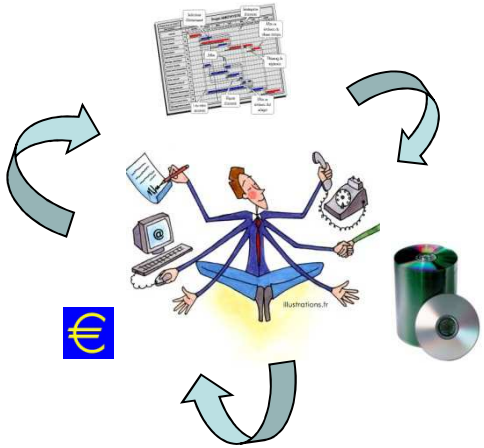
V – Cycle de vie d'un projet



Gestion de Projet

Master Pro Gestion et Informatique

2012-2013

DAUPHINE UNIVERSITÉ PARIS

Plan du chapitre

- **Les grands principes de découpage**
- **Le découpage classique**
- **Les modèles de développement**
- **L'expression des besoins et la définition du périmètre**
- **Analyses des besoins et conception**
- **Réalisation**
- **Qualification – Intégration – Validation – Recette - VABF**
- **Mise en production – Phase pilote**
- **Généralisation et VSR**

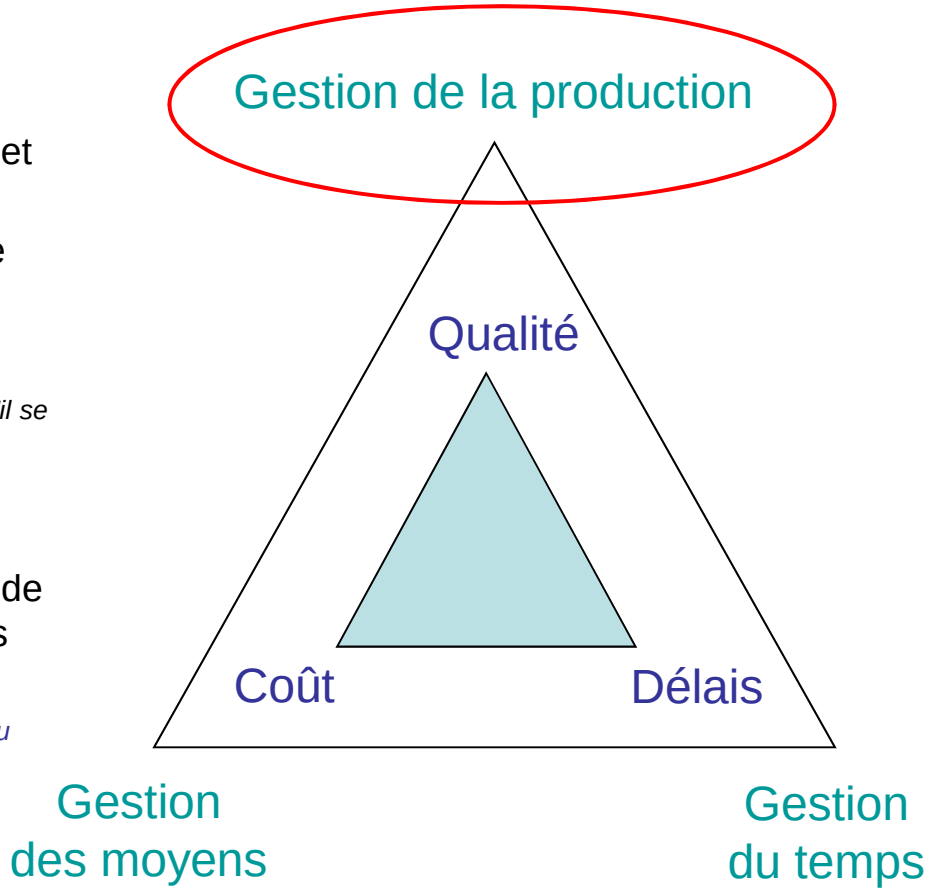
Les grands principes de découpage (1/3)

- Découper le travail à réaliser pour maîtriser la gestion de la production
  - Pouvoir répartir dans le temps la production et les ressources
  - En s'appuyant sur l'approche cartésienne de réduction de la difficulté

    cf Second principe exposé dans le ***Discours de la méthode*** *«diviser chacune des difficultés (…) en autant de parcelles qu'il se pourrait et qu'il serait requis pour mieux les résoudre ».*

    Descartes.

  - Et en s'appuyant sur l'approche systémique de prise en compte des liens entre les éléments
    - *s'attacher d'avantage aux échanges entre les parties du système qu'à l'analyse de chacune d'elles,*
    - *raisonner par rapport à l'objectif du système,*
    - *Établir les états stables possibles du système*.



Les grands principes de découpage (2/3)

- Découper un projet consiste à identifier des **sous-ensembles quasi-autonomes**, présentant les caractéristiques suivantes
  - Chaque sous-ensemble donne lieu à un **résultat** bien identifié
  - La **charge** propre à chacun peut être évaluée
  - Les contraintes **d'enchaînement** entre sous-ensembles sont repérables : certains peuvent être réalisés parallèlement, d'autres sont liés entre eux par des contraintes d'antériorité
  - Le découpage est fait à des **mailles** différentes, un sous-ensemble étant à son tour décomposé.

Les grands principes de découpage (3/3)

- Les deux grands critères de découpage d'un projet sont :

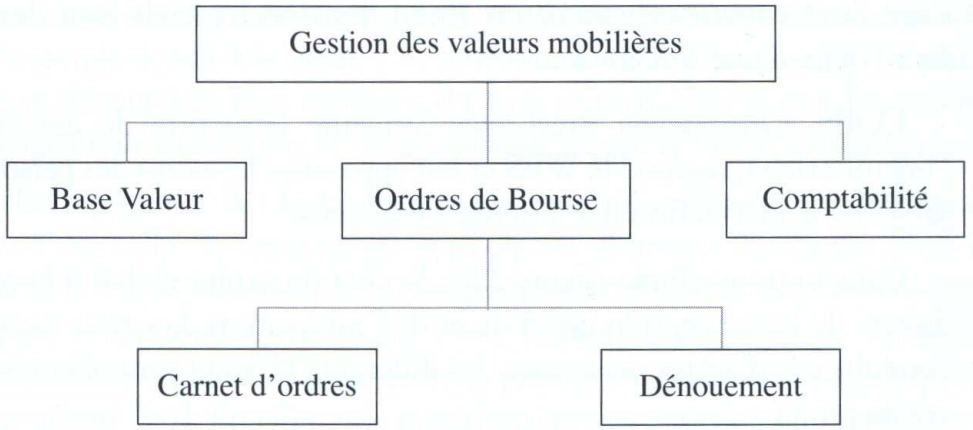
  – Le critère **temporel** :
    - Successions d'étapes constituées de phases, chaque phase étant à son tour décomposée en tâches.
    - A chaque tâche est associé un résultat à atteindre, appelé tangible ou livrable et qui peut faire l'objet d'un engagement contractuel.
    - Le critère temporel balise et guide le projet = **cycle de développement**
    - Généralement de type descendant (top down) : les résultats sont de plus en plus précis et la maille d'étude de plus en plus fine.

  – Le critère **structurel**
    - Organiser le travail en se basant sur la structure du produit final : le découpage fait apparaître les différents modules qu'il faut obtenir
    - La structure peut-être analysée sous l'angle statique (les objets ou données manipulées) et/ou dynamique (les processus , traitement qu'on applique aux objets).

Les découpages normalisés (1/3)

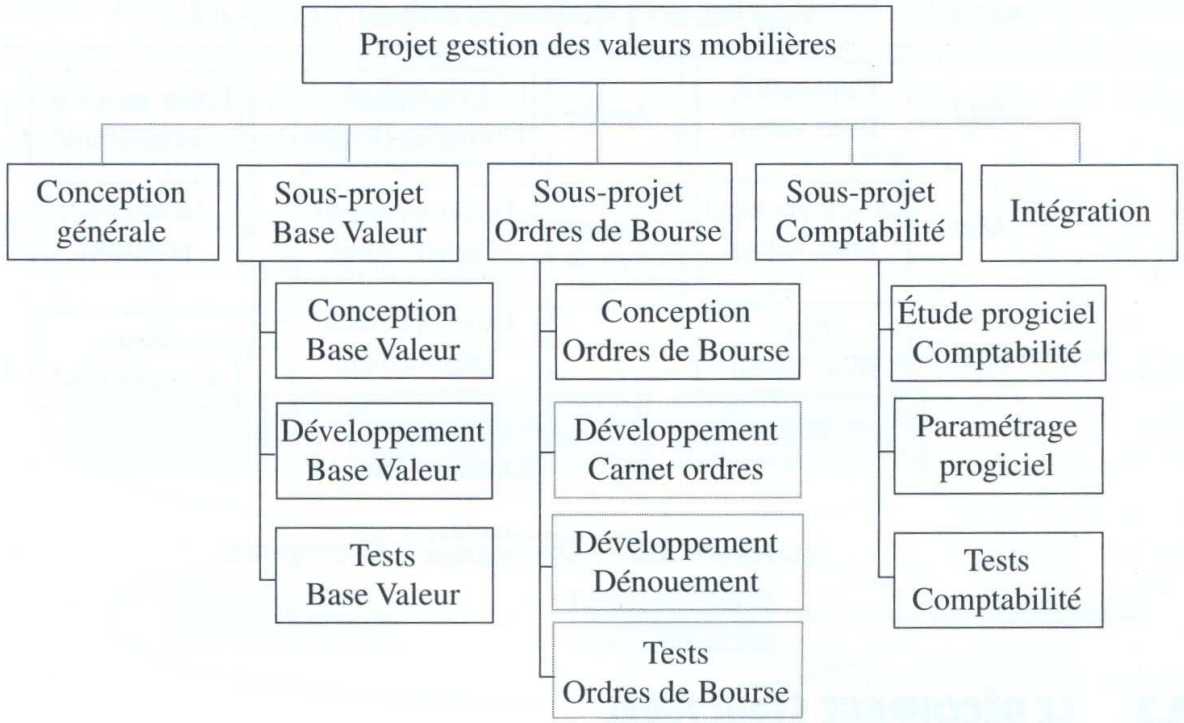
- Les normes internationales proposent 3 découpages : **PBS**, **WBS** et **OBS**

- **PBS = Product Breakdown Structure** (structure de décomposition du produit)
  - Découpage structurel = différents composants du produit final
  - Ex : progiciel de gestion de valeurs mobilières : découpage en modules
    - Référentiel des titres (Base Valeur)
    - Tenue de la comptabilité titres (Comptabilité)
    - Gestion d'un carnet d'ordres (Ordres de Bourse) découpée en enregistrement des ordres et traitement administratif des ordres effectivement passés.



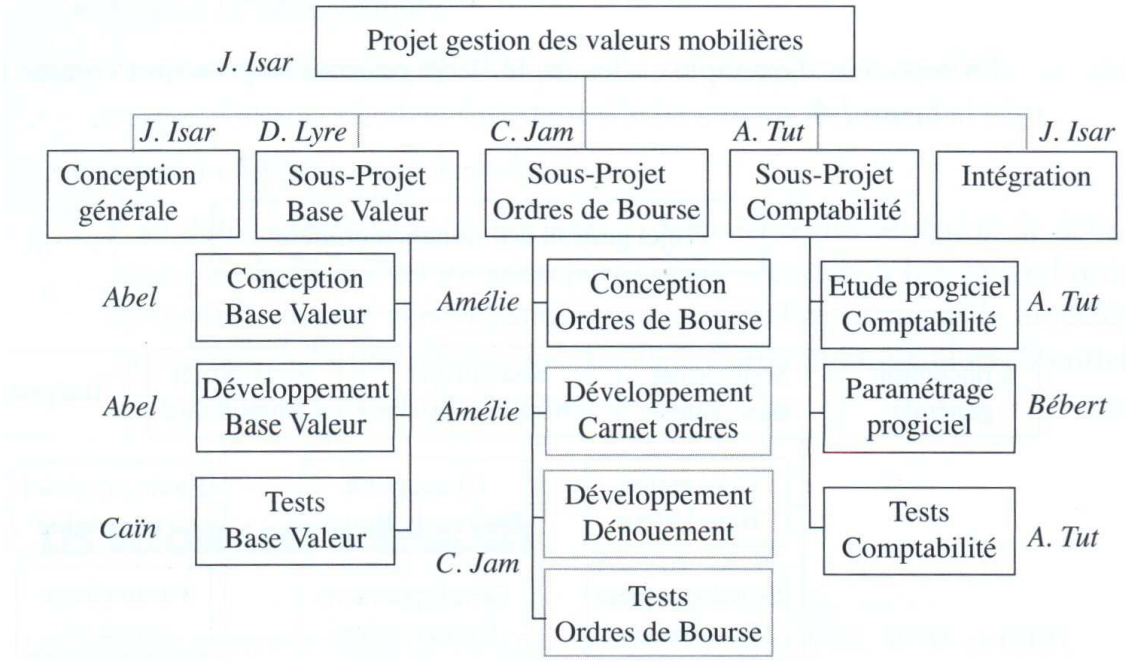
Les découpages normalisés (2/3)

- **WBS = Work Breakdown Structure** (structure de décomposition du travail)
  - Découpage structurel et temporel = représente la façon de parvenir au résultat du PBS.
  - Ex : progiciel de gestion de valeurs mobilières
    - Conception générale sur l'ensemble du domaine
    - 3 sous-projets correspondant aux modules , comportant des phases de conception, des développements puis des tests
    - Une intégration des trois sous parties du logiciel.

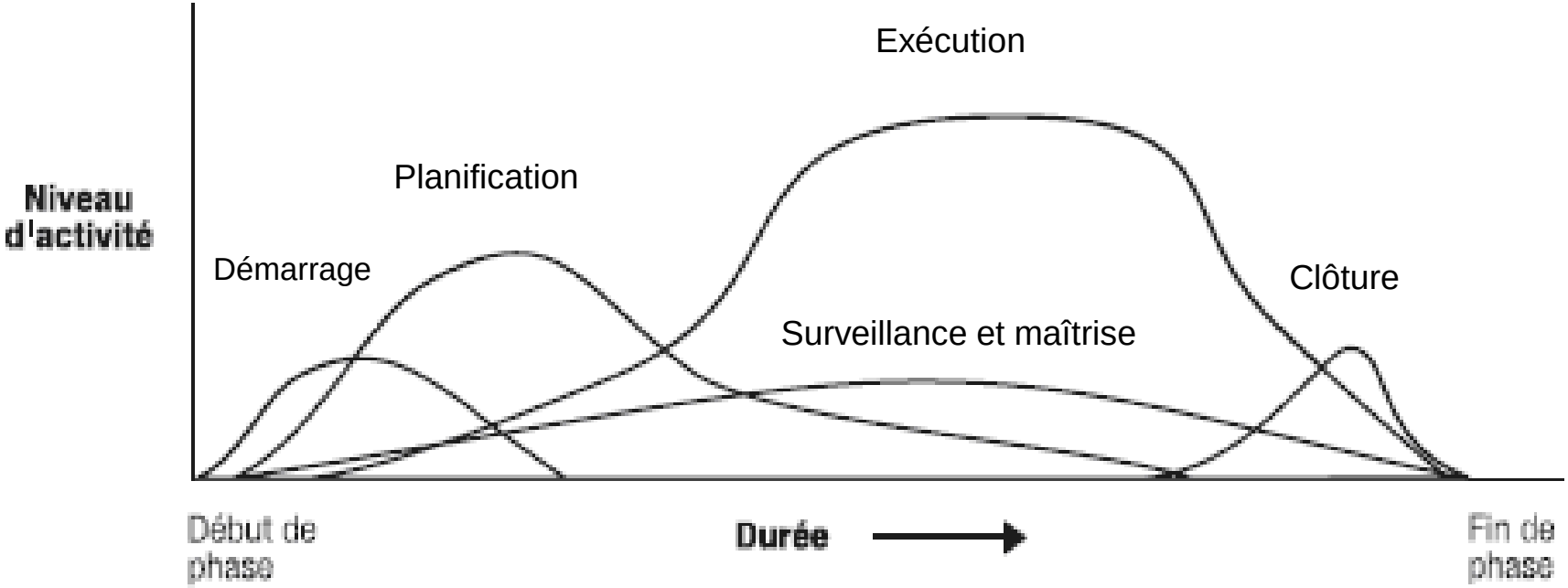


Les découpages normalisés (3/3)

- **OBS** = **Organisation Breakdown Structure** (structure de décomposition de l'organisation)
  - Découpage structurel, temporel avec ajout de l'organisation = représente les personnes responsables de la production des différents éléments.
  - Ex : progiciel de gestion de valeurs mobilières
    - Le Chef de projet global a également la responsabilité directe de la conception générale et de l'intégration
    - 3 sous-projets conduits par des CP dédiés.
    - Identification des ressources affectées aux travaux



Les 5 groupes de processus



Le découpage « classique » (1/8)

- Principales méthodes de développement de systèmes d'information des 40 dernières années ont reposé sur un découpage temporel de référence

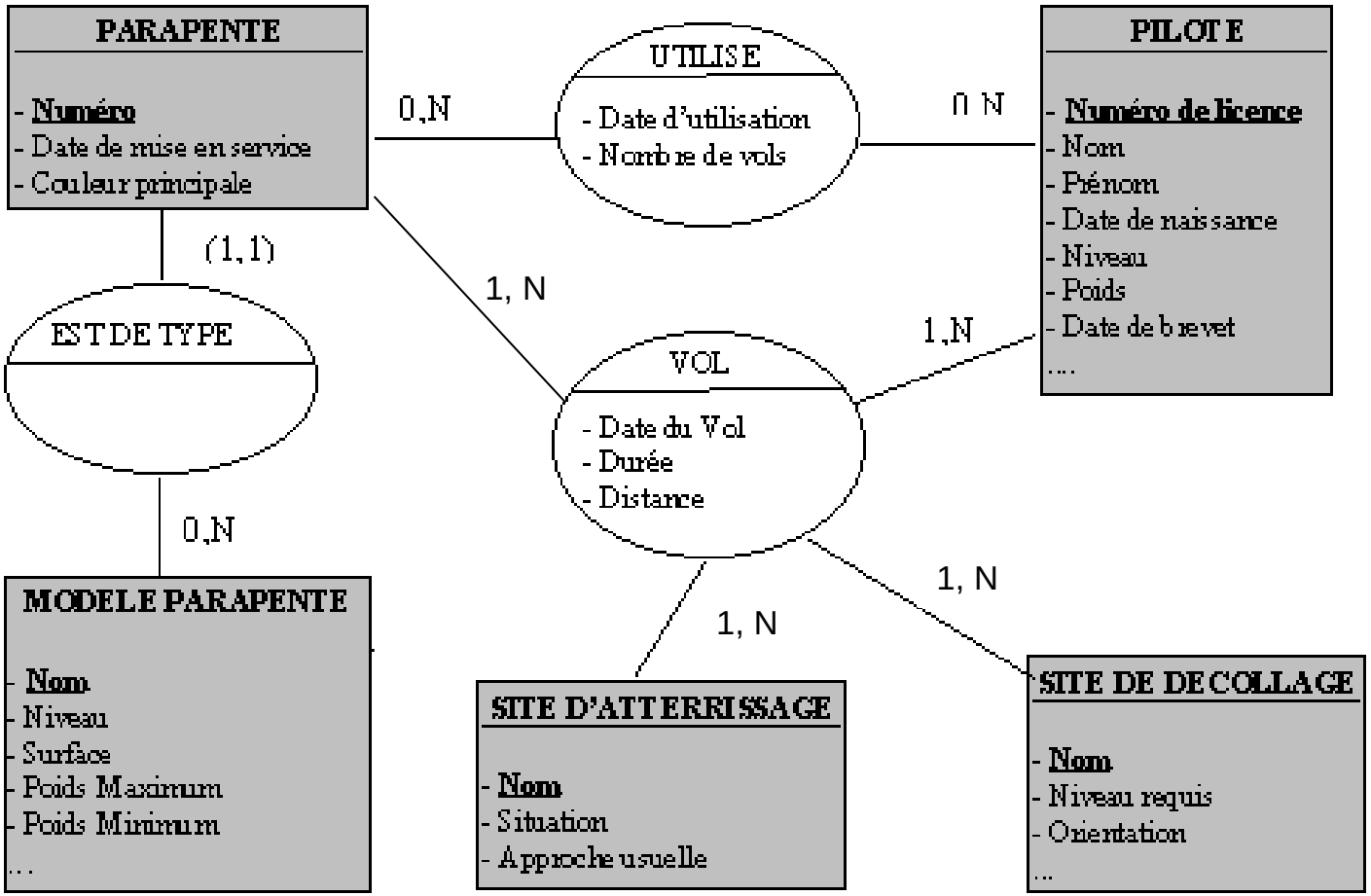
  – **MERISE** : Méthode d'informatisation (1974 – 1978) initiée par les français Hubert Tardieu, Jean-Louis Lemoigne travaillant sur des projets au ministère de l'industrie et dans les services de l'équipement, relayés par des travaux de l'université d'Aix-Marseille et de l'INRIA.
    - MCP : Méthode de Conduite de Projet informatique créée par GEDIN en 1975, modifiée en 1986 en s'inspirant de MERISE

  – La norme **AFNOR Z67-101** (1984) reprend dans ces « recommandations pour la conduite de projets informatiques » les éléments des méthodes MERISE / MCP.

  – **SDMS** : méthode américaine axée sur la description minutieuse du découpage en étapes et phases et qui est souvent couplée avec MERISE

NORME AFNOR Z67-101	MERISE	SDMS
	Schéma directeur	
Étude préalable Exploration Conception Appréciation	Étude préalable Observation Conception/Organisation Appréciation	DBS *(Définition des besoins du système)* CAS *(Conception de l'architecture du système)*
Conception détaillée	Étude détaillée	SES *(Spécifications externes du système)*
Réalisation	Étude technique Réalisation	SIS *(Spécifications internes du système)* Programmation Test
Mise en œuvre	Mise en œuvre	Conversion Installation
Évaluation	Qualification	Bilan

Aparté sur le MCD de la méthode Merise



Le découpage « classique » (2/8)

- **Le schéma directeur (SD)**

  – Objectif : définir le scénario d'évolution du patrimoine informatique, sous l'un ou l'autre des ces 3 angles
    - Évolution de l'architecture technique (matériels, réseaux)
    - Évolution de l'architecture applicative
    - Evolution de la fonction informatique (méthodes, normes, outils)

  – Champ d'application : l'organisation tout entière

  – Livrables
    - Photographie de la situation existante : cartographie des domaines applicatifs, modélisation des principaux concepts
    - Diagnostic
    - Définition des objectifs et priorités par domaine et par application.
    - Orientations d'évolutions choisies , à partir de quelques scénarii.

Le découpage « classique » (3/8)

- **L'étude préalable (EP)**
  - Contexte : à l'issue d'un SD, ou pour repenser une application vieillissante, ou répondre à un besoin nouveau.
  - Objectif :
    - Faire les choix structurants pour la future application : évaluer l'adéquation de la solution envisagée, évaluer l'investissement
    - Fournir une base de référence pour la suite du projet
  - Champ d'application : le projet
  - Livrables :
    - Rapport d'étude préalable , qui pourra servir de cahier des charges pour l'étude détaillée.

Le découpage « classique » (4/8)

- **L'étude détaillée (ED)**
  - Objectif : concevoir et décrire de façon exhaustive la solution sur tout le champ de l'étude
  - Champ d'application : le projet
  - Livrables
    - Spécifications de la solution, validées par les futurs utilisateurs et les informaticiens.
    - Toute la vision externe du système (IHM : maquettes d'écrans, cinématiques), les traitements, les sorties (maquettes d'états).
    - Représente le cahier des charges pour la réalisation

- **L'étude technique (ET)**
  - Objectif : optimiser les structures physiques de données et construire les traitements en anticipant la réutilisation de code
  - Champ d'application : le projet
  - Livrables :
    - Dossiers programmes
    - Normes techniques
    - Structure physique des données

Le découpage « classique » (5/8)

- **La réalisation (REAL) – ou « Développement »**

  – Objectif : produire un logiciel testé
    - Programmation
    - Élaboration de jeux d'essais
    - Test
    - Recette provisoire

  – Champ d'application : le projet

  – Livrables :
    - Logiciel
    - PV de recette (la recette conditionne en général le paiement, dans une relation contractuelle avec un fournisseur).

Le découpage « classique » (6/8)

- **La mise en œuvre (MEO)**

  – Objectif : Préparer le démarrage effectif de la nouvelle application
    - Paramétrage
    - Reprise ou alimentation en données
    - Développement des interfaces
    - Formation des utilisateurs
    - Installation d'environnement d'exploitation

  – Champ d'application : le projet

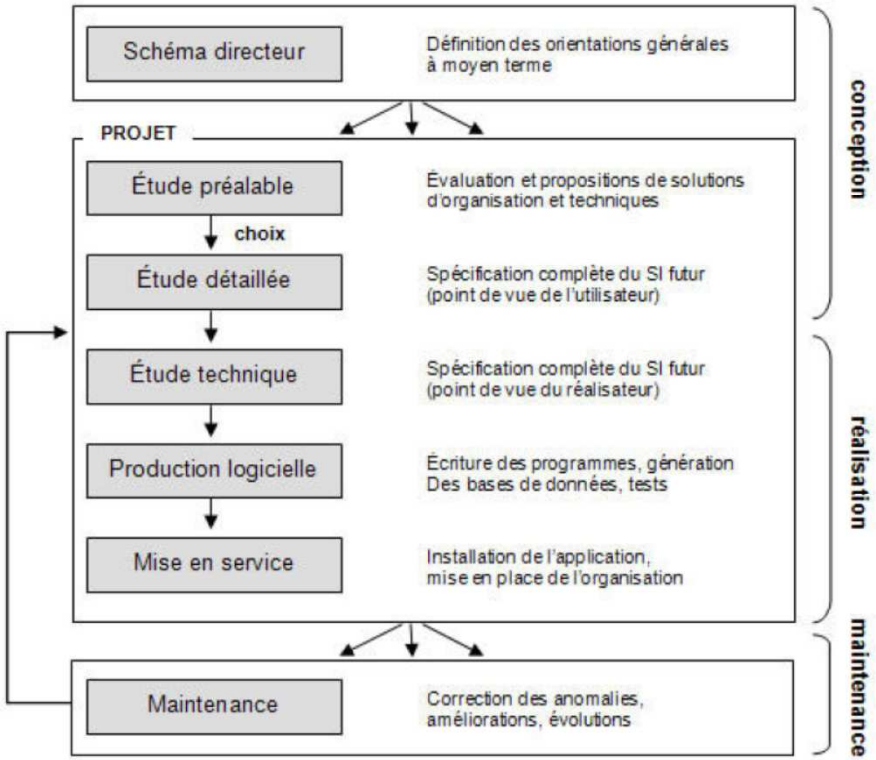
  – Livrables
    - Un logiciel installé en production, prêt à être exploité
    - Le plan de formation et les moyens associé.

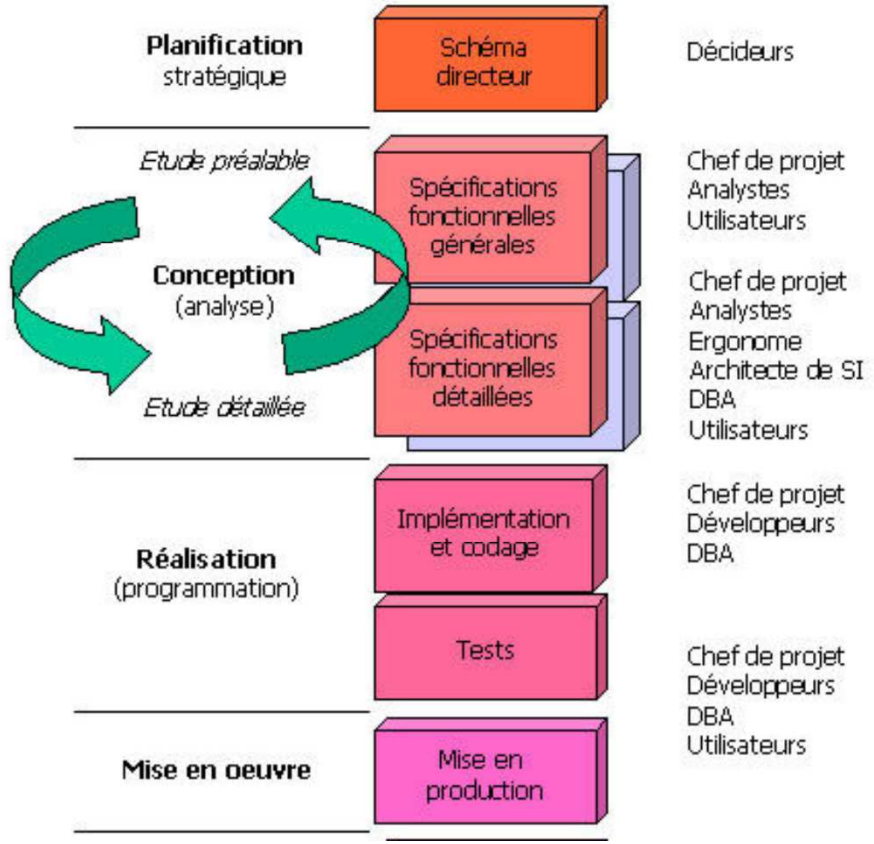
Le découpage « classique » (7/8)

- **La qualification (QUALIF)**
  - Objectif : réaliser des tests dans l'environnement opérationnel et tirer un bilan du système installé.
  - Champ d'application : le projet
  - Livrables :
    - Bilan des tests

- **La maintenance**
  - Objectif : corriger et adapter le logiciel aux évolutions de l'entreprise
  - Champ d'application : la solution en production
  - Livrables : patches correctifs ou évolutifs

Le découpage « classique » (8/8)





Les modèles de développement

- Nécessaire **adaptation** du découpage temporel au contexte
  - La démarche unique fonctionne mal sur certains types de projets
  - Nécessité de construire le découpage temporel en fonction
    - Des caractéristiques de l'entreprise
    - Des caractéristiques des projets

- Définition de découpages temporels génériques, appelés **« modèles de développement »** ou **« cycle de vie »** des projets

Méthodes traditionnelles
- Cascade
- En V
- En W
- Code and fix
- Développement évolutif
- En spirale

Méthodes Agiles
- RAD
- ASD
- DSDM
- Scrum
- XP

Méthodes Spécifiques
- Transformation automatique
- ERP

Les modèles de développement : la cascade

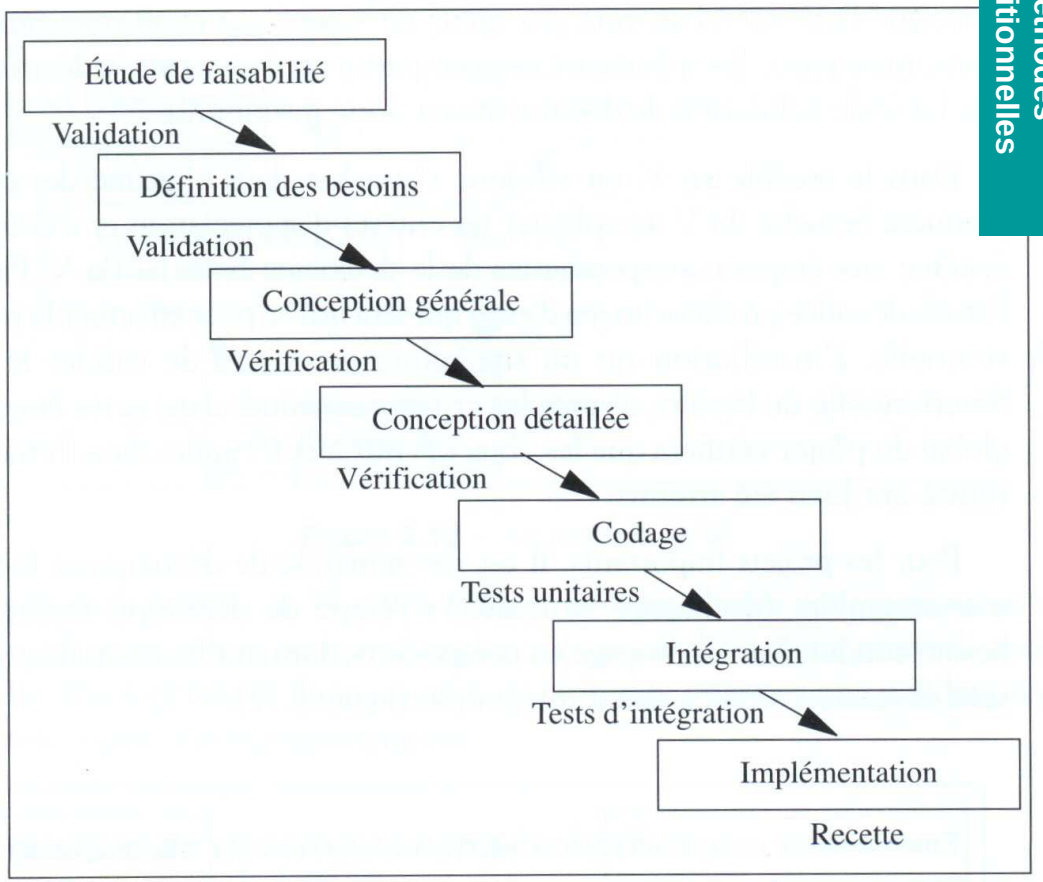
- **Le modèle de la cascade *(waterfall model)***

  – Modèle issu des études de ROYCE en 1970

  – Objectif : jalonner rigoureusement le processus, définir de façon précise les rôles respectifs des fournisseurs et du client.

  – Chaque phase de l'approche descendante donne lieu à une validation officielle (revue de fin de phase) : on en passe à l'étape suivante que si le résultat du contrôle est satisfaisant.

  – Pas de retour possible sur les options validées à l'issue d'étapes antérieures.

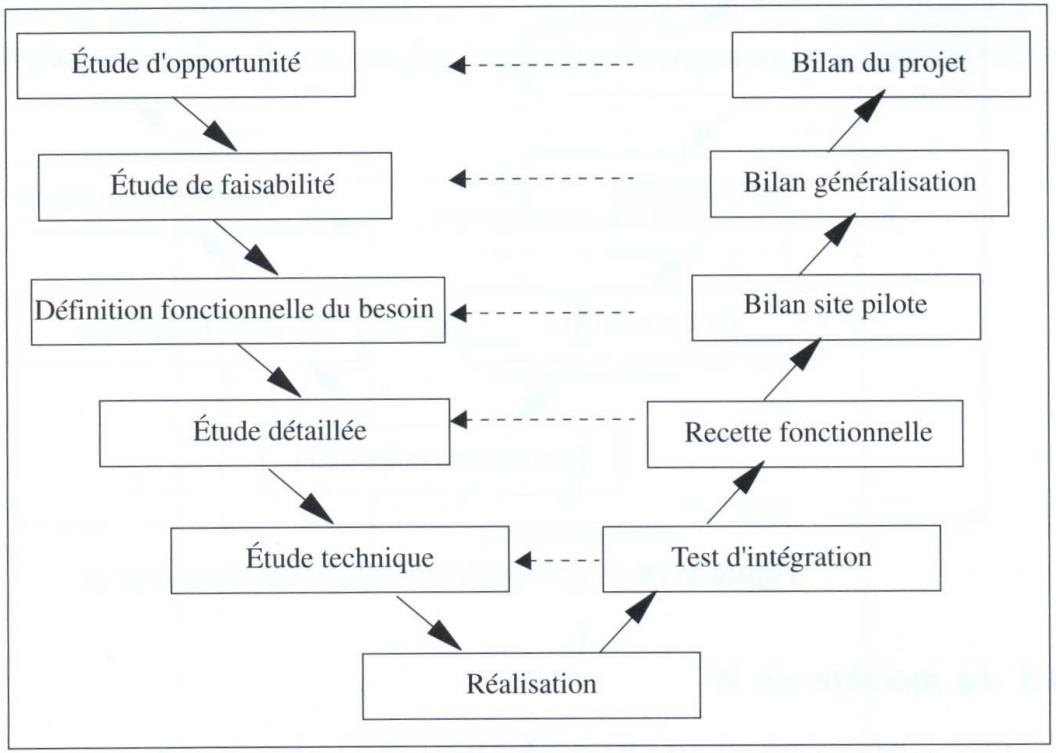


Cascade : zoom sur les activités et livrables

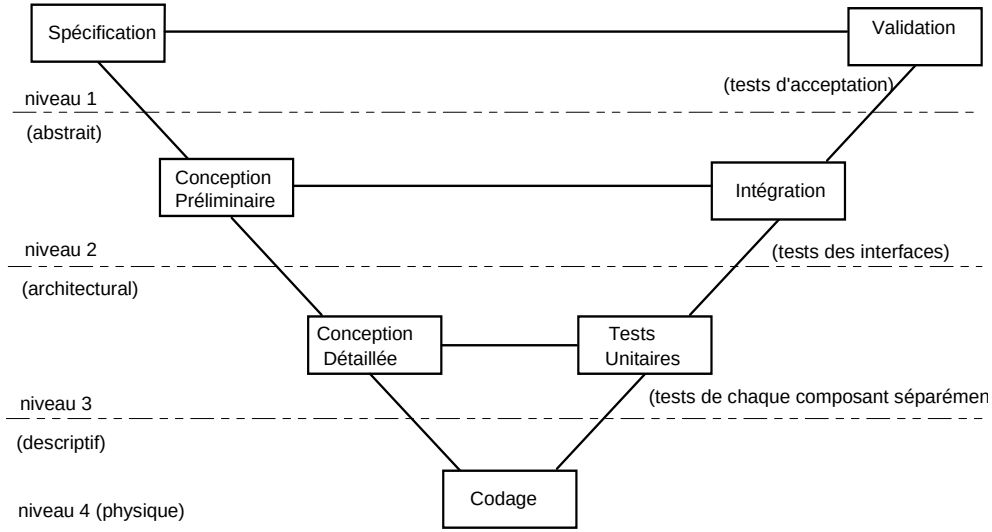
Phase	Activités	Résultats (essentiellement des documents)
TUDE REALABLE	Etude de faisabilité Analyse des besoins	Rapport d'étude Schéma directeur Grandes lignes des besoins Cahier des charges
LANIFICATION t PECIFICATION	Organisation et mise en place du projet - Définition des besoins - Définition de la stratégie de test - Définition des interfaces utilisateurs	Plan de développement (référentiel dont le planning) Spécification technique des besoins Spécification (ou plan) des tests d'acceptation Manuel utilisateur préliminaire
ONCEPTION RELIMINAIRE	Conception de l'architecture - Définition des interfaces - Définition des tests d'intégration	Architecture des modules logiciels Spécification des interfaces Spécification (ou plan) des tests d'intégration
ONCEPTION ETAILLEE	Analyse détaillée des modules - Définition des tests unitaires	Spécification des modules - Spécification (ou plan) des tests unitaires
ODAGE	Codage ou programmation - Tests unitaires de chaque module	Code source des modules - Comptes-rendus des tests unitaires
NTEGRATION	Intégration - Tests d'intégration - Recette usine (tests de validation chez le fournisseur)	 Comptes-rendus des tests d'intégration Comptes-rendus des tests de validation Manuel utilisateur final
NSTALLATION	Livraison chez le client - Installation et mise en œuvre - Recette (validation opérationnelle)	Bordereau de livraison Comptes-rendus des tests de validation Procès verbal d'acceptation par le client
XPLOITATION t AINTENANCE	Formation des utilisateurs - Corrections des anomalies - Améliorations ou évolutions - Gestion en configuration des livraisons	 Rapports d'anomalies Demandes d'évolutions - Versions du produit logiciel

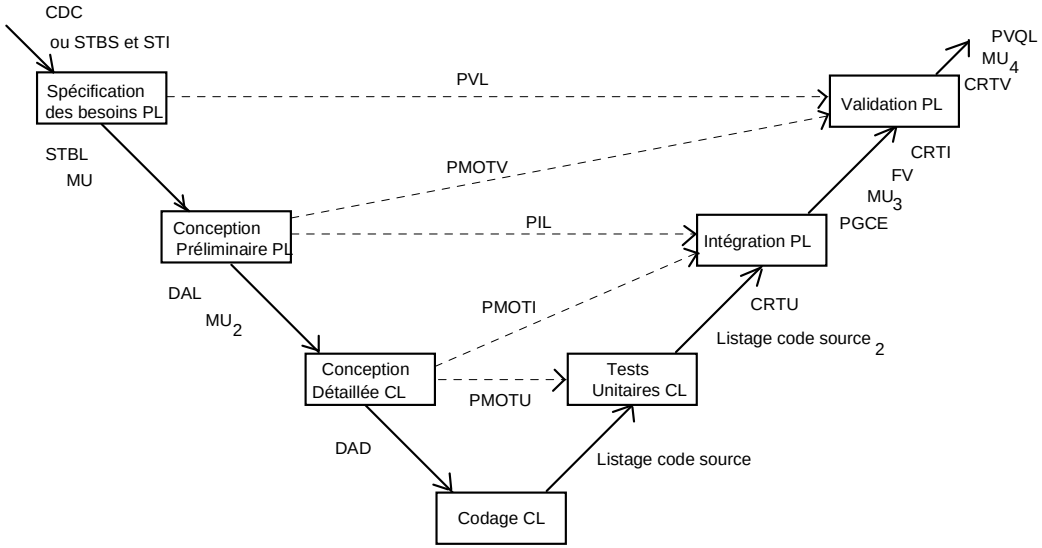
Les modèles de développement : cycle en V

- **Le modèle en V**
  - Amélioration du modèle en cascade
  - Issu des études de Goldberg en 1986
  - Modèle normalisé par l'AFCIQ (Association Française pour le Contrôle Industriel et la Qualité), l'AFNOR et l'ISO
  - Objectif majeur : réduire « l'effet tunnel » , : la perte de visibilité par les clients à un moment donné de l'avancement du projet.
  - Dans chacune des étapes de la branche gauche, on décrit les critères d'appréciation et d'acceptation du système aux étapes correspondantes de la branche droite du V



V : autres implémentations & détail des livrables





CDC = Cahier Des Charges du client dont le CDCF (CDC Fonctionnel)

STBS = Spécification Technique de Besoin du Système

STI = Spécification Technique des Interfaces

STBL = Spécification Technique de Besoin du Logiciel (Produit Logiciel)

MU = Manuel Utilisateur (résultant éventuellement d'une activité de maquettage des IHM)

PVL = Plan de Validation du Logiciel (stratégies, responsabilités, moyens, types de test et types de preuves)

DAL = Document d'Architecture du Logiciel

PIL = Plan d'Intégration du Logiciel

PMOTV = Procédures de Mise en Oeuvre des Tests de Validation (conditions, objectifs, jeux d'essais, résultats attendus)

DAD = Document d'Analyse Détaillée

PMOTI = Procédures de Mise en Oeuvre des Tests d'Intégration

PMOTU = Procédures de Mise en Oeuvre des Tests Unitaires

CRTU = Comptes-Rendus des Tests Unitaires

CRTI = Comptes-Rendus des Tests d'Intégration

PGCE = Procédures de Génération du Code Exécutable

FV = Fiche de Version (identification et description du logiciel)

CRTV = Comptes-Rendus des Tests de Validation

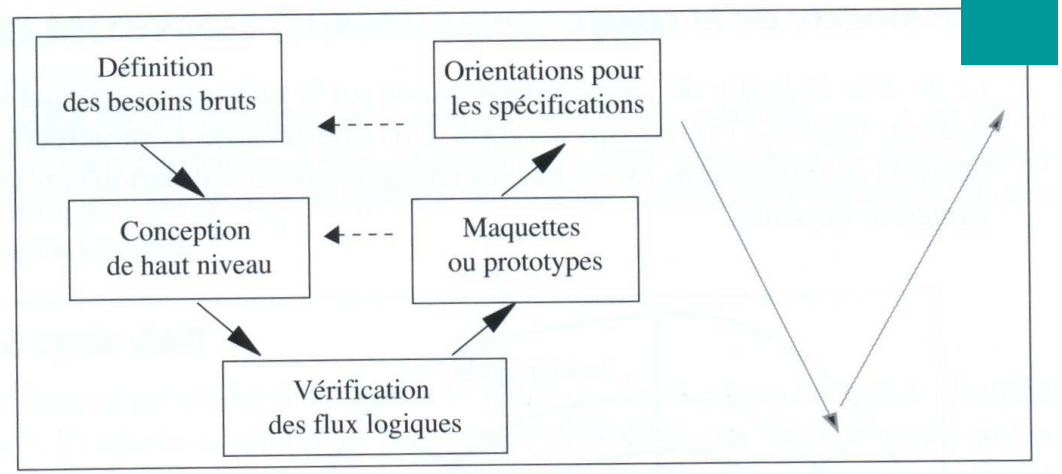
PVQL = Procès Verbal de Qualification du Logiciel (acceptation client)

Les modèles de développement : cycle en W

- **Le modèle en W**

  - Enrichissement du modèle V, dans le même esprit d'anticipation sur le livrable final

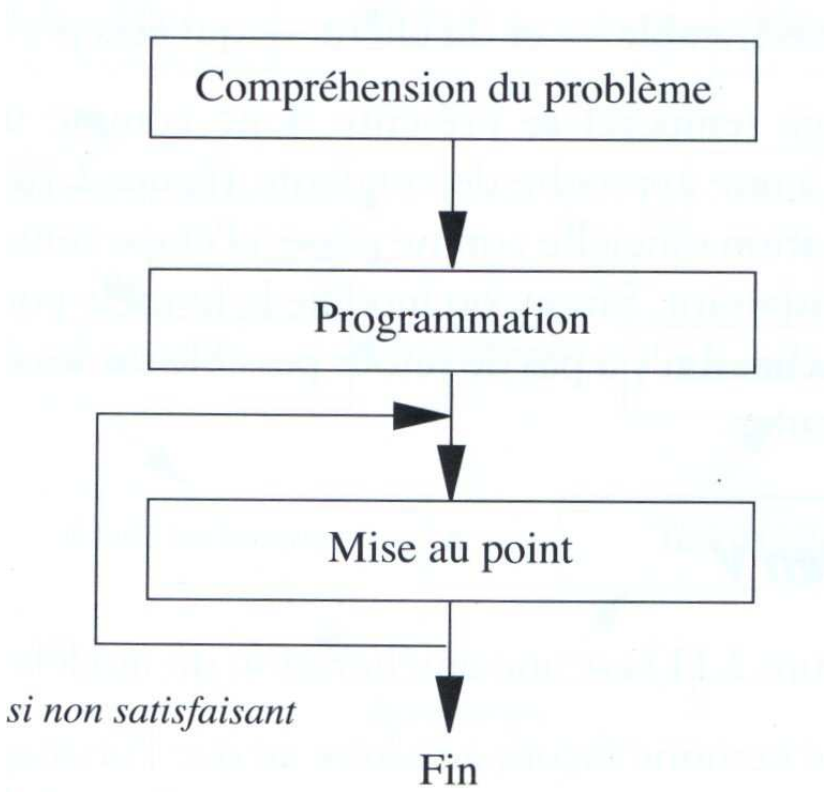
  - La 1ère partie du W vise à dégager avec les utilisateurs des orientations solides pour la conception ou bien à explorer les possibilités d'une nouvelle technique : développement de maquettes ou prototypes permet une validation plus concrète, voire une expérimentation.



  - **Prototype** : logiciel de démonstration possédant certaines des fonctionnalités proposées dans la version finale
  - **Maquette** : simulation de logiciel pour montrer l'allure, les fonctionnalités ne sont pas réalisées, on ne montre que l'interface

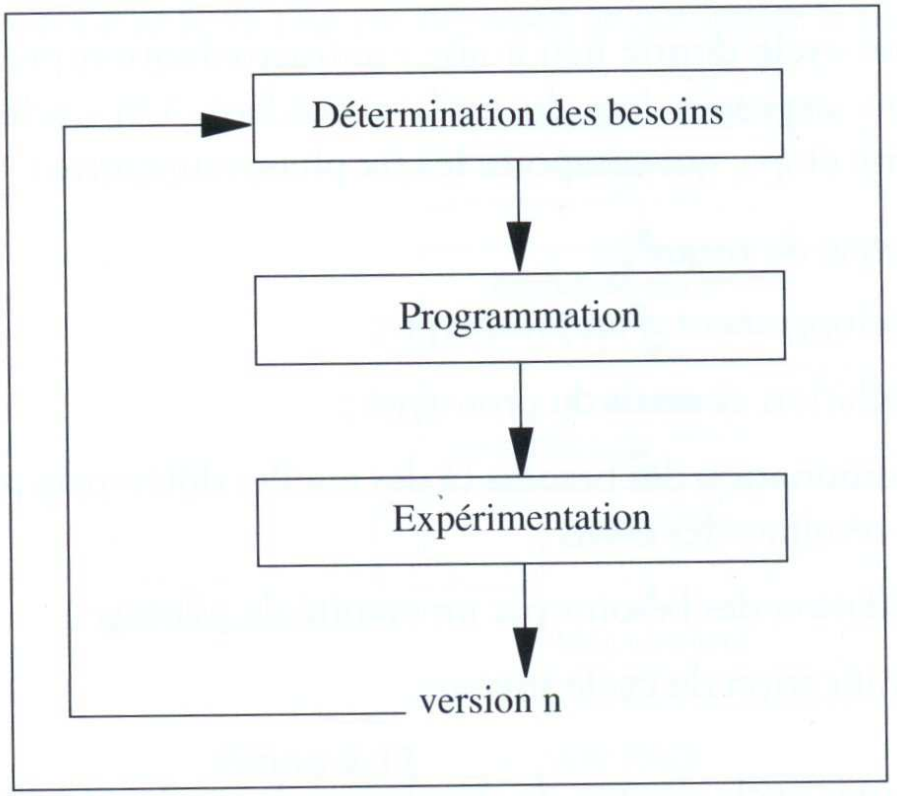
Les modèles de développement : code and fix

- **Le modèle *code-and-fix***
  - Hypothèse projet : détermination facile des besoins.
  - Après une étape brève de compréhension de l'objectif, l'application est développée. Ensuite, plusieurs cycles de mise au point, parfois en collaboration avec l'utilisateur du futur système, permettant d'atteindre le résultat visé.



Les modèles de développement : évolutif

- **Le modèle de développement évolutif *(evolutionnary design model)***
  - Hypothèse : les besoins ne peuvent s'exprimer qu'après une expérimentation
  - Objectif : construction progressive du système de façon participative
  - Chaque cycle aboutit à une nouvelle version du système : on s'arrête lorsque le client juge le système satisfaisant



Les modèles de développement : spirale

- **Le modèle en spirale**

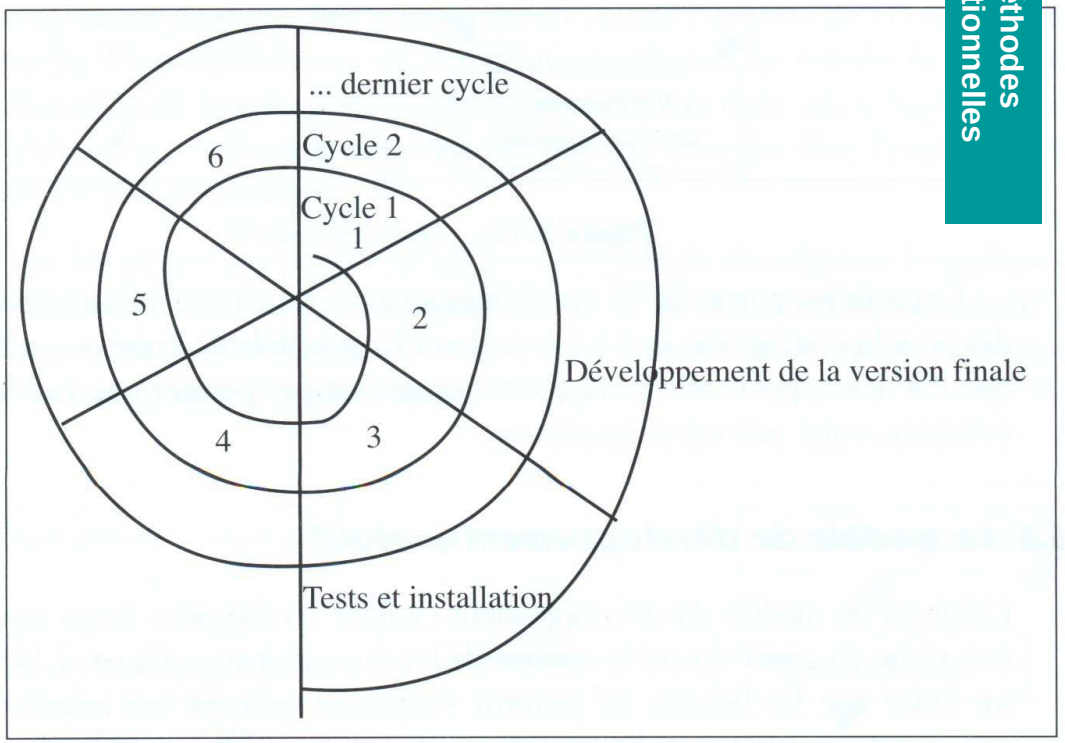
  – Crée par Boehm en 1988

  – Même principe que le modèle évolutif, mais avec une relation contractuelle entre le client et le fournisseur.

  – Les engagements et validation présentent un caractère formalisé : chaque cycle donne lieu à une contractualisation préalable.

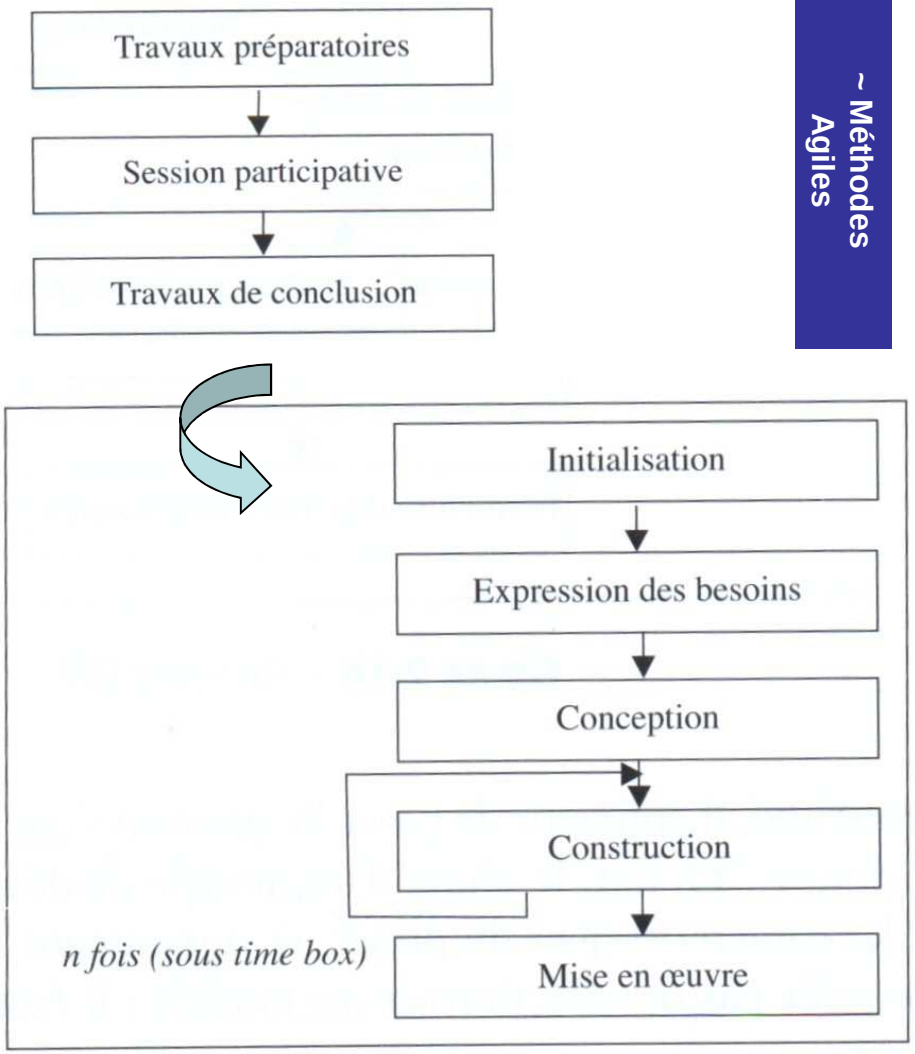
  – Un cycle est composé des étapes suivantes :
    - Analyse du risque
    - Développement d'un prototype
    - Simulation et essais du prototype
    - Détermination des besoins (à des mailles différentes selon le cycle), à partir des résultats des essais.
    - Validation des besoins par un comité de pilotage
    - Planification du cycle suivant.

  – Le dernier cycle permet de développer la version finale et d'implémenter le logiciel.



Les modèles de développement :RAD

- **Le cycle RAD *(Rapid Application Development)***
  - Objectif : Développement rapide dans des délais réduits.
  - Participation organisée et contrôlée des utilisateurs.
  - Découpage temporel introduisant systématiquement une session participative entre l'équipe projet et le le groupe d'utilisateurs au milieu de chaque phase
  - Le cycle global conjugue modèles en cascade et en spirale.
  - L'étape construction se compose d'un nombre variable de cycles de prototypage, mais strictement dans un temps limité (time box) à ne pas dépasser.



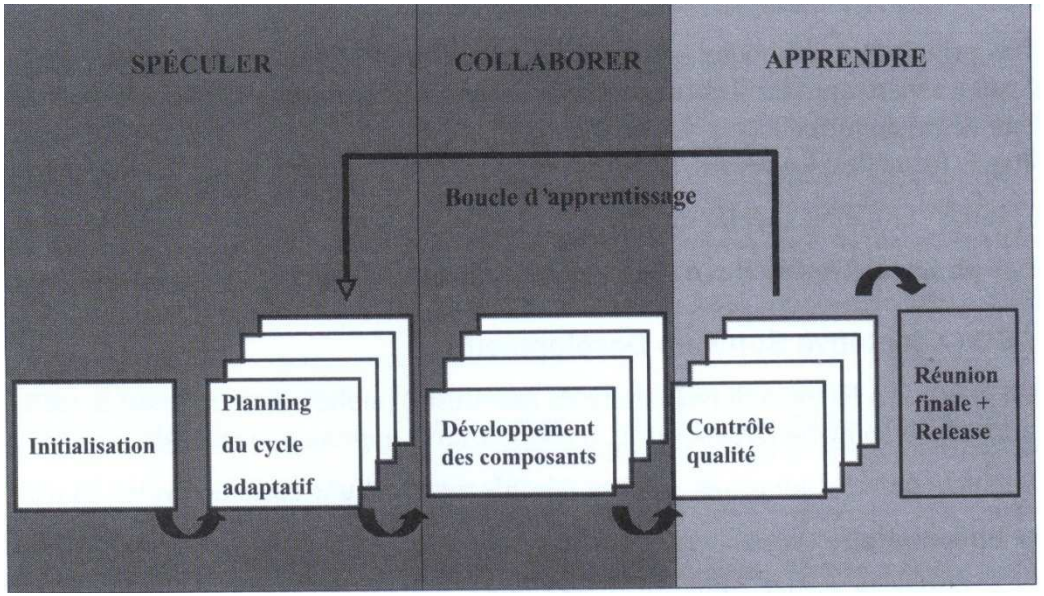
Les modèles de développement : ASD

- **Le cycle ASD *(Adaptative Software Development)***

  – Objectif : Approche collaborative pour gérer les projets complexes

  – Modèle temporel et structurel , proposant une série de cycles en 3 volets :
    - Spéculation : initier le projet, déterminer sa durée, le nombre d'itérations (4 à 8 semaines par itération) , affecter un objectif à chaque itération, dresser la liste des tâches.
    - Collaboration : livraison des composants, communication forte et assez informelle
    - Apprentissage : contrôle qualité, suivi et bilan d'avancement, communication forte et assez informelle

  – Caractéristiques : Focaliser sur l'objectif ; se baser sur des composants; itérer ; *time boxing* ; pilotage par les risques ; accepter le changement.



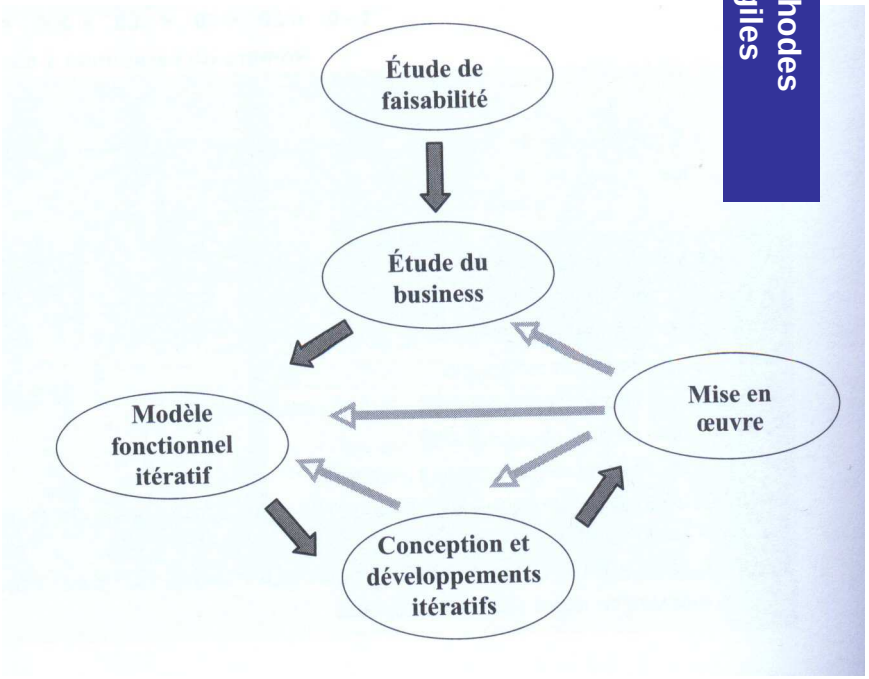
Les modèles de développement : DSDM

- **DSDM *(Dynamic Software Devlopment Method)***

  – Objectif : utilisation de RAD de façon structurée et indépendante (Grande Bretagne)

  – Modèle temporel itératif :
    - Etude de faisabilité : définition du problème, étude faisabilité technique, méthodologique et budgétaire.
    - Etude du business : analyse des processus métier, hiérarchisation des besoins en ateliers avec les utilisateurs ; définition de l'architecture globale et du plan global de prototypage.
    - Modèle fonctionnel itératif : description des besoins en détail, identification des modules logiciels constituant un prototype fonctionnel
    - Conception et développements itératifs : fourniture d'un système conforme aux besoins définis.
    - Mise en œuvre : livraison, prise en main pour test ; bilan.

  – Caractéristiques : implication active des utilisateurs, livraisons fréquentes, développement itératif et incrémental, intégration des tests dans tout le cycle de vie.



Les modèles de développement : Scrum (1/4)

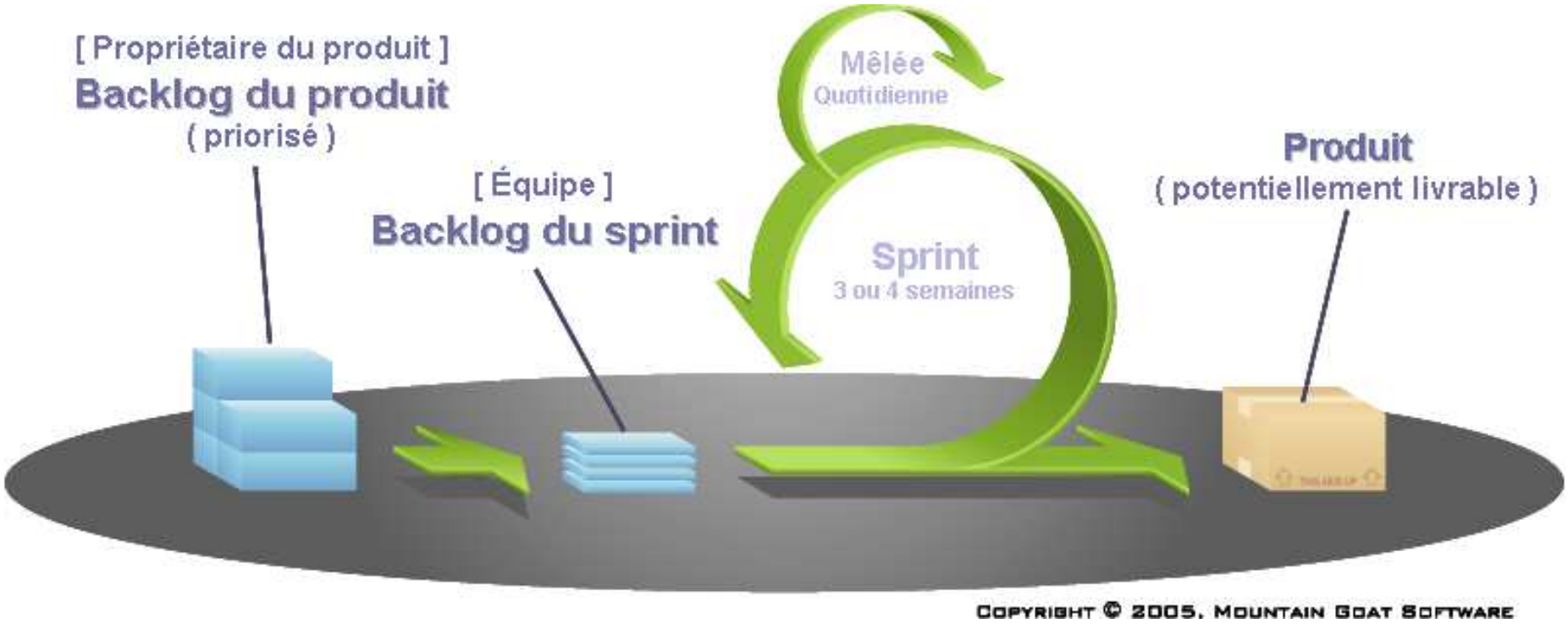
- **Scrum *(mêlée)***

  – Objectif : Travailler en équipes pluridisciplinaires intégrées et soudées dans une stratégie flexible ; principe des petits ajustements fréquents

  – Modèle temporel itératif, reposant sur des sprints de 4 semaines :
    - Sprint planning meeting : sélection des exigences prioritaires pour le client dans le backlog.
    - Réunion d'avancement quotidienne : le scrum, avec toute l'équipe.
    - Sprint Review Meeting : démonstration des derniers développements faite au client
    - Retrospective : revue de fin d'itération, bilan qualitatif sur le fonctionnement de l'équipe.

  – Caractéristiques :
    - Visibilité : une fonctionnalité implémentée est considérée comme terminée lorsque tous les acteurs s'entendent sur les modalités d'évaluation des résultats tangibles.
    - Inspection : vérifier les écarts par rapport à l'objectif initial ; nombreux points de contrôle.
    - Adaptation : en cas de constat d'écarts , ajustements décidés immédiatement.

Scrum : le workflow (2/4)



Scrum : les rôles (3/4)

- Le Product Owner
  - Il représente des clients et des utilisateurs. Son objectif est de maximiser la valeur du produit développé.
  - Il explicite les éléments (***Items***) du Carnet du produit. Ces composantes sont exprimées sous forme d'Histoires Utilisateur (***User Stories***).
  - Il rédige les spécifications et les cas de tests,
  - Il définit l'ordre de priorité des développement, notamment les objectifs de chaque sprint,
  - Il doit avoir le niveau de prise de décision nécessaire et doit être disponible

- Le Scrum Master
  - Il est responsable de la méthode, et rien d'autre !!!
  - Il a un rôle de facilitateur et son objectif est maximiser la valeur produite par l'équipe dans le respect des règles de la méthode Scrum
  - Dans certains cas, il peut être assimilé à un coach.

- Le développeur
  - Tous les autres !
  - Il n'y a pas de rôles prédéfinis puisque l'équipe est « auto-gérée » et autonome,

Scrum : les avantages et inconvénients (4/4)

- Avantages
  - Entièrement développé et testé pour de courtes itérations
  - Simplicité des processus
  - Organisation personnelle
  - Responsabilisation de chacun
  - Cohésion d'équipe
  - Combinaison possible avec XP

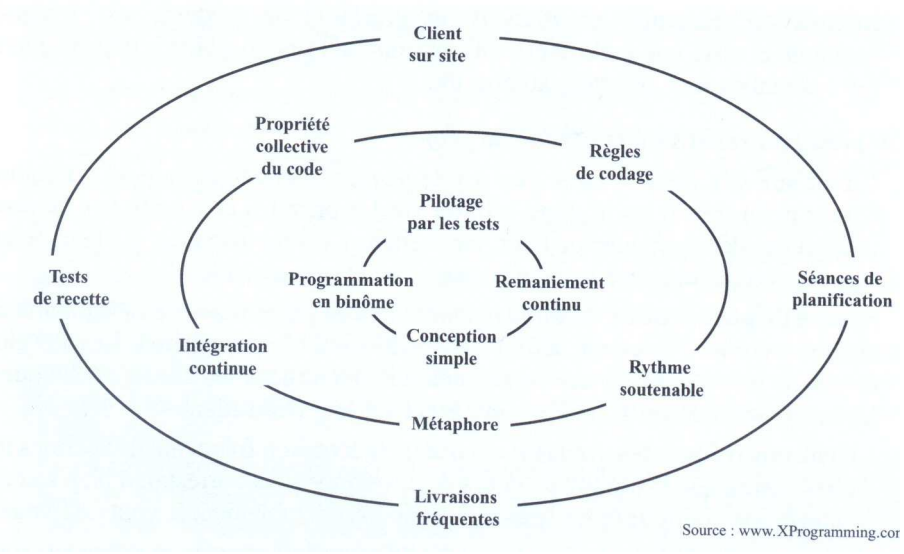
- Inconvénients
  - Peu, voire pas, de documentation écrite
  - Violation de responsabilité (pas de hiérarchie)
  - Pratiques assez nouvelles donc conduite d changement à prévoir
  - Très difficile à mettre en place avec une équipe virtuelle
  - Disponibilité accrue du client

Les modèles de développement : XP

- **XP *(eXtreme Programming)***

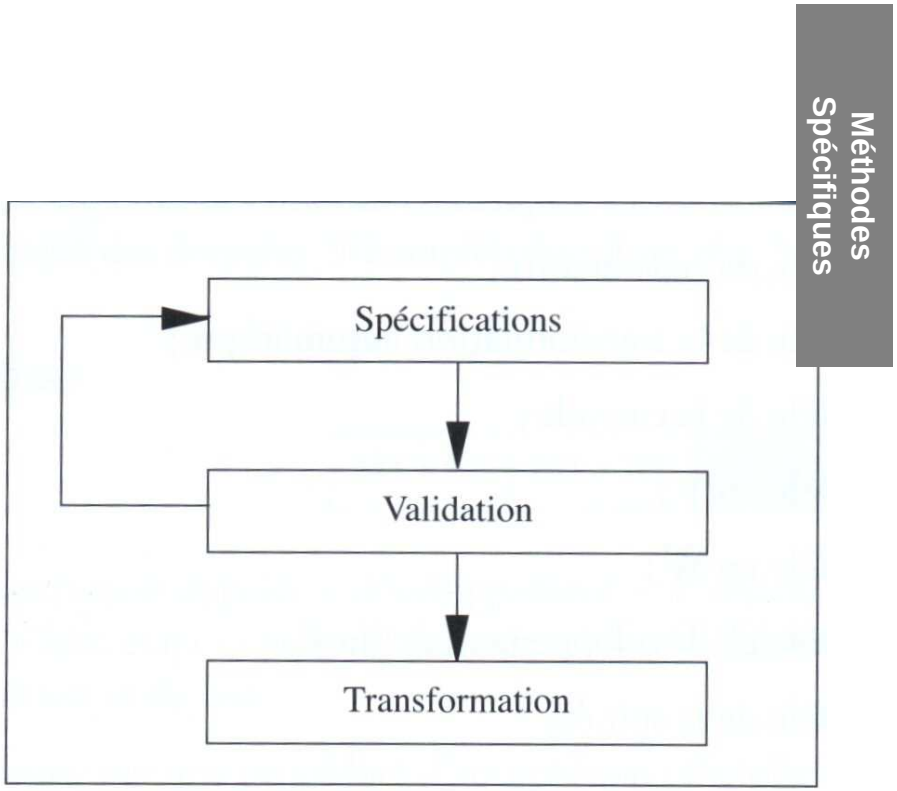
  – Objectif : approche systématique (extrême) des tests, de l'implication des utilisateurs, des livraisons fréquentes. Le développement au cœur du projet.

  – Pratiques de programmation :
    - Conception simple : concision, modularité, lisibilité ; faciliter la maintenance
    - Développement piloté par les tests unitaires
    - Tests de recette
    - Remaniement continu ou refactorisation du code : rendre le code le plus «propre » possible
    - Programmation en binôme
    - Règles de codage
    - Propriété collective du code
    - Métaphore : afin de faciliter échanges entre techniciens et fonctionnels
    - Intégration continue
    - Client sur site
    - Séances de planification
    - Livraisons fréquentes
    - Rythme soutenable par les équipes.



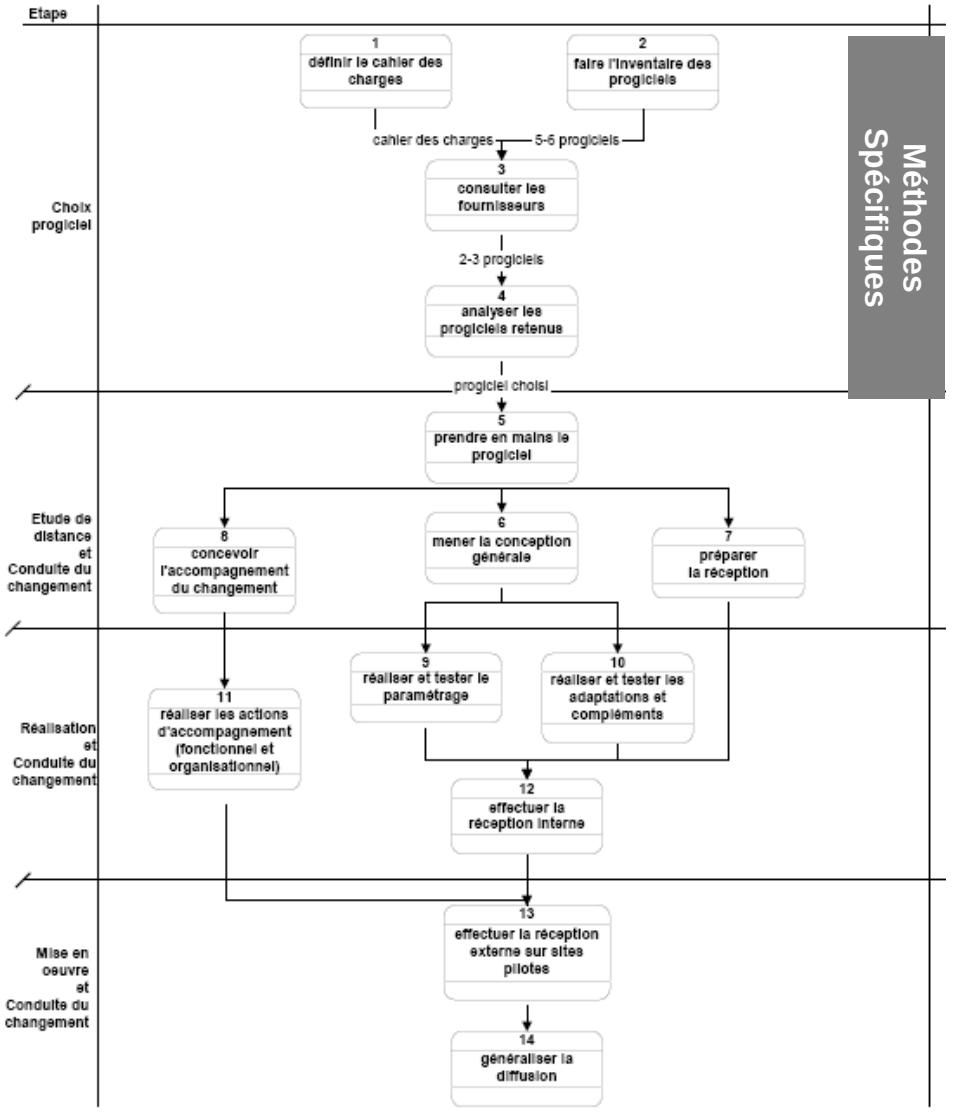
Les modèles de développement : transformation automatique

- **Le modèle de la transformation automatique *(transform model)***
  - Hypothèse projet : possibilité de transformer automatiquement des spécifications en programmes, à l'aide de règles.
  - L'essentiel de l'effort va donc porter sur une description exhaustive des spécifications, qui devront être complètement validées.
  - Une succession de cycles de spécification/validation s'achève par la génération de code.



Les modèles de développement :ERP

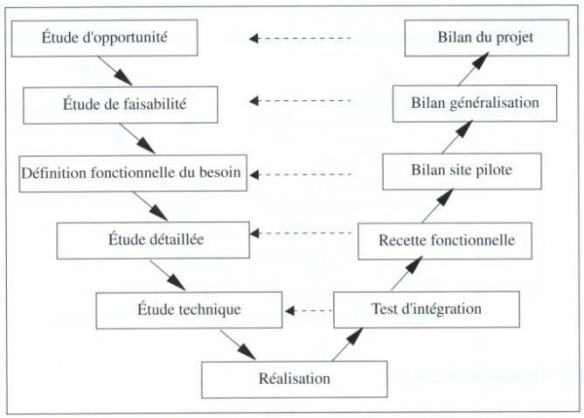
- **Les cycles ERP (*Entreprise Resource Planning* ou Progiciel de Gestion Intégré)**
  - Objectif : construire un système améliorant la performance de l'entreprise en tirant le meilleur parti d'un progiciel
  - Nécessité d'une phase de description / cartographie des processus (normalement en amont du projet , mais souvent pendant le projet)
  - La conception générale peut se faire par étude différentielle vis-à-vis des fonctionnalités de l'ERP
  - Cycles itératifs d'analyse –paramétrage – prototypage par processus.



Les modèles de développement au quotidien

- **Des modèles spécifiques dans chaque organisation**
  - Adaptation des modèles courant aux spécificités de l'organisation
  - Emploi d'un vocabulaire « maison » propre : retrouver les concepts sous-jacents
  - Beaucoup de cycles de vie sont le fruit du mélange de modèles standards
  - Les modèles les mieux adaptés à un projet donné tiennent compte :
    - De la taille et de la durée du projet
    - De la maturité du besoin exprimé
    - De l'appel ou non à la sous-traitance
    - De la complexité technique du projet
    - De l'utilisation ou non d'un progiciel
    - De la culture (latine, anglo-saxone, etc) du projet
    - Du caractère international (multi-culturel) du projet
    - De l'expérience et du nombre de projets annuellement réalisés par l'organisation
    - Du niveau de maturité (éventuelles certifications, ex: CMMI niv3) des processus d'ingénierie de l'entreprise.
    - Des techniques / outils de développement utilisés en réalisation

- Etude de cas : France Telecom , ANPE



D’un cycle à l’autre …

Des principes communs

Exprimer un **besoin**

Définir un périmètre

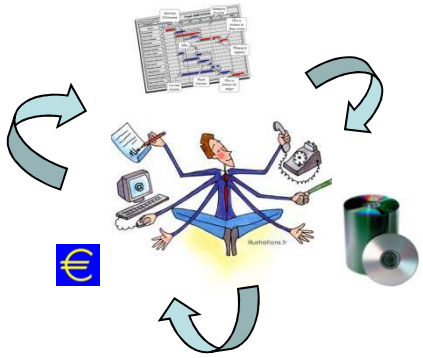
Concevoir une solution

Construire la solution

Tester

Installer

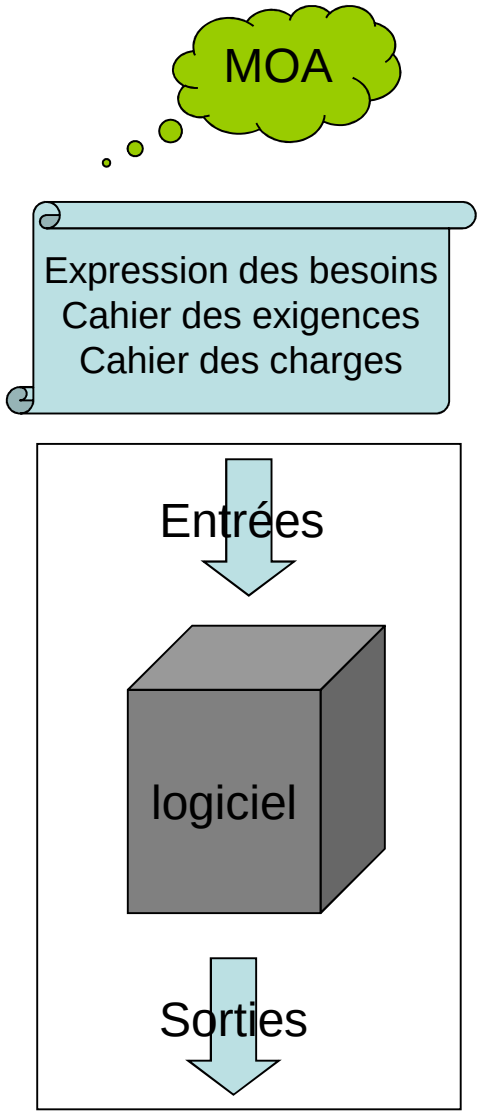
Surveiller



DAUPHINE UNIVERSITÉ PARIS

L'expression des besoins et la définition du périmètre

- Se reporter au chapitre II – Finalité et périmètre d'un projet
  - Notion d'exigences et de contenu
- Le besoin
  - Est extérieur au produit
  - Est du domaine de l'utilisateur
- Les caractéristiques fonctionnelles
  - Sont du domaine du produit
  - Une fonction est l'action d'un produit (ou d'un constituant du produit) exprimer en terme de finalité
  - La finalité est la satisfaction du besoin ou le respect d'une contrainte à remplir
- L'analyse fonctionnelle externe concerne
  - Tous les aspects relatifs à l'environnement du produit
  - Ce qui contraint le fonctionnement
- L'analyse fonctionnelle interne concerne
  - Les relations fonctionnelles entre les diverses parties internes du produit.
- A l'issue de cette phase on considère la solution (ex : le logiciel) comme une boîte noire dont on ne connaît que les entrées et les sorties.
- Livrable en sortie :
  - le cahier des charges

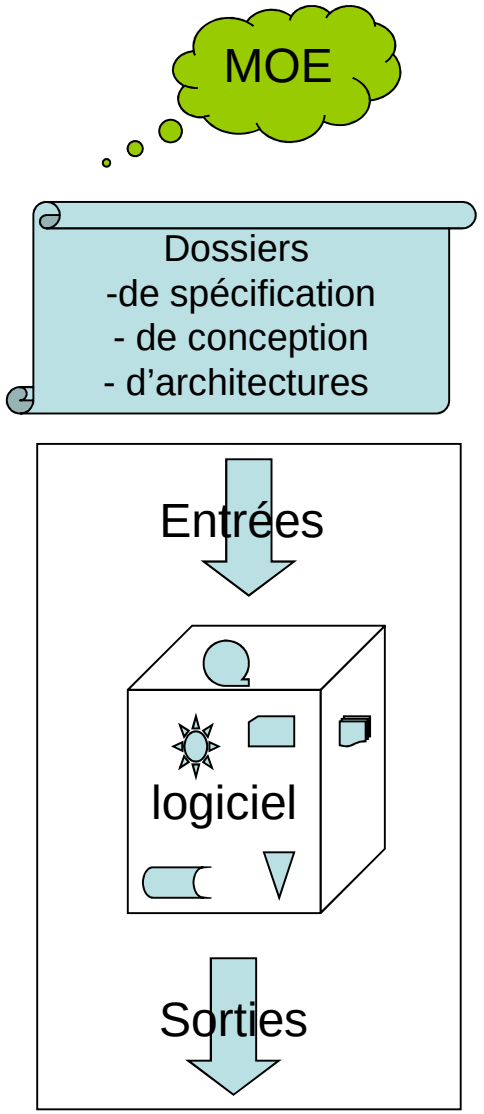


L’analyse des besoins et conception

- Objectif
  - Définir de façon très précise les fonctions et services rendus et l’architecture du logiciel
  - Démontrer (tracer) la couverture des exigences, des besoins par les éléments conçus.

- Activités
  - Les choix techniques sont effectués
  - Spécification des fonctions et des services
  - Regroupement en modules des éléments à produire
  - Identification des algorithmes à mettre en oeuvre
  - Définition de l’enchaînement des fonctions/services et les éléments déclencheurs

- Livrables :
  - dossiers de conception/spécification générale et détaillée, dossiers d’architecture
    - Fonctionnelle
    - Technique
    - Les spécification d’interface, d’IHM, de traitement, d’organisation des données, etc

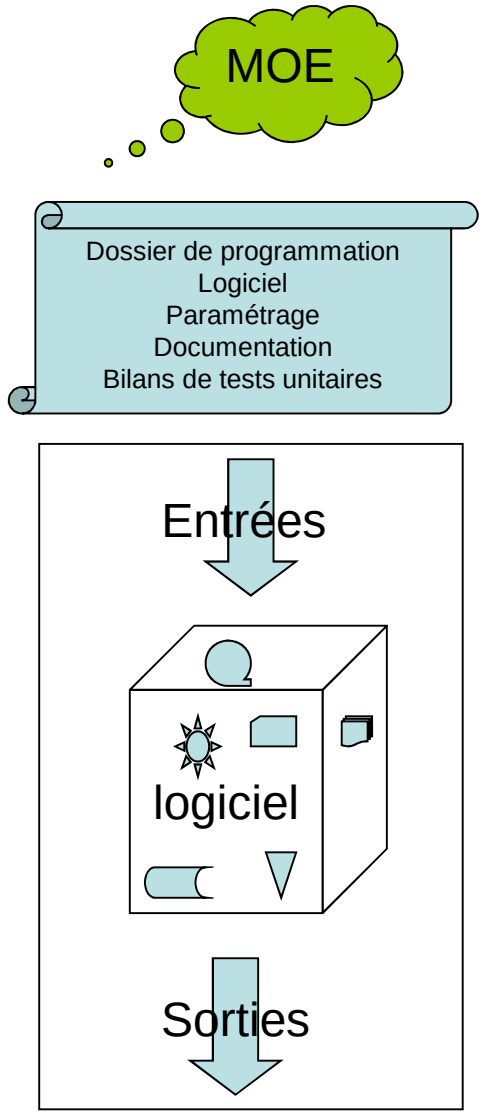


Réalisation – développement – construction : Programmation, paramétrage et tests unitaires

- Objectif
  - Développer, modifier et paramétrer le logiciel
  - Procéder pour chaque module/composant aux tests unitaires

- Activités
  - Etude algorithmique
  - Codage du logiciel
  - Relecture croisée du code
  - Paramétrage
  - Tests unitaires

- Livrables :
  - Logiciel :
    - Code source
    - Procédure de génération (compilation, …)
    - Logiciel installable et exécutable
    - Kit d'installation
  - Fiches de relecture de code
  - Bilan des tests unitaires
  - Paramétrage

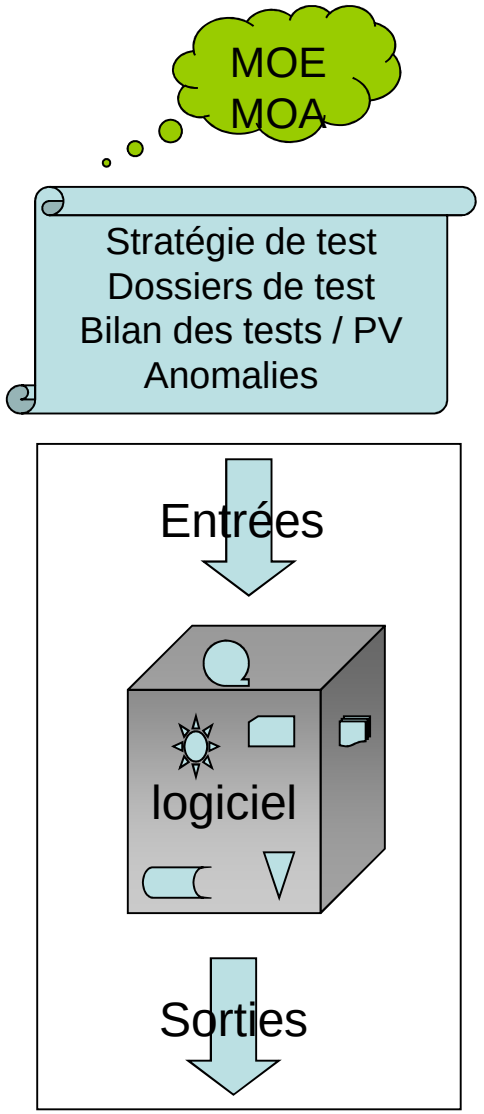


Qualification – Tests – Intégration – Validation - VABF

- Objectif
  - S'assurer que le produit livré est adapté à son usage de destination et satisfait les exigences , les besoins vérifiables.

- Activités
  - Elaborer une stratégie de test
  - Rédiger des dossiers de test
  - Préparer les jeux de données de test
  - Installer le logiciel, le paamétrage
  - Exécuter les tests et signaler les écarts (anomalies, écarts par rapport aux besoins)
  - Tracer les résultats obtenus
  - Faire corriger les anomalies et faire livrer les correctifs

- Livrables :
  - Stratégie de test
  - Dossiers / cahiers de test
    - Scénario
    - Fiches de test
    - Description des jeux de données
  - Bilan des tests
  - PV
  - Anomalies

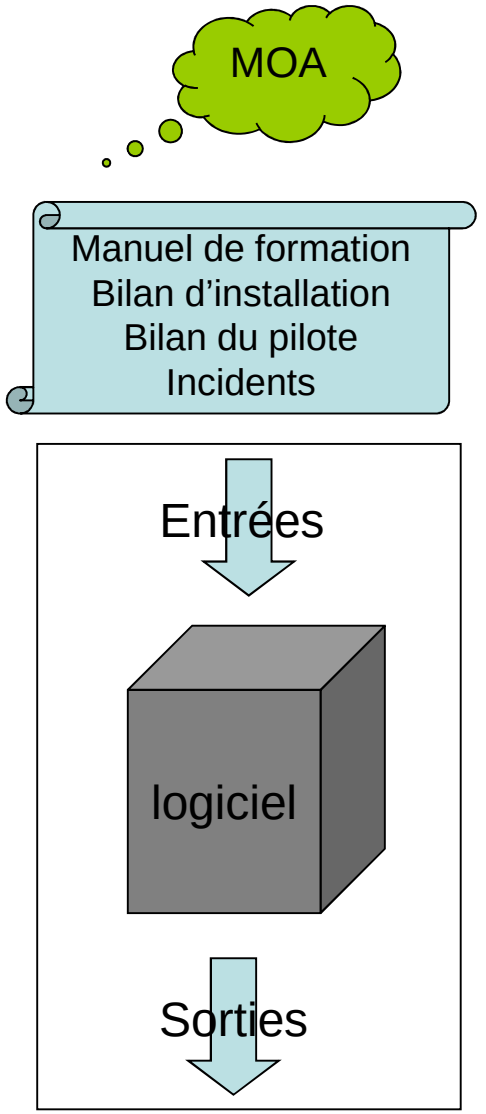


Mise en production – Site pilote

- Objectif
  - Installer le logiciel en production (sur environnement technique, matériel et logiciel d'infrastructure cible)
  - Vérifier que l'utilisation de l'applicatif répond aux exigences définies et aux besoins utilisateurs et que l'organisation pressentie pour le déploiement généralisé est opérationnelle

- Activités
  - Préparation des sites pilotes
  - Installation du logiciel paramétrer
  - Planification des activités pilotes
  - Déroulement de la phase pilote
  - Formation des utilisateurs
  - Correction des anomalies et livraison des correctifs

- Livrables :
  - Documents de formation utilisateur
  - Bilan d'installation
  - Compte-rendu / bilan du pilote
  - Remontée d'incidents / anomalies
  - Feu vert pour généralisation



Généralisation – Vérifications de Service Régulier

- Objectif
  - Utiliser en mode nominal, par toute la population d'utilisateurs concernée, l'ensemble des fonctions du logiciels
  - Surveiller pendant une période définie l'utilisation normale de la solution

- Activités
  - Compléter les installations ou ouvrir les droits d'accès à tous les utilisateurs
  - Surveiller le service régulier
  - Faire corriger les anomalies et faire livrer les correctifs

- Livrables :
  - Compte-rendu / bilan de la période
  - Remontée d'incidents / anomalies
  - Feu vert pour fin de surveillance spécifique.