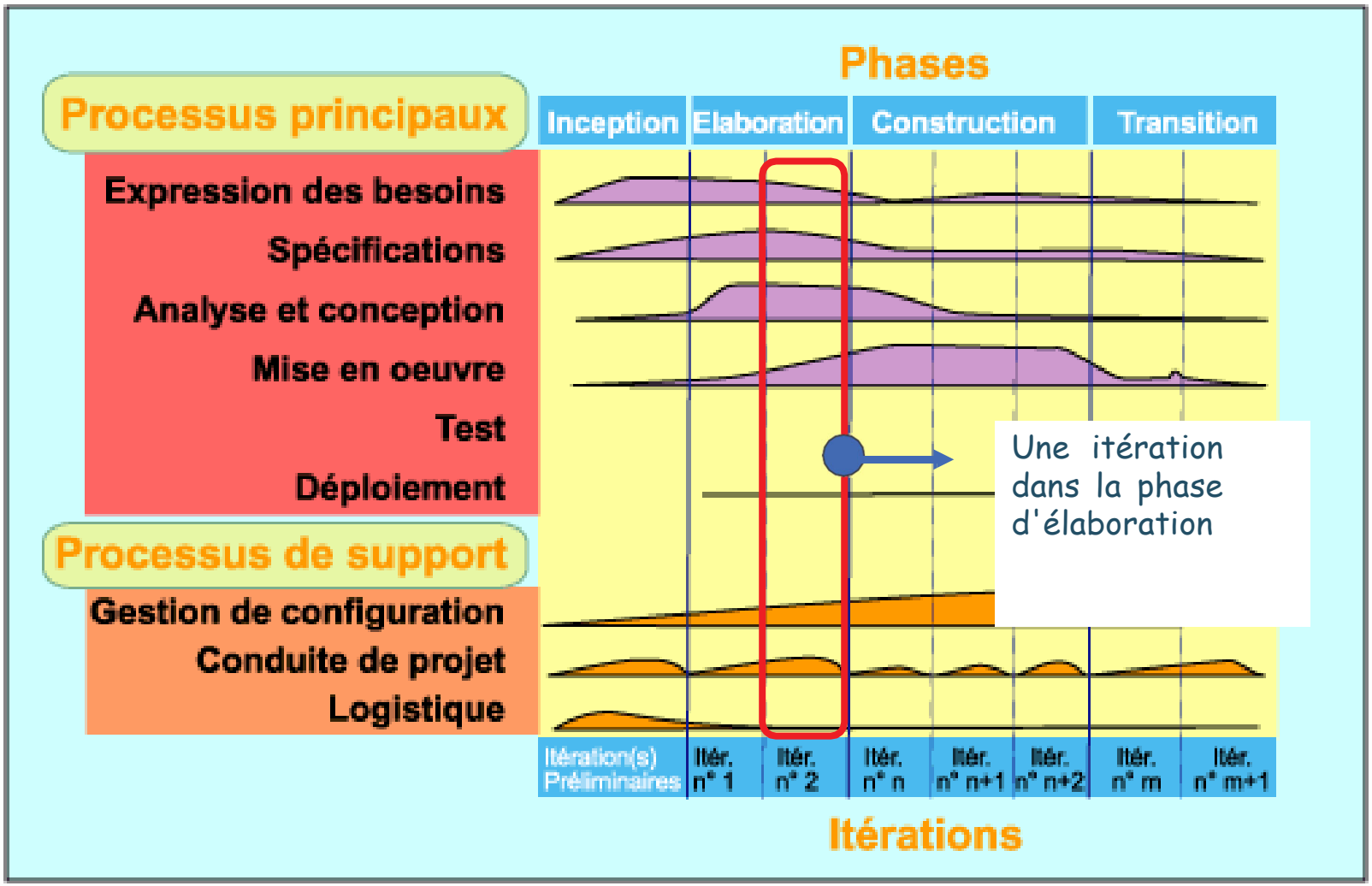
- Historique :
  - 1997
- Principes :
  - Approche itérative et incrémentale
  - L'ordonnancement des itérations est basée sur les priorités entre cas d'utilisation et sur l'étude du risque
  - Chaque prototype réduit une part du risque
    - Un prototype est un programme exécutable qui peut s'évaluer quantitativement
    - Un prototype donné est construit avec des buts précis et clairement exprimés
    - Les priorités et l'ordonnancement de construction des prototypes peuvent changer avec le déroulement du plan

UP/Phases et itérations

- Des phases
  - Inception (étude d'opportunité)
  - Elaboration (architecture, planification)
  - Construction
  - Transition
- Des itérations
  - Chaque cycle donne une génération
  - Chaque cycle est décomposé en phases
  - Chaque phase comprend des itérations
- Des incréments
  - Le logiciel évolue par incrément
  - Une itération correspond à un incrément
  - Les itérations peuvent évoluer en parallèle



UP/Phases et itérations

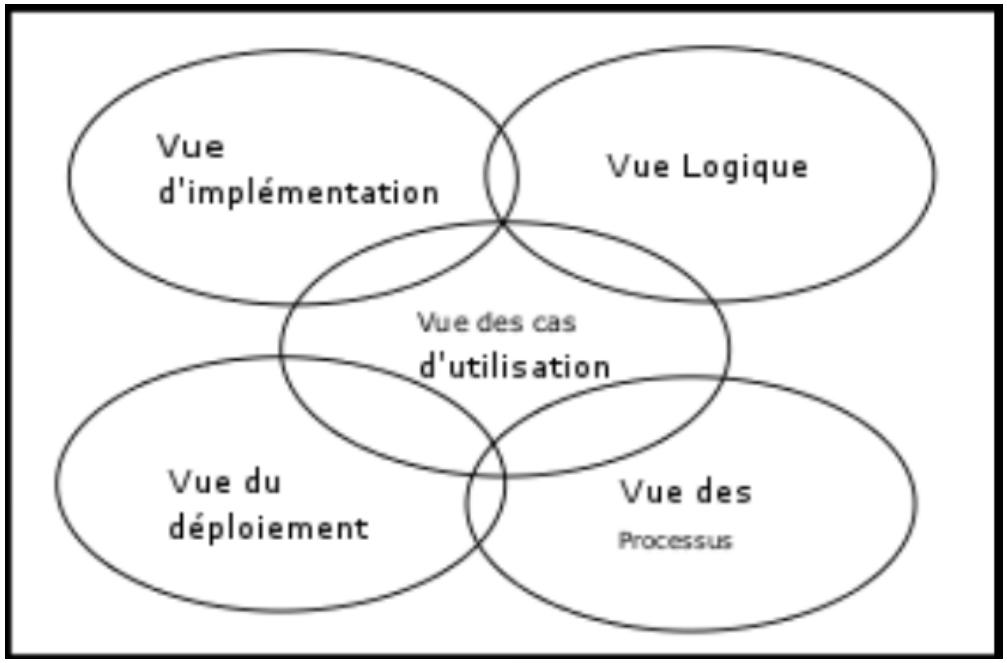


UP/Modélisation

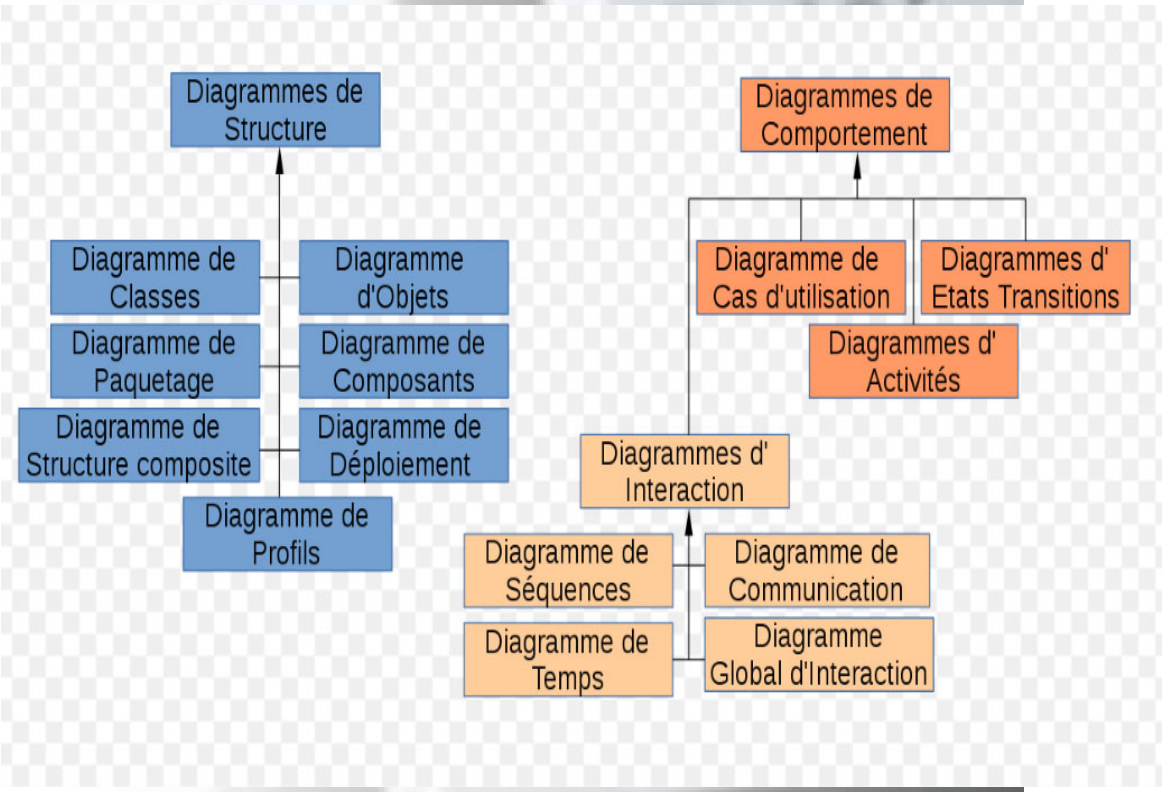
- UML
  - Diagrammes de contexte
  - Diagrammes de cas d'utilisation
  - Diagrammes d'objet
  - Diagrammes de classes
  - Diagrammes de composants
  - Diagrammes de déploiement
  - Diagrammes de collaboration
  - Diagrammes d'état-transition
  - Diagrammes d'activités
  - Diagrammes de séquences

UML

Vues d'UML

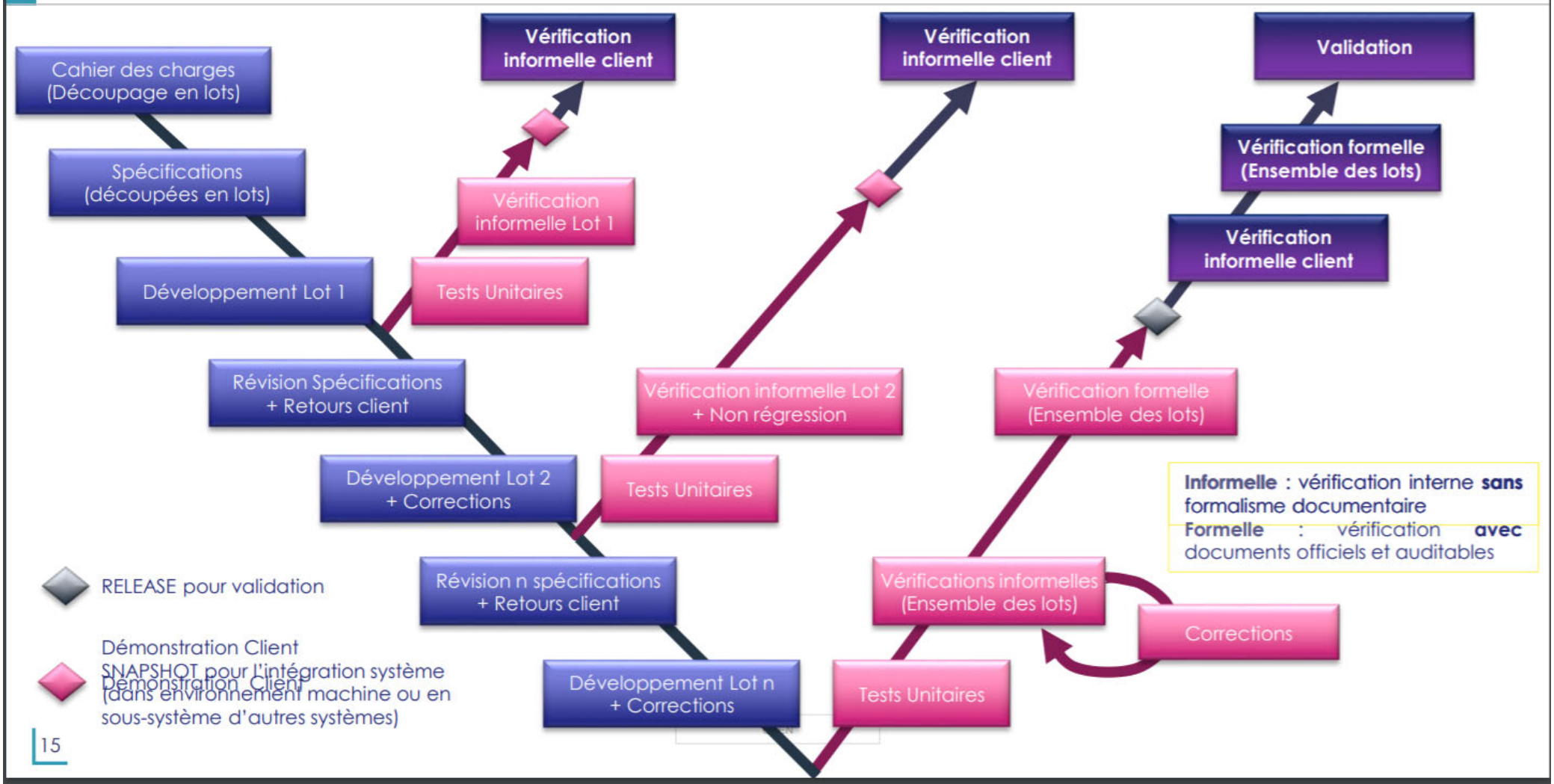


Diagrammes UML



Agiliser le cycle en V

Agiliser le cycle en V



15

Comment tester au plus tôt : agilité dans le cycle en V

Synthèse

- **MERISE** (conduite de projet) (ou SDM/S) : Approche «par la structure» (systémique).
  - Inconvénients (liés à la validation en cascade) : rigidité, manque d'adaptation, éloignement des besoins détaillés des utilisateurs, on valide que du papier,
    «effet tunnel».

- **UP/RUP** : Approche « par la structure » avec validation en cascade (pour maintenir la coherence globale) lors du premier tiers du projet. Puis, approche « par les besoins » avec construction-validation de type itératif-incrémentiel (pour assurer la conformité de l'application aux besoins de l'utilisateur).
  - Inconvénients : pas d'inconvénient «structurel», mais implique d'adapter le processus à la typologie du projet. Gestion des besoins « béton ».

Recommandations (Best Practices ?)

- En cascade :
  - Avec des Phase de taille raisonnable (<90j)
  - Avec Prototypage
  - Faire des Lots techniques ou fonctionnels

- En spirale :
  - Définir pour chaque phase, le nombre d'itérations et leur contenu précis.

- AGILE :
  - Place les besoins du client au centre des priorités du projet
  - Fonctionnalités opérationnelles plutôt que de la documentation exhaustive
  - La collaboration avec le client plutôt que la contractualisation des relations
  - L'acceptation du changement plutôt que le suivi d'un plan

TD 1 : Cycle de vie en logiciel

Objectifs : Connaître les modèles de cycle de vie d'un logiciel

1. Pourquoi parle t-on de "cycle de vie" en logiciel ?

2. Décrivez les principaux cycles de vie du logiciel, montrez leurs principales forces et faiblesses.

3. Expliquer les principales caractéristiques du cycle de développement en spirale, par un petit exemple montrer de quelle manière on peut l'utiliser dans un projet. Faites un schéma du cycle en illustrant l'enchaînement des différentes étapes pour un développement de quelques versions successives (v1.0, v2.0,…) d'un logiciel.

4. Une entreprise de génie logiciel spécialisée en objet souhaite réaliser un petit logiciel de jeux sur Internet, cette demande est inhabituelle pour cette société. Vous maîtrisez très bien la technologie nécessaire au développement de ce projet qui ne comporte pas de risque. Un cahier des charges précis est donné par le client. Que proposez vous comme cycle de vie de développement. Argumentez votre proposition, montrez les avantages et inconvénients de votre proposition par rapport à d'autres possibles.

TD2 : Modèles de Cycle de vie en logiciel (1/5)

Objectifs : Connaître les modèles de cycle de vie d'un logiciel:(Le modèle en cascade ; Le modèle en V ; Le modèle incrémental ; le modèle orienté réutilisation ; le modèle en spirale) et la technique du prototypage.

TD2 Partie 1 : Modèle de cycle de vie

1. **Exercice 1 :**Une entreprise LOG de production logiciel adopte un processus de développement logiciel qui consiste à enchainer les différentes phases de développement : étude de faisabilité, spécification, conception, implémentation, tests et livraison. Les retours en arrière entre ces différentes phases ne sont pas planifiés mais si des erreurs sont détectées pendant les tests, il est possible que l'équipe de développement réadapte la conception et/ou l'implémentation du logiciel. Le succès des projets de développement logiciel de cette entreprise est garanti seulement s'il s'agit de reproduire un projet déjà réalisé.
   - ❑ ***Question*** ⍰ Déterminez le modèle de cycle de vie utilisé par cette entreprise.

2. **Exercice 2** :Les jalons (milestones) sont des évènements qui servent à indiquer le degré d'avancement d'un projet de logiciel comme l'achèvement du manuel d'utilisateur.
   - ❑ ***Question 1*** ⍰ En quoi un modèle de cycle de vie divisé en phases aide-t-il à la gestion du développement d'un logiciel ?
   - ❑ ***Question 2*** ⍰ Quelles sont les deux caractéristiques obligatoires d'un jalon (milestone) ?

TD2 : Modèles de Cycle de vie en logiciel (2/ 5)

Exercice 3:

En considérant le cycle de vie d'un logiciel

- ***Question 1*** : Indiquer la ou les phases où est produit chacun des documents suivants : Manuel d'utilisation, conception architecturale, plan d'assurance qualité, spécification des modules, code source, cahier de charges, plan de test, manuel utilisateur préliminaire, conception détaillée, estimation des couts, calendrier du projet, rapport des tests, documentation.

- ***Question 2*** : Quelles différences y a-t-il avec un modèle de processus ?

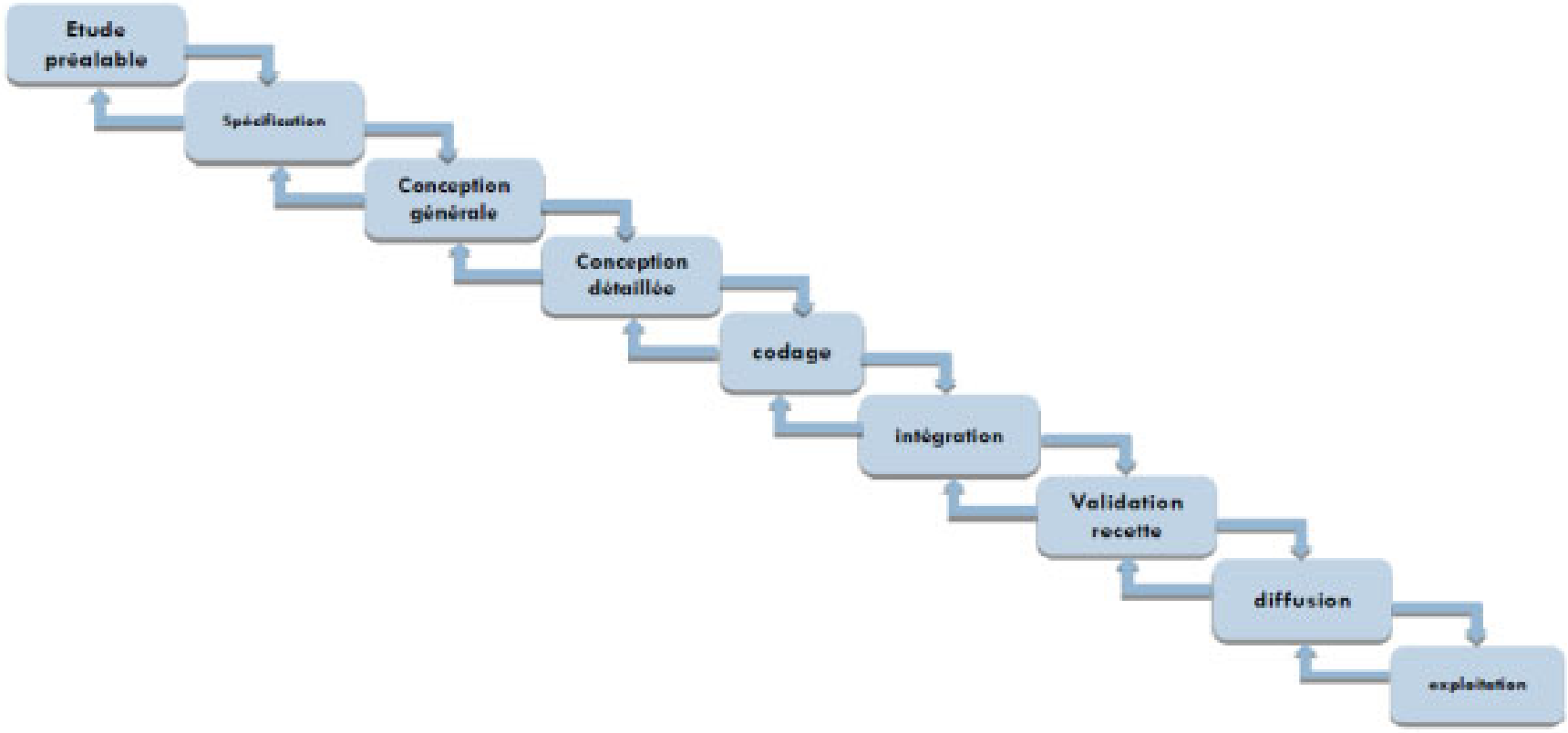
Exercice 4:

Comment peut-on combiner le modèle en cascade ou en V avec le modèle en spirale ?

TD2 : Modèles de Cycle de vie en logiciel (3/5)

Exercice 5:

Dans la représentation graphique suivante du modèle en cascade :



- ***Question 1*** ? Préciser les entrées et sorties principales (pas forcément des documents) pour chaque phase.

- ***Question 2*** ? Quelles sont les phases concernées par la vérification et/ou la validation ?

TD2 : Modèles de Cycle de vie en logiciel (4/5)

TD2 Partie 2 : Modèle de processus

Exercice 1:

Pour la peinture des murs d'une pièce, on considère :

(1) les tâches suivantes : choisir la couleur, acheter la peinture, nettoyer les murs, préparer la peinture et peindre les murs ;

(2) les artefacts suivants : choix de la couleur, pots de peinture achetés, murs propres, peinture mélangée, murs peints.

- ***Question 1*** Dessiner un modèle de processus pour la peinture des murs.

Exercice 2:

Pour assurer un enseignement à distance aux étudiants, l'instructeur divise les élèves en équipes et affiche un problème sur une page Web. Les équipes travaillent sur le problème en utilisant le tchat, ils posent des questions à l'instructeur en utilisant un forum, et ils soumettent les solutions par email. L'instructeur évalue ensuite les solutions en fonction d'un barème préétabli. Dessiner un modèle de processus pour préparer les sessions interactives. Exercice 3: Soit les trois types de tests: tests unitaires, d'intégration et d'acceptation. Dessiner un modèle de processus pour chaque type de test.

TD2 : Modèles de Cycle de vie en logiciel (5/5)

Exercice 2:

Pour assurer un enseignement à distance aux étudiants, l'instructeur divise les élèves en équipes et affiche un problème sur une page Web.

Les équipes travaillent sur le problème en utilisant le tchat, ils posent des questions à l'instructeur en utilisant un forum, et ils soumettent les solutions par email.

L'instructeur évalue ensuite les solutions en fonction d'un barème préétabli.

- ***Question*** ? Dessiner un modèle de processus pour préparer les sessions interactives.

Exercice 3:

Soit les trois types de tests: tests unitaires, d'intégration et d'acceptation.

- ***Question*** ? Dessiner un modèle de processus pour chaque type de test.