

Conduite de projet

Qu'est-ce qu'un projet ? : Réaliser quelque chose (le produit) pour faire quelque chose (le besoin) porté par les clients / les utilisateurs

1 Objectifs

Objectifs de l'entreprise	Objectifs humains
Economiques : résultat financier (marge, bénéfice) Image de marque, référence Acquisition technique, R&D	Travail intéressant/motivant pour les membres de l'équipe Formation, gain d'expérience, etc.

1.1 Succès d'un projet ?

Satisfaction du besoin (réel) du client (utilisateur)	Satisfaction de l'entreprise	Satisfaction de l'équipe projet
Qualité / complétude du produit réalisé : qu'il remplisse sa fonction Conformité au cahier des charges : qu'il réponde au besoin Livraison dans les délais : qu'il soit prêt à temps	Bénéfice financier (respecter le budget) Bonne image chez le client (bon pour avoir d'autres contrats) Capitalisation de l'expérience acquise (pour s'améliorer)	Bonne ambiance, objectifs communs Progression, objectifs personnels

1.2 Les moyens pour y arriver

- Comprendre le périmètre du projet
- Définir des objectifs clairs
- Quantifier/mesurer les risques et trouver des solutions palliatives !
- Assurer le suivi et le contrôle (méthode cohérente de traçabilité, gestion optimale des demandes de modification)
- Assurer une communication efficace auprès du client et de l'équipe projet

1.3 Maîtriser les délais

- Comment raccourcir un délai ? :
 - Augmenter la taille de l'équipe ... augmente aussi le coût
 - Travailler la nuit aussi ... pas trop longtemps !
 - Faire au plus vite « quick and dirty » ... attention à la qualité du résultat !
 - D'abord éviter de l'allonger (travail inutile, ou à refaire, etc.)
 - Proposer une modification du contenu (par exemple en reportant certaines fonctions à une version ultérieure)
- Prévoir les délais : planifier le projet
 - Délais d'approvisionnement
 - Délais de prise de décisions ... par le client ou par la hiérarchie
 - Minimiser les inter-dépendances entre travaux, autant que possible

Un projet démarré sans marge de planning est déjà en danger ...

1.4 Maîtriser le coût de production

- Plus une erreur est détectée tard, plus son coût est élevé
- Trouver des solutions faciles à réaliser : en chercher plusieurs
- Identifier les surcoûts le plus tôt possible (ou risques de surcoûts)

- S'assurer des marges de budget (provisions)
- Ne pas faire des choses inutiles
 - Fonctions pas nécessaires aux utilisateurs (sur-spécification)
 - Ne pas faire les mêmes choses plusieurs fois
 - Eviter de devoir recommencer des développements

Attention, une économie faite à un certain moment ne signifie pas forcément une économie à la fin du projet.
Exemple : espace disque prévu trop petit, d'où blocages, énergie dépensée à trouver des solutions « bouts de chandelles »

1.5 Conclusion

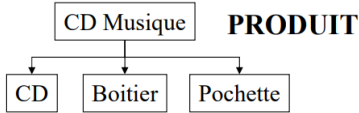
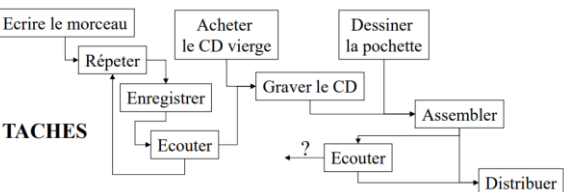
La conduite de projet n'est pas une science exacte mais l'utilisation de méthodes est très utile (profiter de l'expérience des prédécesseurs) mais cela n'exclut pas la créativité (chaque projet est spécifique)

Quelques qualités à entretenir : Bon sens et pragmatisme, communication (à tous les niveaux), sens de l'organisation, anticipation

À tout moment bien connaître ses objectifs et ses priorités (équilibre qualité / coût / délai)

2 Structuration d'un projet

But : Structurer / décomposer le travail à faire

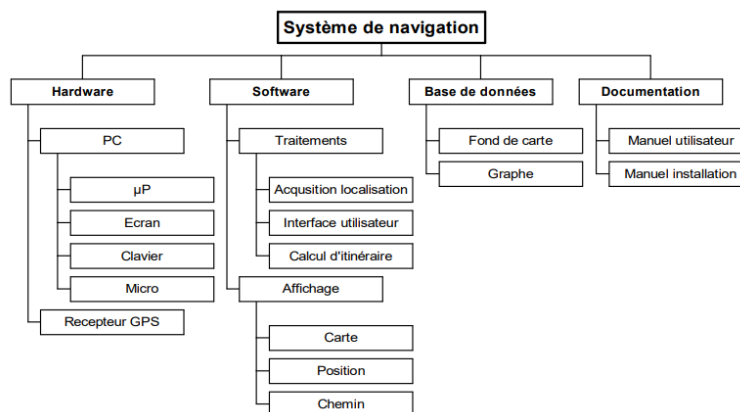
Arbre produit / Product tree / Product Breakdown Structure	Organigramme des Tâches / Work Breakdown Structure
Décomposition hiérarchique en éléments à produire Correspondance avec l'architecture logicielle / matérielle et le prix de vente	Décomposition hiérarchique en tâches à réaliser Correspondance avec la planification, l'organisation du projet et le coût de production
	

2.1 Arbre produit / Product tree

- Décomposition par niveaux successifs du produit final en produits intermédiaires
- Feuille de l'arbre = élément à produire
 - Composants matériels (CPU, disque, etc.)
 - Composants logiciels (application, base de données, etc.)
 - Documents (manuel utilisateur, etc.)
 - Moyens de développement (outils de test, debugger, etc.)
- Il faut savoir s'arrêter à un certain niveau de détail -> Inutile de décomposer des éléments étant construits ensemble ou des éléments approvisionnés (achetés à l'extérieur)

Exemple : Système de navigation pour véhicule dont les fonctions attendues sont :

- Visualiser une carte routière
- Localiser le véhicule sur la carte
- Indiquer le chemin à suivre pour atteindre un but donné



2.2 Organigramme des tâches (OT) Work Breakdown Structure (WBS)

- Décomposition hiérarchique en tâches à effectuer
- Feuille de l'arbre = tâche élémentaire (lot de travaux) : Spécification, codage, tests, rédaction de docs, management, etc.
- Règles de base : Contient toutes les tâches liées au projet, peut être affiné progressivement, éléments de base du planning et de la structure de coût
- Démarche générale
 - 1. Définir l'arbre produit
 - 2. Identifier les tâches à effectuer pour aboutir à chaque produit
 - 3. Regrouper ces tâches en lots de travaux

2.2.1 Lot de travaux (LT) Work Package (WP)

Un WP est affecté à un responsable unique

Est défini par :

- Produit des fournitures qui traduisent la fin de la tâche
- Nécessite des entrées (externes ou fournitures d'autres tâches)
- Tâches de même nature
- Ressources nécessaires à son accomplissement

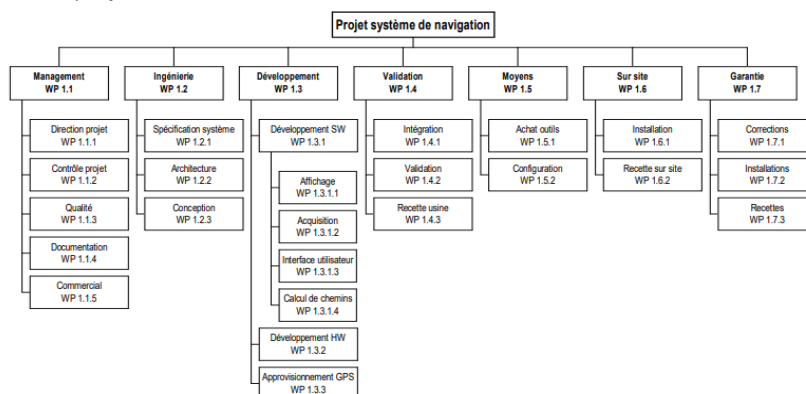
Est caractérisé par :

- Un planning : date de début et date de fin
- Un budget : estimation de coût (devis)

Un WP peut être confié à quelqu'un (sous-projet, sous-traitance)

Exemple : Système de navigation pour véhicule dont les fonctions attendues sont :

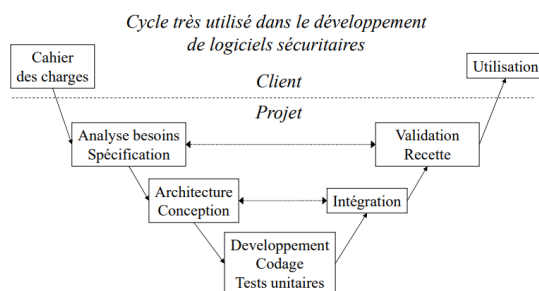
- Visualiser une carte routière
- Localiser le véhicule sur la carte
- Indiquer le chemin à suivre pour atteindre un but donné



3 Cycles de développement

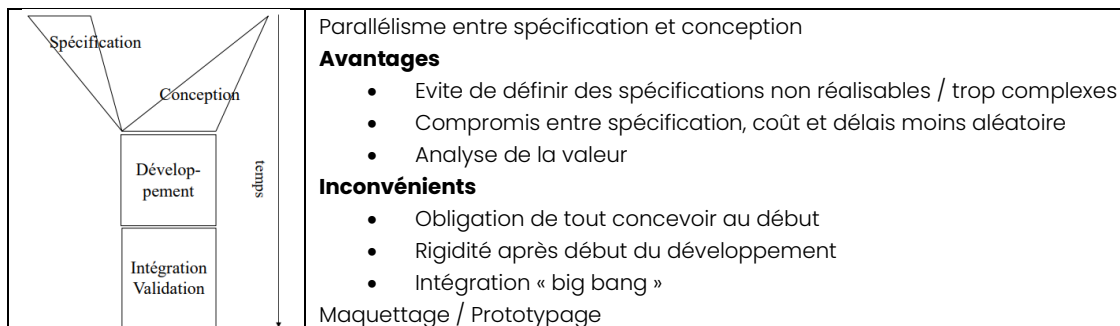
- Les grandes étapes d'un projet
 - Phase A – **Avant-projet** (Que nous faut-il ?) -> **Besoin**
 - Phase B – **Définition** (Que faut-il fabriquer ?) -> **Spécification**
 - Phase C – **Conception** (Comment le fabriquer ?) -> **Architecture**
 - Phase D – **Développement** (Fabrication proprement dite) -> **Produit**
- Phases de développement standard
- Les différents cycles de vie logiciel : Cycle en V, Maquettage/prototypage, Incrémental, Itératif, Méthodes « Agiles »

3.1 Cycle de vie en V



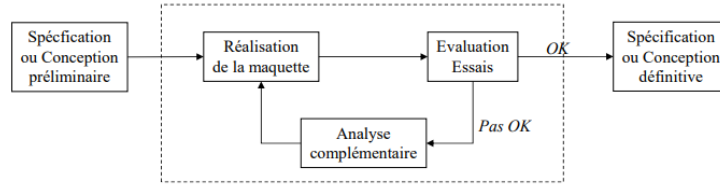
Avantages	Inconvénients
<ul style="list-style-type: none"> Structuration par rapport à l'approche codage direct Discipline le processus de développement Oblige à terminer une étape avant de commencer la suivante (revues) Evite de laisser certaines choses « pour plus tard » ... et de les oublier ! 	<ul style="list-style-type: none"> Illusion de réussir chaque étape du premier coup Risques : spécification / coût / délais \diamond Compromis « à l'aveuglette » Réussir une « bonne » spécification/conception est difficile, il faut quelquefois « se donner le temps » Des évolutions sont inévitables au cours du projet, comment les gérer ? Obligation de tout concevoir en détail avant de commencer le développement On intègre tous ensemble à la fin : ça passe ou ça casse ...

3.2 Cycle de vie en Y



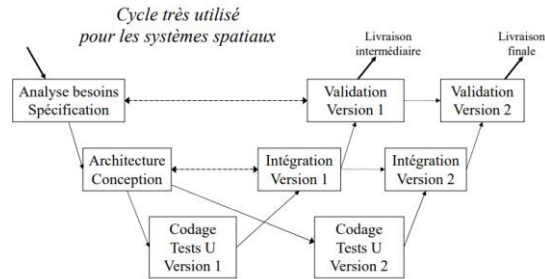
Maquettage / Prototypage

- Pour quoi faire ? : Pour approfondir / valider les besoins / spécifications ou pour valider un choix de conception (faisabilité)
- Comment ? : Ne prend pas en compte toutes les contraintes (fonctionnelle / archi / perfs)



- Maquette = réalisation rapide, non réutilisée par la suite
- Prototype = implémentation partielle, base pour construire le produit final

3.3 Cycle de vie incrémental



Principe : Développer le système ou le logiciel en plusieurs versions intermédiaires successives de plus en plus complètes

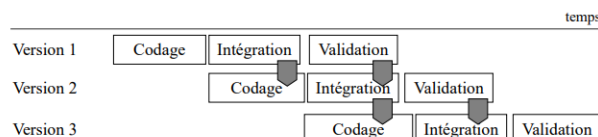
Versions intermédiaires • Livraisons intermédiaires vers le client – A identifier dès le début du projet – Souvent demandées par le client pour les gros systèmes • Concrétise le futur système plus tôt • Permet aux futurs utilisateurs de le manipuler plus tôt • Revient à une sorte de prototypage grandeur nature – Peuvent aider à résoudre un problème de délai • Le client ne veut pas toujours « tout, tout de suite » ! • Intégration & validation incrémentales – Les première versions permettent de mettre au point les outils de tests– Versions intermédiaires demandées aux fournisseurs (évite « effet tunnel »)

Avantages

- Permet de donner plus de temps à la conception détaillée des composants les moins bien connus au départ
- Permet de s'attaquer rapidement aux composants critiques sans attendre de définir dans le détail les composants accessoires
- Première version incluant les mécanismes de base du système (maquette d'architecture = interfaces internes seulement)
 - Permet d'en valider le fonctionnement plus tôt
 - Permet de les valider plus longuement, meilleure confiance
 - Peut servir de base pour l'ajout des autres fonctions
- Etale les développements et fractionne la validation

Inconvénients

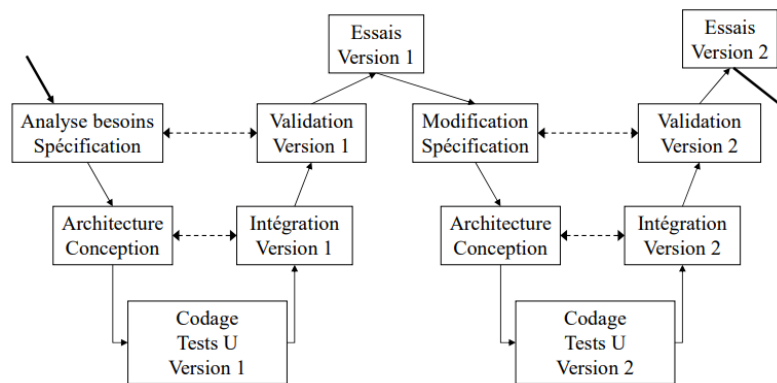
- Coût plus élevé – Tests de régression des fonctions déjà validées sur les versions suivantes– Plus de revues, d'installations, de recettes – Nécessité de gérer plusieurs versions en parallèle



- Remise en cause de l'architecture possible par une version évoluée
- Toujours une conception linéaire en début de projet (spécification)

3.4 Cycle itératif

- Pas toujours possible de faire une spécification complète, stable et satisfaisant le besoin réel au début du projet !
 - Le client ne sait pas bien ce qu'il veut
 - Priorité au délai plus qu'à la spécification
 - Les premières versions donnent des idées de solutions meilleures
 - Problème délicat de faisabilité à résoudre d'abord
- Idée : Itérations successives de cycles complets Spécification / Conception / Développement / Validation
 - Le besoin se précise sur la base des versions successives : démonstrations, discussions, brainstorming
 - La conception elle aussi peut s'affiner et s'améliorer



- Relation contractuelle « non classique »
 - Difficile de s'engager sur un prix au départ
 - Le cycle s'arrête quand le client est satisfait, au bout d'un temps fixé ou au bout d'un budget donné, possibilité de « refaire un cycle »
 - Fondé sur une relation de confiance
- Suivant la taille du projet
 - Bien adapté aux projets de petite taille (logiciels de bureau)
 - Jamais vu sur des gros systèmes opérationnels critiques
- Exemples
 - Rapid Application Development (RAD)
 - Extreme programming

3.5 Méthodes « Agiles »

Méthode de développement flexibles : RAD, Extreme, SCRUM, etc.

- Démarche
 - Spécification grossière en entrée (p. ex. « logiciel de gestion de stock »)
 - Développement (rapide) d'un premier prototype
 - Cycle itératif court (mensuel, hebdo, ...) :
 - Démonstration au client / aux utilisateurs
 - Commentaires, critiques, idées → nouvelle spécification
 - Implémentation des modifications
 - Domaines d'application
 - Besoin grossier, mais besoin urgent

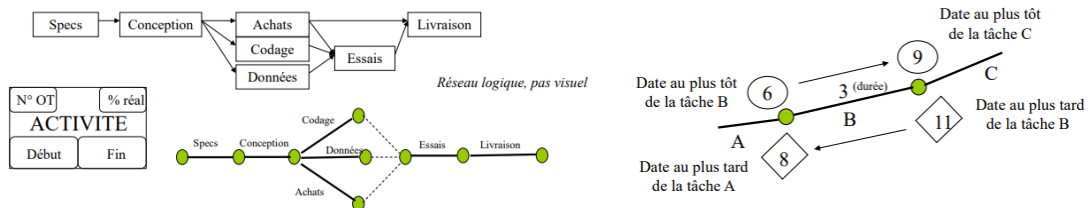
- Logiciel simple (pas système distribué sinon modifications trop coûteuses)
- Phase de maquettage

4 Planification

- Qu'est-ce que le planning d'un projet ?
 - Il définit l'ordre et les dates d'exécution des tâches
 - C'est un vecteur majeur de l'organisation entre les membres de l'équipe projet, de la communication vers le projet / l'entreprise / le client et le Suivi & Gestion du projet
 - Il doit prendre en compte l'organigramme des tâches (OT / WBS), la logique de déroulement du projet, les ressources disponibles et les contraintes du projet.
- Un planning est composé de
 - Activités = tâches de l'OT qui ont une durée et nécessitent des ressources pour être réalisées
 - Événements = livraisons, réunions, revues qui n'ont pas de durée (date ponctuelle) et n'ont pas besoin de ressources
 - Ressources : personnel ou moyens (machines, instruments, données, etc.)
 - Liens temporels entre activités et événements

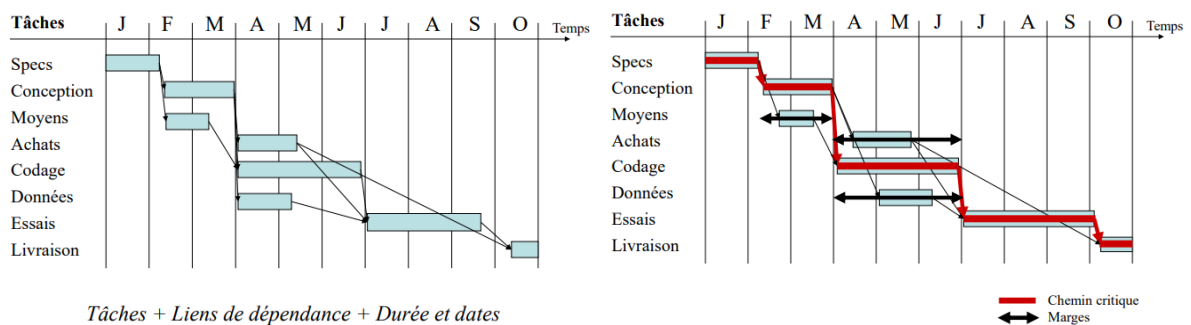
4.1 Réseau PERT

- Réseau logique d'enchaînement des tâches
- Tâches + liens de séquence entre elles
- Indication des informations de planning (dates, durées)
- Permet le calcul/évaluations de propriétés du planning



- Date au plus tôt : la tâche ne peut pas commencer avant cette date
- Date au plus tard : la tâche doit être achevée à cette date

4.2 Diagramme de GANTT



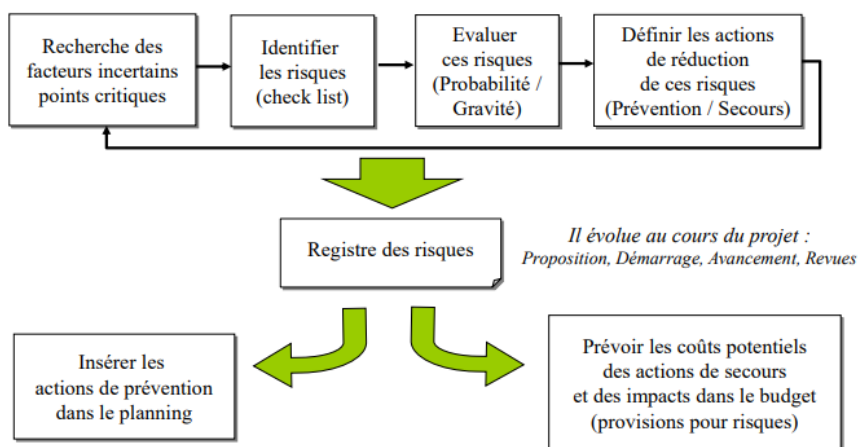
- **Marge globale** du projet : Possibilité de retard de la date de fin prévue sans problème de livraison. Fin contractuelle (promise) – Fin au plus tôt (prévue)
- **Marge brute** (ou globale) d'une activité : Possibilité de retard de sa date de fin sans problème de livraison
- **Marge propre** d'une activité : Possibilité de retard de sa date de fin sans retarder la fin du projet. C'est une marge au départ et/ou dans la durée

- **Tâche critique** : Activité dont la marge brute est égale à la marge globale du projet. Tout retard impacte directement la fin du projet
- **Chemin critique** – Chemin allant du début à la fin du projet, composé uniquement de tâches critiques

4.3 Conseil planning

- Conserver une marge globale
- Pas plus d'un chemin critique
- Minimiser les interdépendances entre tâches : en particulier les dépendances à cause des ressources
- Associer les marges possibles aux tâches les plus risquées
- Anticiper autant que possible
- Faire les tâches risquées au plus tôt

5 Analyse des risques

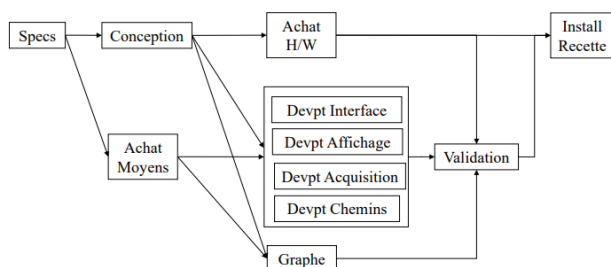


Caractéristiques d'un risque

- Description : événement pouvant arriver et gênant les objectifs du projet
- Causes : causes directes, ou plus profondes
- Date/Jalon d'occurrence : quand le risque peut arriver
- Effets : impacts sur les objectifs (délais, coût, qualité, etc.)
- Probabilité d'occurrence (Haute, Moyenne, Basse)
- Gravité des effets (Haute, Moyenne, Basse)
- Criticité du risque

Gravité \ Proba	Proba		
	H	M	B
H	H	H	H
M	H	H	M
B	H	M	B

- **Actions préventives** : Ce que l'on peut faire avant pour réduire la probabilité d'occurrence / Ce que l'on peut faire avant pour réduire la gravité des effets
- **Actions correctives / de secours** : Ce que l'on peut faire après pour réduire la gravité des effets
- Si la **criticité** du risque est **Haute** (H) -> On intègre les actions préventives dans le plan de développement
- Si la **criticité** du risque est **Moyenne** (M) -> On intègre le coût d'actions correctives dans les provisions pour risque (marges de manœuvre)



Définition du risque	Proba	Gravité	Causes	Effets	Actions (P) Préventives (C) Correctives

6 Suivi

Le chef de projet doit connaître l'état de réalisation de son projet : Etat d'exécution de toutes les tâches du projet, besoin d'une information (synthétique, objective, quantitative, fiable, facile à obtenir)

Indicateurs d'avancement → « **Tableau de bord** »

Avancement physique = pourcentage des travaux réalisés par rapport aux travaux prévus

Exemples : 3 unités livrées sur 4, Spécifications faites à 50%, 5 fenêtres d'IHM réalisées sur 7, Tests réussis à 20%, etc.

6.1 Méthodes de mesure

- A l'estime : la plus rapide et la plus risquée → Syndrome des 90%
- Décomposer les travaux en petites unités de réalisation : Nombre d'unités à livrer ou à réceptionner, Nombre de tests à écrire/à préparer/à passer, Nombre de spécifications à écrire, Nombre de classes à développer ... → Risque de lourdeur/complexité
- D'où les indicateurs d'avancement

6.2 Indicateurs de base

- Avancement des phases amont (specs, conception) : pas évident
- Avancement du développement logiciel : décomposition en modules élémentaires, associer développement et tests unitaires
- Avancement des tests d'intégration et de validation : Nb de tests préparés/écrits/passés/OK/NOK, Nb d'anomalies/critiques, majeures, mineures/ouvertes, closes → pour éviter les surprises
- Avancement de la documentation : Nombre de documents rédigés, relus, finalisés
- Autres indicateurs : au besoin du projet

6.3 Suivi des coûts (Notions de base)

- **Prix** : montant agréé avec le client = recettes
 - Plan de paiement : paiement échelonné, par tranches
- **Coût** : montant utilisé pour réaliser le projet = dépenses
 - Heures de travail, Achats de matériel, de licences
 - Sous-projets sous-traités
 - Voyages, frais de représentation, hébergement, etc.
 - Services internes (machines, reprographie des documents, etc...)
- **Marge** = Prix – Coût : Marge positive = gain/Marge négative = perte

COUT	MARGE
PRIX	

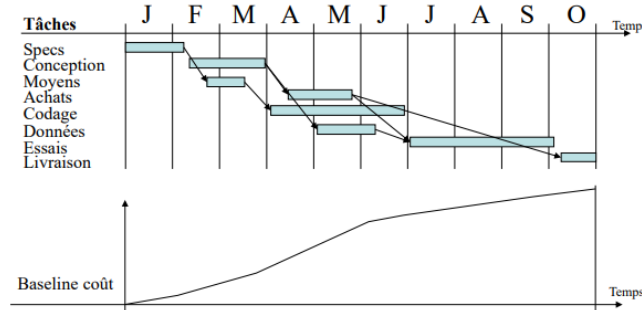
- **Devis** : coût estimé au départ (budget)
- **Réalisé** : coût réel déjà dépensé
- **Engagé** : coût réel décidé mais pas encore dépensé effectivement (exemple : commande de matériel pas encore payé)
- **Reste à faire** : coût estimé pour terminer le projet
- **Prévision à Fin d'Affaire** (PFA) = (Réalisé + Engagé) + Reste à faire Marge = Prix – PFA

A suivre par rapport au devis

Réalisé	Engagé	Reste à faire	MARGE	Réel
DEVIS			MARGE	Prévu
PRIX				

Le suivi se fait par rapport à une « baseline »

- Programme nominal prévu du projet
- Etalement dans le temps des travaux devant être accomplis et de leur valeur
- La Baseline est un planning de l'avancement et des coûts prévus



6.4 Valeur réalisée / Earned value

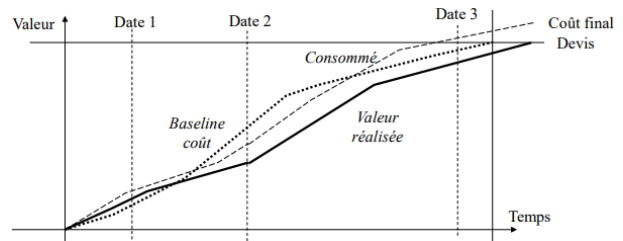
Valeur des travaux déjà réalisés à une date donnée

$$\sum \text{tâches} (\text{Coût estimé tâche} \times \text{Avancement physique})$$

C'est une valorisation de l'avancement physique

→ permet de comparer sur un même schéma :

Baseline, Consommé et Valeur réalisée



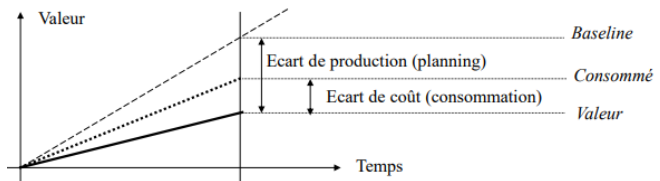
6.5 Estimation des écarts

$$(\text{Valeur réalisée} - \text{Baseline}) = \text{Ecart planning}$$

Indique s'il y a + ou - de choses faites par

rapport au plan prévu au départ

→ positif = avance planning

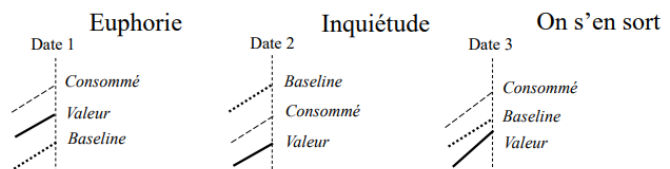


$$(\text{Consommé} - \text{Valeur réalisée}) = \text{Ecart de coût}$$

Indique s'il a été + ou - consommé que prévu

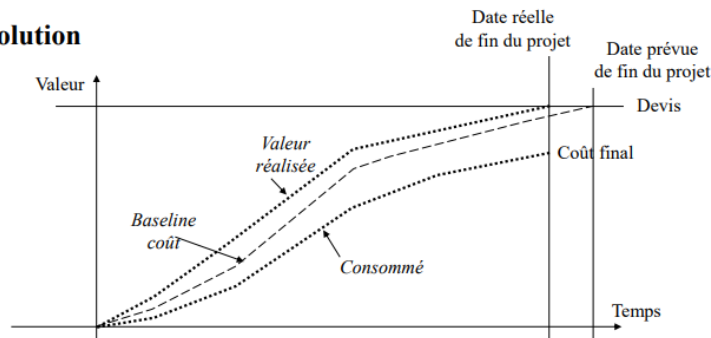
pour faire ce qui a été déjà fait

→ positif = surcoût



Exercice : Dessiner les courbes Baseline / Consommé / Valeur réalisée

Solution



Réestimation régulière du coût final et du planning

- Recueil auprès des membres de l'équipe pour chaque tâche
 - Avancement physique (indicateurs) -> Objectif
 - Coût - Réalisé + Engagé -> Objectif
 - Coût - Reste à faire -> Subjectif ... C'est le plus difficile
- Consolidation globale
 - Coût : Prévion de coût à Fin d'Affaire
 - Avancement : indicateurs d'avancement
- Mise à jour du planning à partir du reste à faire

Détecter les éventuelles dérives le plus tôt possible et réagir en cas de « dérive » (éviter que le problème s'aggrave) ou en cas d'amélioration (en profiter pour se donner des marges)

Plus une erreur est détectée tard, plus ses conséquences sont graves

7 Architecture des systèmes

Architecture : Décomposition d'un système en éléments. Les éléments sont assemblés et interagissent via les interfaces

Type d'architecture

Architecture fonctionnelle	Le système fait quoi ?	Spécification
Architecture logicielle (organique)	Le système est fait comment ?	Conception
Architecture physique (organique)		

Les trois architectures sont liées : Les fonctions sont réalisées par des logiciels et du matériel et les logiciels fonctionnent sur du matériel

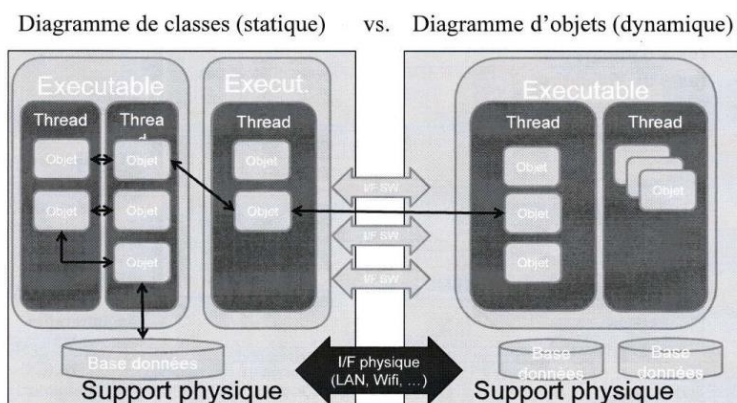
7.1 Architecture fonctionnelle

<p>Structure du système en éléments fonctionnels : Fonctions (boîtes), Interfaces fonctionnelles (échange de données, commande (flèches))</p> <p>Bien adaptée à l'analyse des besoins : Approche intuitive, expression du besoin de l'utilisateur</p> <p>Ne préjuge pas des produits à réaliser : Ce n'est ni une architecture logicielle ni une architecture matérielle. Une fonction est réalisée (en général) par du logiciel et du matériel.</p>	<pre> graph LR SD[Stockage des données] -- lecture --> AD[Affichage des données] SD -- lecture --> ED[Edition des données] ED -- écriture --> SD </pre>
--	--

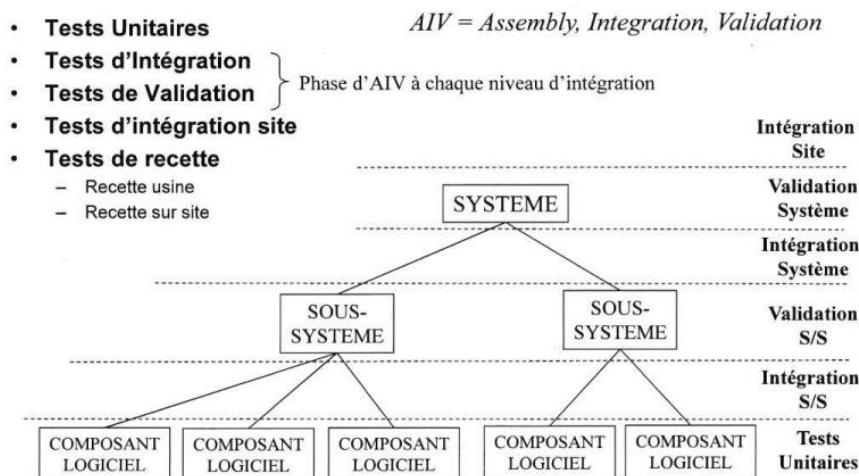
7.2 Architecture matérielle

<p>Structure du système en éléments matériels (hardware) : éléments matériels (électroniques, mécaniques), Interface physique (bus, liaisons, mécaniques), une instance par instance physique réelle.</p> <p>Phase de conception du système</p> <p>Ne montre pas les données échangées entre les logiciels</p>	<pre> graph TD E1[Ecran 1] -- VGA --> CG[Carte graphique] E2[Ecran 2] -- VGA --> CG C[Clavier] -- USB --> CC[Carte CPU] GPS[GPS] -- série --> CC CG <--> bus CC </pre>
--	--

7.3 Architecture logicielle



8 Phase de vérification (AIV)



8.1 Anticipation des vérifications

- Phase de spécification : plan de tests préliminaires (RDP)
-