




# Gestion de Projets Informatique

Chapitre: Estimation des charges

# Sommaire

- Comprendre les concepts de l'évaluation de charge avant d'appliquer les méthodes
- Les méthodes disponibles
- La démarche préconisée



## Concepts liés à l'évaluation des charges

# Précautions d'usage...

- Restons humbles : Il n'y a pas de recette miracle ...
- Il y a autant de recettes que de cuisiniers...
- Il faut adapter sa recette aux ingrédients...
- On ne calcule jamais des charges, on peut tout au mieux les évaluer...
- Pour un besoin fonctionnel donné, il peut y avoir plusieurs résultats : intervalle de confiance, évaluation des risques.
- Une bonne méthode est guidée par le retour d'expérience
- Les bases mathématiques nécessaires : la règle de trois.

# Notions et définitions

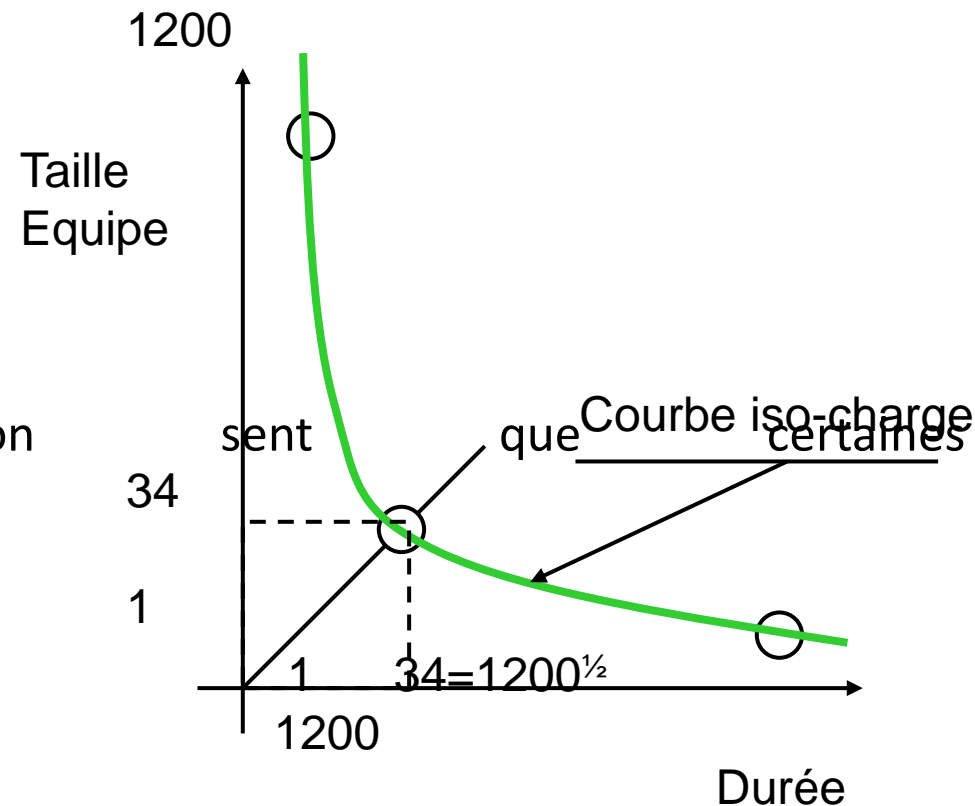
- **Charge** : quantité de travail nécessaire pour effectuer des travaux. Mesurée en jours.homme (et non pas jours/homme). Dérivés : mois.homme, années.homme.
- **Durée** : Temps alloué au projet. Prise en compte des contraintes de planning (schéma directeur)
- **Effort** : **taille de l'équipe = Charge / Durée**
- Sauf que l'échelle « temps » est très particulière :
  - Un mois comporte 20 jours...
  - Une année comporte 240 jours pour une ressource externe...
  - Une année comporte 200 jours pour une ressource interne.

# Les limites de la règle de trois...

Soit un projet de 1200 jours.hommes (60 mois.homme). Il existe une infinité a priori de solutions pour sa réalisation :

- 1 homme pendant 1200 jours (6 ans) ?
- 1 jour pour 1200 hommes ?
- 34 hommes pendant 34 jours ?
- 6 hommes pendant un an ?
- 12 hommes sur 6 mois ?
- 72 hommes sur 1 mois ?

Par expérience, on sent que certaines répartitions sont plus logiques...



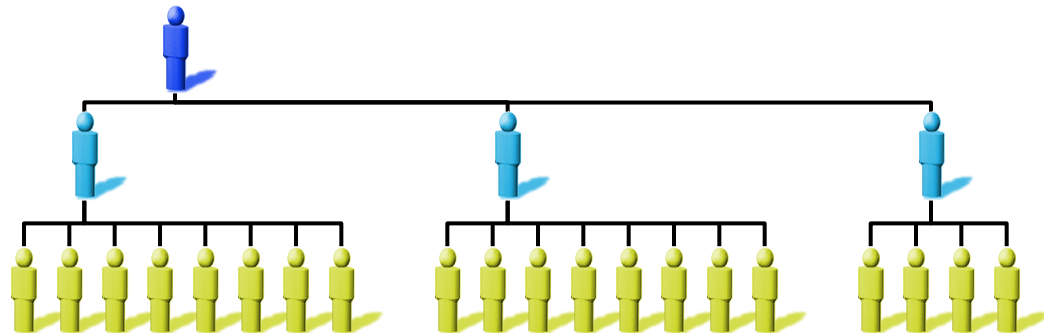
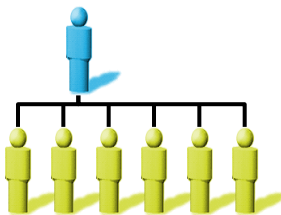
# Loi de la racine carrée

Le Chef de projet pourra également valider ses estimations de charge en validant la formule suivante issue du Modèle COCOMO.

$$\text{Délai du Projet (en Mois)} = \sqrt{\text{Charges du Projet (En Mois X hommes)}}$$

# La structuration de l'équipe projet

- Une équipe projet doit se modéliser sous l'aspect d'un «râteau» à huit dents (appelé aussi «éventail de contrôle») :
  - On considère qu'il faut une ressource « d'encadrement » en moyenne pour 8 «employés»
  - Cela peut aller jusqu'à 10 personnes maxi
  - Pilotage optimal : 5 à 6 personnes
- Equipe de 6 personnes :      Equipe de 20 personnes :



- Par ailleurs, tout ceci n'est qu'un découpage théorique, le découpage réel doit suivre le découpage opérationnel

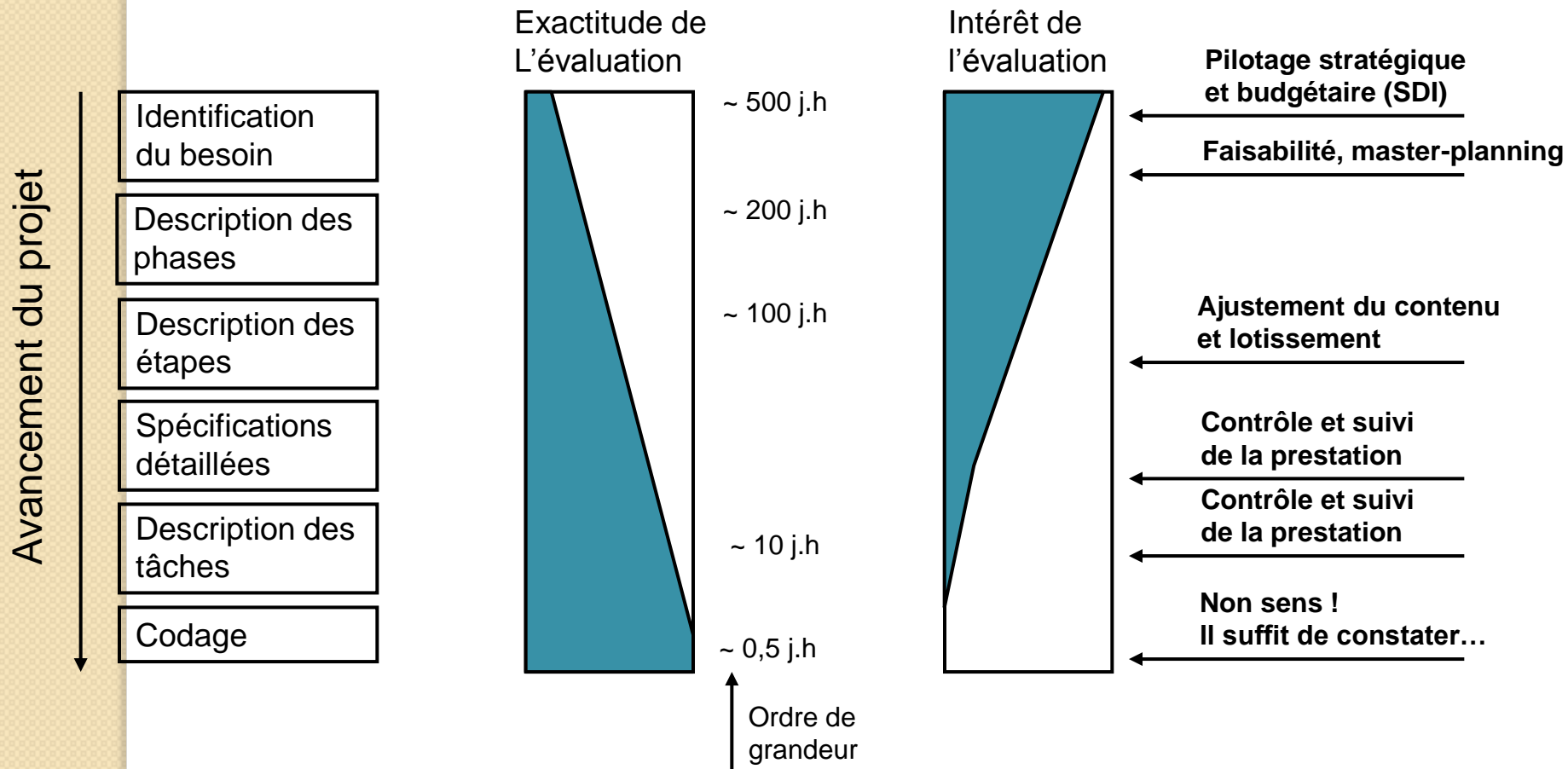


# Conséquence de la théorie du râteau et limites.

- Les charges... mais au fait de quelles charges parle t'on ?
- Il s'agit généralement de charges de développement auxquelles il faut ajouter les charges dites connexes, par exemple le pilotage de projet
- Selon la théorie du râteau : 1 « CP » pour 8 développeurs => charge de pilotage = 12,5% , à laquelle il faut ajouter 2,5% pour les projets plus importants ( > 8 personnes). En moyenne, on peut l'évaluer à 15%.
- Quelques contraintes supplémentaires :
  - Prendre en compte les très petits projets : ex : projet de 80 jours sur 4 mois : 1 développeur temps plein + 1 CP à 15% de son temps ?
  - Prendre en compte les projets très importants : on ne peut pas empiler indéfiniment des couches de pilotage ; effet de masse, d'apprentissage, économies d'échelle, ...

# Tout le dilemme de l'évaluation de charges...

- On peut d'autant plus finement estimer les charges que l'estimation ne sert plus à rien...



# L'évaluation des charges : une logique floue...

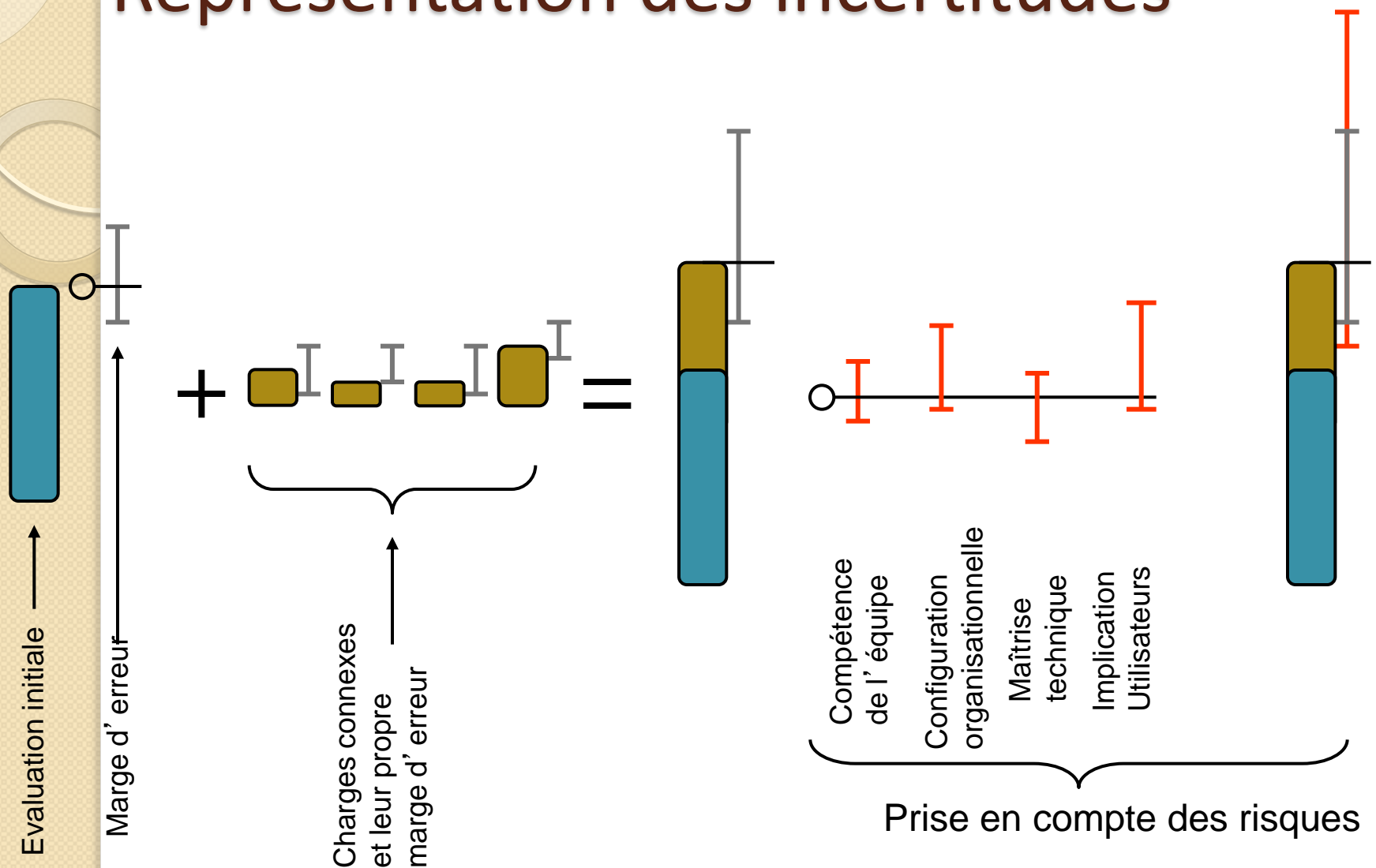
- Comme on l'a vu, on ne peut au mieux qu'évaluer (estimer) les charges
- On commence généralement par évaluer les charges de développement, desquelles on déduit des charges connexes, et auxquelles on ajoute des coefficients...

Evaluation des charges de développement → Ajout charges connexes → Prise en compte de contraintes et de risques

« tangible », mathématique ←————→ Flou, subjectif

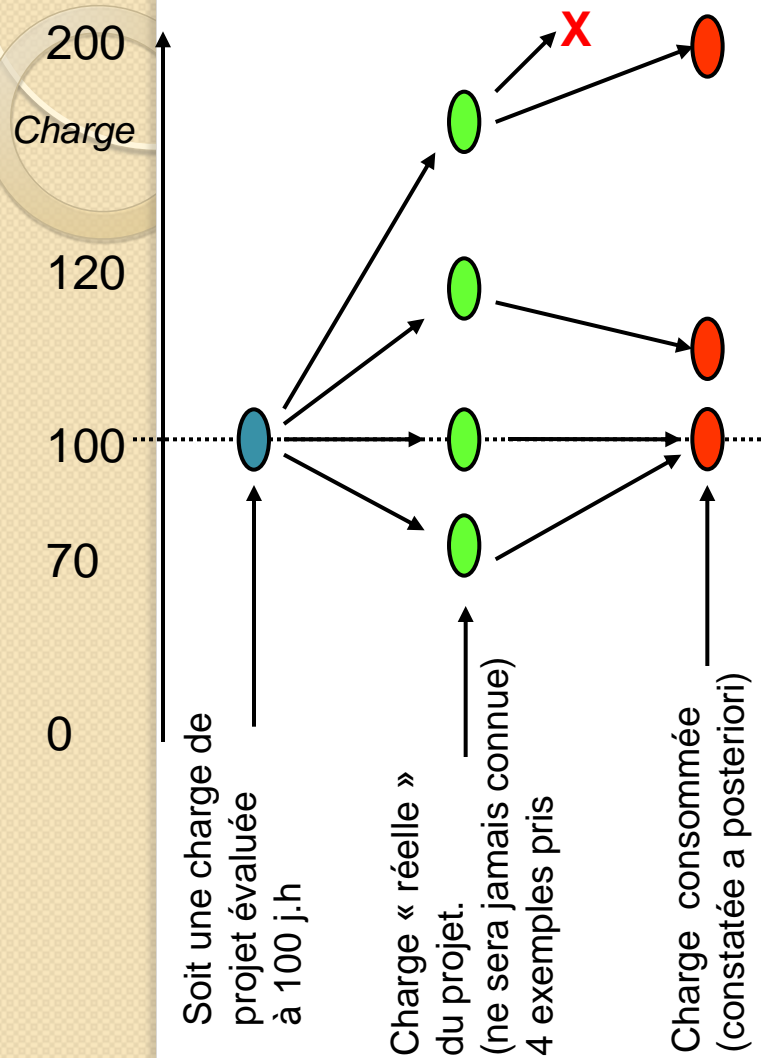
- ... d'où l'importance de l'estimation des charges de développement
- ... d'où l'importance du retour d'expérience pour les charges connexes
- ... et toute la difficulté de la subjectivité concernant les coefficients de majoration

# Représentation des incertitudes



**NB : ne pas oublier que les incertitudes ne se compensent jamais, elles s'ajoutent !**

# Les dangers d'une mauvaise évaluation



- Si charge réelle = 100 j.h : aucun problème
- Si charge réelle = 70 j.h, la charge consommée = 100 j.h : «loi de Parkinson» : «le travail se dilate jusqu'à remplir le temps disponible»
- Si charge réelle = 120 j.h : possibilité de réduire la charge réelle vers 110 j.h à condition de faire des concessions (qualité) et de supporter la pression (stress) : ne peut être que temporaire
- Si charge réelle = 200 j.h : la charge consommée sera en pratique encore supérieure (220 j.h ?), due aux conflits sur le projet. Le niveau de qualité sera vraisemblablement mauvais. Au pire, le projet peut être abandonné.

# Les « méthodes »

# Les « classes de méthodes »

- **Les méthodes basées sur le « jugement d'experts » :**
  - évaluation subjective selon l'expérience de chacun + challenge entre experts.
  - Difficulté : il faut beaucoup d'experts disposant d'une forte expérience + avoir formalisé le retour d'expérience
- **Les méthodes dites « analytiques » car basées sur les principes de la comptabilité analytique : les unités d'œuvre.**
  - Difficultés : quelles unités d'œuvre ? Quelle évaluation unitaire ? Pb de cohérence et passerelles entre les unités d'œuvre ? Quelle évaluation de charges pour des compétences transverses ? Suppose de connaître le projet de manière assez détaillée.
- **Les méthodes de répartition proportionnelle :**
  - à partir d'une charge pour un périmètre donné, en déduire les charges « connexes » ou la décomposition interne de charges
- **« Non méthode » : par l'occupation prévisionnelle des ressources.**

# La méthode Delphi

- Basée sur le jugement d'experts.
- Principes, sur la base de deux exemples :
  - Ex 1 : gestion de clients pour une banque A = 5000 j.h. Sachant qu'on l'applique à une banque B pour laquelle l'architecture technique est mieux maîtrisée et pour laquelle on peut migrer une partie des développements sans pour autant tout recoder => 3500 j.h
  - Ex 2 : Projet a créer qui reprend 30% des fonctionnalités d'une application de 9000 j.h + 50% d'un projet technique réalisé initialement pour 1500 j.h =>  $9000 \times 0,3 + 1500 \times 0,5 = 3750$  j.h
- C'est une technique d'évaluation « au doigt mouillé ».
- Il s'agit de bien évaluer les proportions (subjectives) et d'éviter les pièges : un projet de gestion d'un dossier client dans une banque A comptant 500.000 clients n'est pas 4 fois moins complexe que pour une banque B comportant 2.000.000 de clients, mais ce n'est pas non plus la même chose...



# La méthode Delphi (2)

- En pratique :
  - Il faut au moins 3 experts
  - D'autant plus réaliste qu'il y a d'experts, mais une multitude d'experts n'est pas réaliste (sinon ce ne sont plus des experts...)
- Organisation :
  - 1) Chaque expert formule sa proposition sur la base de sa propre expérience
  - 2) Les jugements sont rendus publics tout en restant anonymes
  - 3) Chaque expert refait une nouvelle estimation, chacun devant considérer que les autres propositions sont faites par des experts (il doit se poser des questions...)
  - 4) Mise en commun des estimations, de manière nominative. Chaque expert doit « défendre » son estimation et se justifier à partir de sa propre base de connaissance
  - 5) Révision en séance des estimations de chacun.
  - 6) Sauf conflits, le groupement d'expert arrive généralement à un consensus

# Méthode de répartition proportionnelle

- Pré-requis :
  - Avoir une estimation globale du projet sur tout ou partie (par ex. sur la base d'une estimation Delphi)
  - Définir des clés de répartition par étape du projet et par type d'acteur
- Trois approches (souvent complémentaires) :
  - Approche descendante : décomposition proportionnelle pour connaître les sous-étapes
  - Approche ascendante : à partir d'une estimation partielle, déduction des phases connexes
  - Approche dynamique (en cours de projet) : on a constaté une consommation de charges pour un lot donné, il faut appliquer les constatations (et les adapter) aux lots suivants.
- Les proportions définies dépendent de la structure, du type de projet
- Elles doivent être considérées en partie comme des recommandation et en partie comme des règles

# Méthode de répartition proportionnelle (2)

- Quelques constats :
  - Étude préalable = 10% du total du projet
  - Etude détaillée = 20 à 30% du total du projet
  - Etude technique = 5 à 15% du total du projet
  - Réalisation = 2 \* charge d' étude détaillée
  - Mise en œuvre = 30 à 40% de la charge de réalisation
- Attention ! Toutes ces répartitions sont basées sur des considérations de socle technique opérationnel et maîtrisé, compétences acquises par les équipes, infrastructure OK, ...
- Elles sont donc : hors mise en place de framework, reprise des données, installation et mise en œuvre de nouveaux outils, préparation des environnements (dev, recette, prod), formation des personnels, ...

# Méthode de répartition proportionnelle (3)

- Une méthode « récursive » qui s'applique à elle-même pour le redécoupage et/ou l'extension des étapes :
  - Etude préalable (base 100%) :
    - Phase d'observation : 30 à 40%
    - Phase conception/ organisation : 50 à 60%
    - Phase d'appréciation : 10%
  - Pilotage de projet :
    - 20% de la charge de réalisation pour la phase de réalisation
    - 10% aux autres étapes
  - Recette : 20% de la charge de réalisation
  - Documentation utilisateur : 5% de la charge de réalisation

# Les modèles Cocomo et Diebold

- Base de la méthode : un informaticien a une idée précise de la charge pour développer 100 lignes de code, mais pas vraiment de la charge globale d'évaluation du projet.
- C'est une méthode de corrélation entre le nombre de lignes de code et la charge de travail .
- Elle prend en entrée le nombre de milliers d'instruction source (lignes de programmes) écrites et testées, en dehors des lignes de commentaire **(kisl)**
- Elle produit en sortie :
  - La charge totale du projet
  - Le délai normal du projet
  - La taille moyenne de l'équipe
- Elle intègre des facteurs de risque selon 4 sources : les exigences attendues du logiciel, les caractéristiques de l'environnement technique, les caractéristiques de l'équipe projet et l'environnement même du projet

# Les modèles Cocomo et Diebold (2)

- Les formules pour l'évaluation des charges et des délais : (formules très simplifiées...)

Type de projet	Charge (mois.homme)	Délais (mois)
<b>Simple</b>	$3,2 \times (\text{kisl})^{1,05}$	$2,5 \times (\text{kisl})^{0,38}$
<b>Moyen</b>	$3,0 \times (\text{kisl})^{1,12}$	$2,5 \times (\text{kisl})^{0,35}$
<b>complexe</b>	$2,8 \times (\text{kisl})^{1,2}$	$2,5 \times (\text{kisl})^{0,32}$

- Exemple : programme simple de 40.000 instructions source (40 kisl) :
  - Charge =  $3,2 \times (40)^{1,05} = 154$  mois.homme
  - Délai =  $2,5 \times (40)^{0,38} = 17$  mois
  - Taille moyenne de l' équipe =  $154/17 = 9$  personnes

# Les modèles Cocomo et Diebold (3)

- Les facteurs de correction :
  - les exigences attendues du logiciel :
    - Fiabilité : si forte exigence de fiabilité (ex : contrôle aérien) : ↗ cher
    - Complexité de la base de données
    - Complexité des algorithmes
    - Temps d'exécution (ex : contraintes temps réel)
  - les caractéristiques de l'environnement technique :
    - Taille mémoire nécessaire (désuet)
    - Stabilité de l'environnement technique
    - Disponibilité des machines de tests / de recette
  - les caractéristiques de l'équipe projet
    - Compétences et expérience des concepteurs
    - Compétences des développeurs
    - Connaissance de l'environnement technique
    - Connaissance du langage
  - L'environnement même du projet
    - Utilisation de méthodes ou d'un AGL (désuet)
    - Forte contrainte de délai

NB : La **valeur** des coefficients n'a que peu d'intérêt car à portée interne à la société dirigée par le créateur de la méthode

# Les modèles Cocomo et Diebold (4)

- Le modèle Diebold est une variante de Cocomo (simplifiée) :
  - **Temps = (complexité) \* (savoir-faire) \* (connaissance) \* KISL**
  - Complexité du logiciel : entre 10 et 40 (très complexe)
  - Savoir faire = expérience de l'équipe : entre 0,65 et 2 (peu de savoir faire)
  - Connaissance : celle de l'environnement fonctionnel et technique : entre 1 (OK) et 2 (faible connaissance)
- Conséquence : pour un programme de 1000 lignes, l'estimation peut varier de 6,5 jours à 160 jours !
- Ce dernier cas est peu vraisemblable car on donne alors un projet très complexe à une équipe peu expérimentée qui ne connaît rien de l'environnement du projet... il faut aimer les risques !



# La méthode de l'évaluation analytique

- Méthode très utilisée pour évaluer la charge de réalisation.
- Couplée à la méthode de répartition proportionnelle, elle permet d'évaluer le globalité du projet
- Méthode dite « analytique » car inspirée de la comptabilité analytique :
  - Identification des unités d'œuvre élémentaires et la typologie de leur difficulté (quel que soit le type de projet)
  - Valorisation des unités d'œuvre (suivant le type de projet)
  - Identification pour un projet de la répartition des unités d'œuvre à consommer
- => évaluation de la charge de réalisation
- **Contrainte : nécessite d'avoir accès au cahier des charges, ou mieux, aux spécifications fonctionnelles**
- Ne prend pas en considération les charges techniques annexes, il s'agit uniquement de tâches de développement applicatif

# La méthode de l'évaluation analytique (2)

- Exemple : applications transactionnelles

<i>(charges en j.h)</i>	Facile	Moyen	Difficile
Menu	0,25	0,5	1
Consultation	1	2,5	4
Mise à jour	1,5	3	5
Edition temps réel	1	2	4

- Exemple : programmes batch

<i>(charges en j.h)</i>	Facile	Moyen	Difficile
Extraction	0,5	1	1,5
Mise à jour	2	3	5
Edition	1,5	2,5	4

- Ces exemples n'ont que peu de valeur tant que l'on a pas spécifié précisément :
  - ce que l'on comprend dans les unités d'œuvre (ex : tests unitaires ? Tests d'intégration ? Documentation ? Suivi de projet ?)
  - ni la caractérisation des niveaux (simple, moyen, difficile)

# La méthode de l'évaluation analytique (3)

- Généralement, les charges couvrent : le développement, les TU, les TI, la documentation technique de l'application.
- Sont hors évaluation : l'évaluation du besoin (cahier des charges), la rédaction de spécifications (puisqu'elles sont supposées exister afin de pouvoir effectuer le découpage analytique), la recette, le pilotage de projet.
- Il faut donc généralement ajouter 10% au titre de la vérification technique (pré-VA) et 20% au titre du pilotage de projet
- Autres éléments liés à cette méthode :
  - Taille de l'équipe : une règle empirique indique que la taille de l'équipe doit être inférieure ou égale à la durée du projet en mois (cf liens avec râteau et durée max du projet = 12 mois)
  - Durée minimum du projet : elle est évaluée à  $2,5 (\text{charge})^{1/3}$ , où la charge est exprimée en mois.homme.

# La méthode des points de fonction

- Principe : faire une évaluation du logiciel d'un point de vue externe, ses « fonctions ».
- Cette méthode distingue cinq type d'unités d'œuvre et trois degrés de complexité.
- Le principe de fonctionnement est ensuite très approchant à celui de l'évaluation analytique, sauf que l'on arrive pas directement à une charge en j.h mais en points de fonction. Les méthodes pour passer des points de fonction aux charges dépendent de l'environnement technique.
- La méthode inclut une correction de la charge brute en fonction de 14 critères. La correction peut être de +/- 35 %
- La méthode inclut un retour d'expérience : on évalue les charges avant et après le projet. Les écarts sont répartis en deux catégories : ceux qui résultent d'un changement de périmètre sont écartés. L'analyse des résultats finaux permet d'affiner les coefficients de détermination des points de fonction.

# La méthode des points de fonction (2)

- Calcul de la taille du projet : composants relatifs aux données :
  - Unité d'œuvre GDI : groupe logique de données internes logiquement liées
    - En pratique, il s'agit d'un écran porteur d'une entité de la base de données.
    - Les éléments d'un GDI sont accessibles en lecture et écriture
    - Quelquefois, des entités de cardinalité (1,1) peuvent être considérées comme un même GDI
    - Un GDI comporte des Données Élémentaires (DE) : les propriétés de l'entité. On compte une DE par champ, y compris pour les champs techniques (PK, FK)
    - Un GDI est éventuellement composé de sous ensembles logiques de données (SLD)
  - Unité d'œuvre GDE : groupe logique de données externes
    - Principe idem GDI, y compris pour DE et SLD
    - sauf qu'il s'agit du GDI appartenant à un autre domaine : on ne fait que l'interroger

# La méthode des points de fonction (3)

- Complexité des GDI et des GDE = fct (DE, SLD)

	1 à 19 DE	20 à 50 DE	+ 51 DE
1 SLD	Faible	Faible	Moyenne
2 à 5 SLD	Faible	Moyenne	Elevée
+ 6 SLD	Moyenne	Elevée	Elevée

- Nombre de points de fonction GDI

	Faible	Moyenne	Elevée
Pt Fonction GDI	7	10	15

- Nombre de points de fonction GDE

	Faible	Moyenne	Elevée
Pt Fonction GDE	5	7	10

# La méthode des points de fonction (4)

- Calcul de la taille du projet : composants relatifs aux traitements :
  - ENT : Entrée
    - Il s'agit d'un flux de données , provenant d'un écran ou d'un flux externe, qui met à jour une ou plusieurs entités du système. Critère : à la fin d'un traitement, le système doit être dans un état stable (ie : si MAJ de données interdépendantes dans des tables relationnelles, il s'agit de l'ensemble de la MAJ de ces tables)
    - On compte le nombre de DE (Attention : on ne compte qu'une fois une DE même si le champ est répétitif)
    - On introduit la notion de GDR (groupe de données référencées) : c'est soit un GDI, soit un GDE
  - SOR : Sortie
    - Il s'agit d'un flux de données envoyé à l'extérieur du domaine, qui ne modifie pas le domaine. Les DE en sortie peuvent être des champs calculés.
    - Il peut s'agir d'une édition, d'un flux de données externe
  - INT : interrogation
    - Ce sont des flux qui ne mettent pas à jour le système, ils sont à destination interne du système et ne génèrent pas de champs calculés

# La méthode des points de fonction (5)

- Complexité des ENT, SOR et INT = fct (DE, GDR)

	1 à 5 DE	6 à 19 DE	+ 20 DE
0 ou 1 GDR	Faible	Faible	Moyenne
2 à 3 GDR	Faible	Moyenne	Elevée
+ 4 GDR	Moyenne	Elevée	Elevée

- Nombre de points de fonction ENT

	Faible	Moyenne	Elevée
Pt Fonction ENT	3	4	6

- Nombre de points de fonction SOR

	Faible	Moyenne	Elevée
Pt Fonction SOR	4	5	7

- Nombre de points de fonction INT

	Faible	Moyenne	Elevée
Pt Fonction INT	3	4	6



# La méthode des points de fonction (6)

- Deuxième étape : l'ajustement de la taille
  - Principe : ajuster le nb de points de fonctions obtenus en fonction de 14 critères appelés « caractéristiques générales du système » (CGS)
  - Chaque critère se voit attribuer une note entre 0 et 5 appelée « degré d' influence » (DI)
    - 0 : degré d'influence très faible, sujet bien maîtrisé, impliquera une diminution des points de fonction
    - 5 : degré d'influence fort, risques importants, impliquera une augmentation des points de fonction
  - L' ensemble des caractéristiques permet de faire varier les points de fonction de +/- 35%
- Ajustement de la taille : calcul des points de fonction ajustés (PFA)
  - Facteur d'ajustement :  $FA = 0,65 + DI / 100$
  - $PFA = FA * PFB$

# La méthode des points de fonction (7)

- Les Caractéristiques générales du système (CGS)

N°	Caractéristique	Notation (entre 0 et 5)
1	Communication des données	
2	Système distribué	
3	Performance	
4	Intensité d' utilisation de la configuration matérielle	
5	Taux de transaction	
6	Saisie Interactive	
7	Convivialité	
8	Mise à jour en temps réel des GDI	
9	Complexité des traitements	
10	Réutilisation du code de l' application	
11	Facilité d' installation	
12	Facilité d' exploitation	
13	Portabilité de l' application	
14	Facilité d' adaptation	
	Total : Degré d' Influence Total (DIT)	

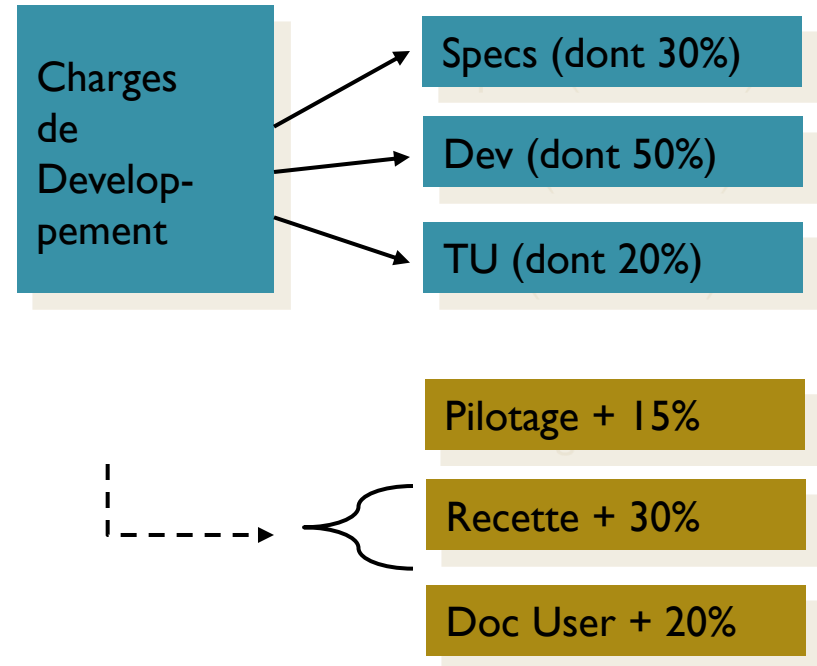
# La méthode des points de fonction (7)

- Dernière étape : transformer les points de fonction en charge
  - Il s'agit d'appliquer un coefficient, dont la valeur dépend de l'environnement technique du projet
  - Il doit en pratique être déterminé par le retour d'expérience de l'entreprise
  - Ce coefficient donne en sortie une **charge de développement**
- Méthode de conversion des PFA en ISL (pour application de la méthode Cocomo) :  $ISL(Cobol) = 118,7 * PFA - 6490$ . N'a d'intérêt que pour les projets Cobol
- Autres estimations :
  - En fin d'étude préalable : 1 point de fonction = 3 jours.homme (2 jours pour petits projets, 4 jours pour projets importants)
  - En fin d'étude détaillée : 1 point de fonction =
    - 1 jour.homme si environnement client serveur ou L4G
    - 2 jours.homme si environnement grand système
    - 0,5 jour.homme si développement RAD
    - Moyenne standard : entre 1,3 et 1,8 jour.homme

# Démarche préconisée

# Quels éléments à évaluer ?

- Première étape : définir ce que l'on entend par « charges de développement » et l'estimer
  - Toutes (presque) les méthodes d'évaluation des charges se concentrent sur les charges de développement (l'élément le plus analytique et « concret »)
  - Ces charges de développement couvrent généralement les tâches de réalisation et de tests unitaires + règles de gestion ? + spécifications fonctionnelles détaillées ?
- Deuxième étape : étendre l'estimation en amont et en aval (répartition proportionnelle) :



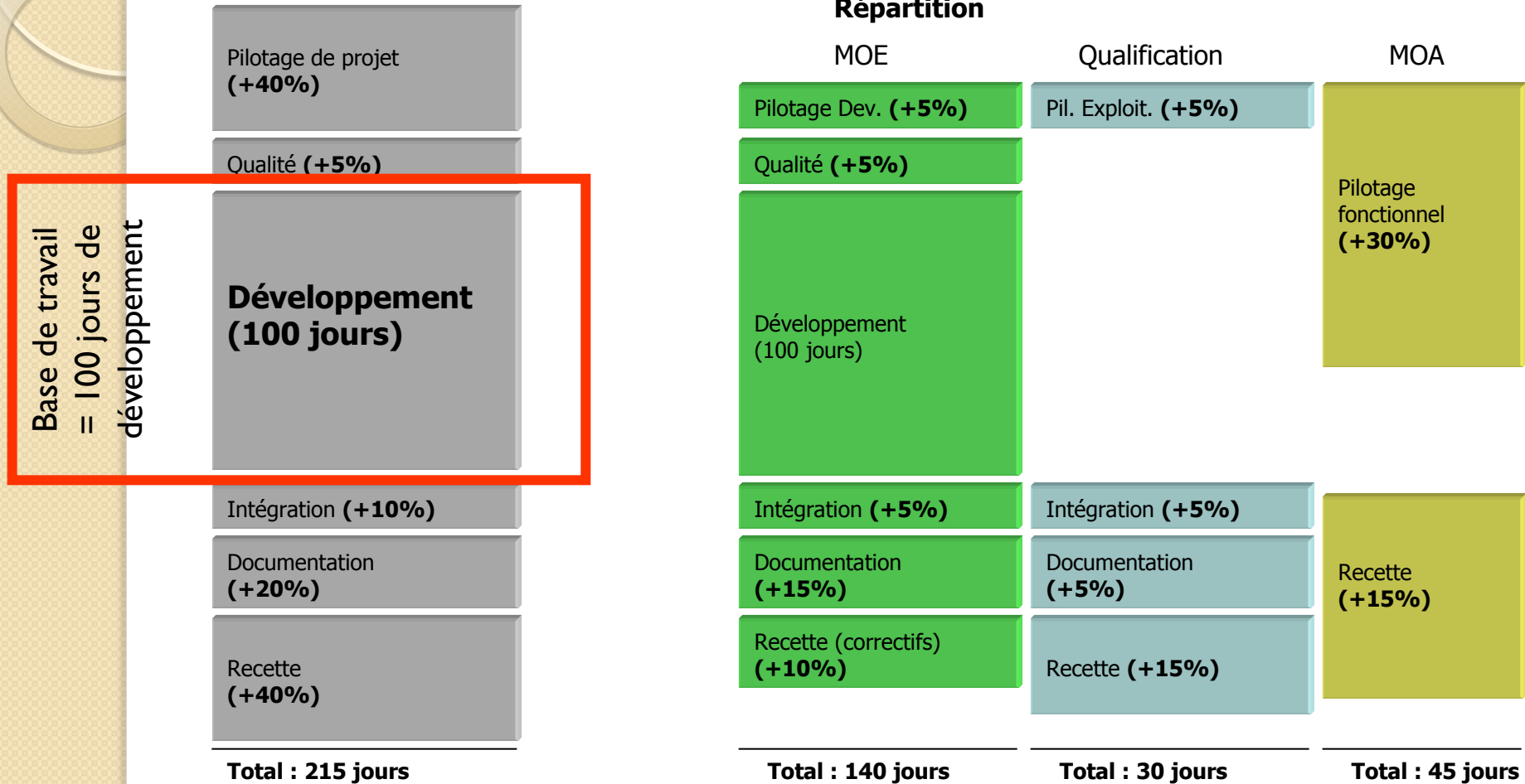
# Les charges connexes

- Les charges déduites « en aval » des charges de développement sont appelées charges connexes.
- On évalue les charges connexes côté AMOE, par exemple :
  - Pilotage de projet (participation réunions, interlocuteur client, suivi des ressources, vérification des engagements)
  - Evaluation de projet (évaluation d'un besoin et des bons de commande)
  - Assurance Qualité (PAQ + audits + revues)
  - Assistance intégration
  - Assistance recette (VA)
  - Assistance recette (VSR) + période de garantie
  - Support aux utilisateurs
  - Documentation utilisateur (autres documentations incluses dans les phases de réalisation)
  - ...

# Les charges connexes (2)

- Par extension, on peut déduire également des charges connexes côté MOA, car les opérations de gestion du besoin, de recette MOA, de support utilisateur peuvent être, elles aussi, proportionnelles à la taille du projet. Par exemple : analyse du besoin, réalisation du cahier des charges, assistance à la recette technique, VA, VSR, VSP, suivi des incidents en production,...
- Garder le principe de modularité, car il permet :
  - Une bonne ventilation, notamment pour le retour sur expérience
  - De « débrayer » certaines ventilations automatiques sous forme de prorata des charges de développement si l'intervention peut être mesurée plus précisément
- Par exemple :
  - si le pilotage de projet se limite aux présences aux réunions, l'évaluation de charge se fait par rapport au temps de présence à ces réunions ;
  - si certaines tâches sont relatives à la production de documentation, on peut appliquer la technique des notaires...

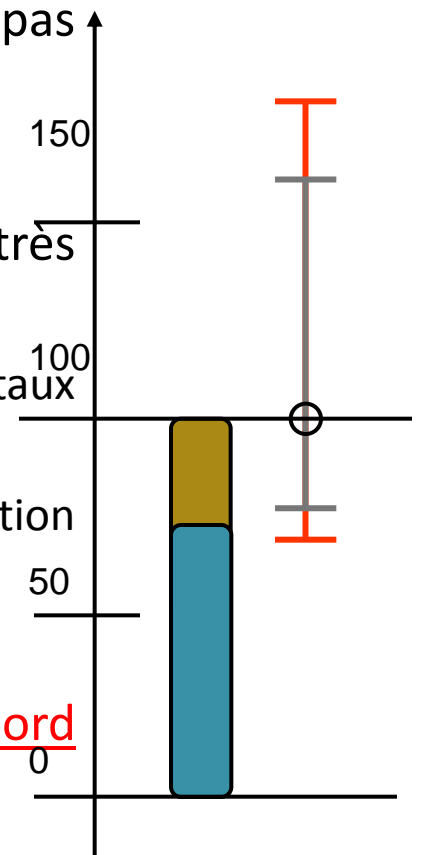
# Exemple de ventilation de charges connexes





# Evaluation de l'intervalle de confiance

- Très difficile à évaluer car il s'agit généralement de prendre en compte des risques, qui ne devraient pas arriver... parce qu'on les prend en compte !!!
- Approche du modèle « Use Case Points » très intéressante, car :
  - elle distingue les facteurs environnementaux des facteurs techniques
  - elle ne donne pas la même pondération aux facteurs de risque
- Prendre en compte les risques, c'est d'abord les évaluer



# Charges : Pour qui ? Pour quoi ? Quand ?

- Il faut savoir :
  - Quelles charges mesurer ? (dev ? globales ? )
  - Pour qui : MOE ? MOA ?
  - A quel moment ? Analyse du besoin ? Cahier des charges ?  
Spécifications détaillées ?
  - Evaluation des charges ? De l' équipe ? De la durée ?
- Il faut savoir « débrayer » les méthodes. Elles s'appliquent pour des projets « standard », mais ne sont pas fiables :
  - Pour les petits projets : il faut une masse critique suffisante (3 pers.)
  - Pour les très grands et/ou longs projets : il faut prendre en compte des facteurs d' adaptation tels que l'apprentissage, l'essoufflement de la demande (TMA), les économies d'échelle pour les charges connexes, ...
  - Pour les projets à dominante trop technique (ex : migration de BDD, ...), il faut évaluer les charges techniques en supplément sur la base d' autres abaques
- Principe : appliquer ces méthodes ... quand on ne sait pas faire autrement !

# Les tests de robustesse et de sensibilité

- Une méthode n'est efficace qu'aux deux conditions suivantes :
  - Elle est ROBUSTE : plusieurs utilisations de la méthode sur des cas similaires produit un résultat similaire
  - Elle est SENSIBLE : une variation en entrée des paramètres produit en sortie une variation du résultat qui est proportionnelle
- Comment faire ???
  - Robustesse : s'évalue par le retour d'expérience, de deux manières :
    - Pour chaque projet, analyse des charges initialement évaluées et des charges constatées. A quoi est dû l'écart (éléments oubliés lors de l'évaluation ou méthode non appropriée ?)
    - Etablir une base de comparaison inter-projet comme base de travail Delphi
  - Sensibilité : dans le cas de projets très complexes prenant en compte plusieurs sous-ensembles fonctionnels, évaluer le projet dans son ensemble puis individuellement pour chaque sous-ensemble. Les clés de répartition obtenues doivent être logiques (ie représenter la complexité globale et non dépendre soit trop des écrans, soit trop des règles de gestion).

# La « bonne » méthode

- Préconisations :

- Delphi : ☒☒☐☐☐ (utilisation informelle)
- Répartition proportionnelle : ☒☒☒☒☒
- Cocomo / Diebold : ☐☐☐☐☐ (sauf démarche de correction)
- Evaluation analytique : ☒☒☒☒☐ (UO adaptées)
- Points de fonction : ☒☒☒☐☐ (mix du principe avec éval. analytique)
- Use Case points : ☒☒☐☐☐ (facteurs de correction)

- La démarche :

- Evaluation d'UO (écrans, requêtes, règles de gestion)
- Application de coefficients de pondération
- Application de « passerelles » pour des cas particuliers
- Application de charges connexes

# Méthode préconisée

- **A] Evaluation des CHarges Brutes (CHB)**
- Elles contiennent : la réalisation des spécifications fonctionnelles détaillées, les règles de gestion, le codage, les tests unitaires, les tests d'intégration « usine ».
- Evaluation basée sur trois unités d'œuvre :
  - UO « écran » : permet de mesurer la complexité de la partie IHM du projet. Par extension, ce groupe fonctionnel est également utilisé quand il s'agit d'évaluer la complexité des éditions. Exclu : traitements métier propres aux champs à traiter (voir UO « règles de gestion »)
  - UO « requête » : permet de mesurer la complexité des accès aux données et le nombre d'accès aux tables effectués depuis les écrans (pour un même nombre de données affichées, un écran de consultation et un écran de modification ne représentent pas la même charge)
  - UO « règles de gestion » : permet de mesurer la complexité des aspects « métier ». Chaque zone d'écran engendre au moins une règle de gestion (celle relative au format primaire de la donnée : texte, date, montant, ...).

# Méthode préconisée (2)

- UO « Ecran »

- Décomposition :

- Nb écrans de moins de 4 champs : écrans comportant moins de 4 champs (1)
    - Nb écrans de 4 à 7 champs : écrans comportant de 4 à 7 champs (1)
    - Nb écrans de 8 à 11 champs : écrans comportant de 8 à 11 champs (1)
    - Nb écrans de 12 à 15 champs : écrans comportant de 12 à 15 champs (1)
    - Nb écrans de plus de 16 champs : écrans comportant de 16 à 25 champs (1). Au-delà de 25 champs, on compte plusieurs écrans de type « plus de 16 champs » pour chaque tranche de 25 champs (2).

- Notes :

- (1) : compter TOUS les champs de l'écran (consultation, saisie, cachés) sauf les boutons. Groupes RB = 1 champ ; 1 case à cocher = 1 champ. Si présence d'onglets : chaque onglet = 1 écran. Ne pas compter l'écran maître des onglets (sauf s'il dispose de champs propres).
    - (2) : exemples : un écran de 20 champs est comptabilisé comme « 1 écran de + de 16 champs » ; un écran de 30 champs est comptabilisé comme « 2 écrans de + de 16 champs » ; un écran de 52 champs est comptabilisé comme « 3 écrans de + de 16 champs ».

# Méthode préconisée (3)

- UO « Requêtes » :
  - Décomposition :
    - Nb requêtes simples : nombre de requêtes de type « consultation » : une lecture simple + prise en charge des données dans l'application (mapping données – écran)
    - Nb requêtes normales : nombre de requêtes de type « mise à jour simple » : formatage des données et écriture dans une seule entité
    - Nb requêtes complexes : nombre de « mises à jour complexes » : 1 requête de formatage de données et écriture simultanée dans plusieurs tables (tables relationnelles)
  - Note : Dans les cas où plus de trois tables sont mises à jour simultanément pour un écran, on compte un nombre de « requêtes complexes » par tranche de 4 tables mises à jour. Par exemple : 5 tables mises à jour = 2 « requêtes complexes » comptabilisées ; 9 tables mises à jour = 3 « requêtes complexes » comptabilisées, ...

# Méthode préconisée (4)

- UO « Requêtes » :

- Exemples :

- pour un écran de consultation simple, on compte 1 requête « simple » qui consiste à récupérer les données et les préparer pour l'affichage
    - pour un écran de saisie simple (= 1 table uniquement), on compte une requête « simple » de récupération des données (si les données proviennent de la base ; dans le cas où elles sont chargées en mémoire, on ne compte pas de requête) + une requête de mise à jour simple
    - pour un écran de saisie complexe (modification simultanée de plusieurs tables, par exemple : personne + personne physique + état civil), on compte une requête « simple » de récupération des données (si les données proviennent de la base ; dans le cas où elles sont chargées en mémoire, on ne compte pas de requête) + une requête de mise à jour complexe



# Méthode préconisée (5)

- UO « Règles de gestion »
  - Décomposition :
    - Nb règles de gestion de base : traitement élémentaire d'une donnée d'affichage (date, code nomenclature = vérification de la présence dans une table, montant, caractères autorisés, ... dans un but d'affichage uniquement)
    - Nb règles de gestion avec un paramètre (zone de saisie simple : exemple : code présent dans une table nomenclature, code sexe, montant)
    - Nb règles de gestion de moins de 4 paramètres (zones de saisie faiblement dépendantes, avec moins de 4 critères : ex : « date de fin > date de début » + « date de fin > date de naissance » + « date de fin > date d' application »)
    - Nb règles de gestion de 4 paramètres et plus (concerne les zones de saisie très fortement dépendantes)
  - Notes :
    - 1 - les règles de gestion de type « présence obligatoire », « valeur de préaffichage » ne comptent pas comme un paramètre. Leur évaluation est déjà comprise dans les coefficients de valorisation des écrans
    - 2 - Pour les contrôles très complexes, on compte un nombre de « règles complexes » par tranche de 6 critères à prendre en compte.

# Méthode préconisée (6)

- Coefficients de complexité relative

- UO « Ecrans / Editions »

- Nb écrans de moins de 4 champs 1
    - Nb écrans de 4 à 7 champs 2
    - Nb écrans de 8 à 11 champs 3
    - Nb écrans de 12 à 15 champs 4
    - Nb écrans de plus de 16 champs 8

- UO « Echanges »

- Nb requêtes simples 1
    - Nb requêtes normales 2
    - Nb requêtes complexes 3

- UO « Règles de gestion »

- Nb règles de gestion de base (affichage) 0,1
    - Nb règles de gestion avec un paramètre 1
    - Nb règles de gestion de moins de 4 paramètres 2
    - Nb règles de gestion de 4 paramètres et plus 4

# Méthode préconisée (7)

- Coefficient de valorisation : permet de passer des UO aux jours de développement :

	Projet développement J2EE	Maintenance projet J2EE	Réalisation batch	Maintenance batch
UO Ecrans / Editions	0,6	0,3	0,3	0,2
UO Requêtes	0,5	0,3	0,5	0,3
UO Règles de gestion	0,5	0,4	0,4	0,2

- Attention, il ne s'agit que d'un exemple. Ces critères dépendent du contexte client (ils intègrent en partie la notion de risque)

# Exemple

Libellé	Qté	Poids	Coef.	Total
---------	-----	-------	-------	-------

## Ecrans / Editions

Nb écrans de moins de 4 champs	63	1	0,6	37,8
Nb écrans de 4 à 7 champs	40	2	0,6	48
Nb écrans de 8 à 11 champs	41	3	0,6	73,8
Nb écrans de 12 à 15 champs	26	4	0,6	62,4
Nb écrans de plus de 16 champs	87	8	0,6	417,6
<b>TOTAL charge IHM</b>				<b>639,6</b>

## Echanges

Nb requêtes simples	234	1	0,5	117
Nb requêtes normales	229	2	0,5	229
Nb requêtes complexes		3	0,5	0
<b>TOTAL charge requêtes</b>				<b>346</b>

## Règles de gestion

Nb règles de gestion de base (affichage)	2814	0,1	0,5	140,7
Nb règles de gestion (1 paramètre)	169	1	0,5	84,5
Nb règles de gestion (- de 4 paramètres)	100	2	0,5	100
Nb règles de gestion (4 paramètres et +)	35	4	0,5	70
<b>TOTAL charges règles de gestion</b>				<b>395,2</b>

<b>Total : Charges brutes de réalisation</b>				<b>1380,8</b>
--	--	--	--	---------------

# Exemple (suite)

Type de charge connexe	Description	Valeur (*)
Evaluation de projet	Prise en charge d'un besoin en amont du projet. Définition des impacts fonctionnels et applicatifs. Rédaction de l'analyse des besoins. Etablissement du chiffrage de réalisation (dans le cas de grosses évolutions)	5%
Pilotage de projet	Charge du chef de projet côté Maîtrise d'œuvre	15%
Assurance Qualité	Mise en place de PAQ et suivi des objectifs de qualité	3%
Intégration	Tests techniques de réception coté client, tests de non régression fonctionnelle	15%
Recette	VABF + VSR, sites pilote	15%
Documentation utilisateur	Guide d'utilisation	10%
<b>Total charges connexes</b>		<b>63%</b>

# Méthode préconisée : limites, conclusions

- Ce que ne gère pas cette méthode
  - la mise en place des éléments d'infrastructure
  - le développement de couches techniques génériques (API d'accès, frameworks, ...)
  - L'expertise technique sur des nouveaux produits
  - la définition d'architectures techniques
  - Le support (hotline, assistance sur site, astreintes, déplacements)
- Éléments compris mais non facturés (implicites)
  - :
  - Conseil (devoir de conseil)
  - Période de garantie
  - Suivi de la facturation et des aspects contractuels