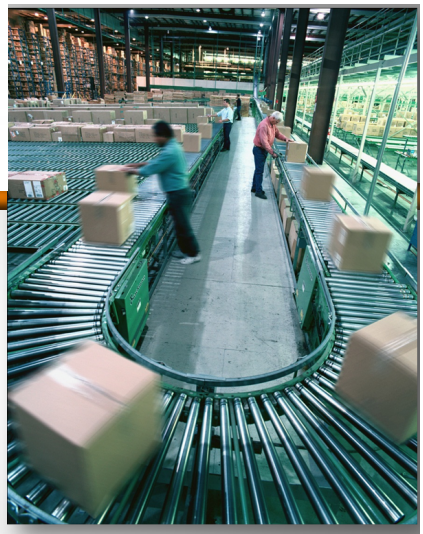


L'estimation des charges

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Mr Sellami K
Informatiqueii9@gmail.com

ESTIMATION DES CHARGES

Charge et durée

- Notions de base
  - La CHARGE représente une quantité de travail nécessaire, indépendamment du nombre de personnes.
    - Elle permet d 'obtenir un coût prévisionnel.
    - Elle s 'exprime en mois/homme.
    - Elle aide à définir la taille d 'un projet.
      - Projet < 6 m/H => très petit
      - 6 m/H < Projet < 12m/H => petit
      - 12m/H < Projet < 30m/H=> moyen
      - 30m/H < Projet < 100m/H=> grand
      - Projet > 100 m/H => très grand (année/homme).

ESTIMATION DES CHARGES

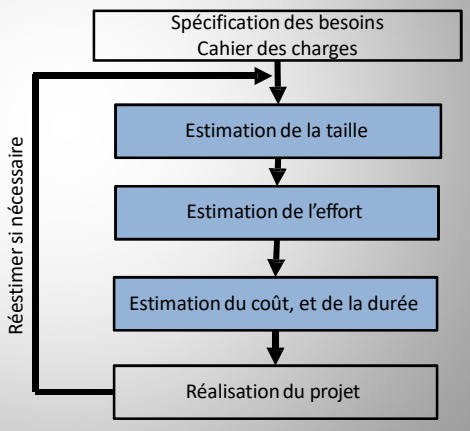
- Charge et durée
  - Notions de base
    - La DURÉE est le temps consommé par le projet.
    - Elle dépend du nombre de personnes, mais l 'évaluation n'est pas isotrope
      - (100 personnes pendant un mois ne sont pas équivalentes à 1 personne pendant 100 mois)

Les besoins en estimation

- L'estimation de l'effort (charge), des coûts et des durées est l'une des plus importantes activités d'un projet.
- Les besoins d'estimation se situent au niveaux: du *projet*, des *phases* et des *tâches*.
  - Au niveau du projet: Le MOE (fournisseur) et le MOA (client) font chacun une estimation de la durée et du coût du projet pour:
    - Déterminer une enveloppe budgétaire ;
    - Faire une estimation de la rentabilité de l'investissement ;
    - Évaluer une durée vraisemblable du projet.
  - Au niveau des phases et tâches: l'estimation est nécessaire pour :
    - Faire une planification précise ;
    - Annoncer un calendrier de remise des différents résultats intermédiaires;
    - Effectuer un suivi du projet pour surveiller les écarts ;
    - Prévoir l'affectation des ressources,

2.ETAPES D'ESTIMATION

- L'estimation d'un projet informatique comprend deux étapes :
  1. Estimation de la taille,
  2. Estimation de la charge(effort), du coût, et de la durée,



Processus d'estimation

2.ETAPES D'ESTIMATION

2.1. Estimation de la taille

- La taille d'un logiciel est la quantification des exigences fonctionnelles exprimées par les utilisateurs.
- S'appuie sur les spécifications des besoins et du cahier des charges exprimant les exigences du client.
- Elle peut être exprimée par plusieurs unités:
  - Points de fonction (Function Points: FP)
  - Lignes de code source (Line Of Code: LOC)
  - ..
- L'estimation de la taille constitue une base pour L'estimation de l'effort, du coût et de la durée du projet.

2.ETAPES D'ESTIMATION

2.2. Estimation de la charge, le durée et le coût

- Après avoir estimé la taille du logiciel à produire, on peut convertir la taille du logiciel en charge.
- La charge (effort) d'un logiciel est la quantité du travail nécessaire indépendamment du nombre de personnes qui vont réaliser ce travail.
  - Il s'exprime généralement en *Homme-mois*.
  - Un *Homme-mois* représente le travail d'une personne pendant un mois.
- Exemple:
  - Si 10 personnes ont travaillé pendant 5 mois dans un projet, l'effort de développement de ce projet est alors 50 Homme-mois.
  - Si on évalue le coût d'un Homme-mois à 20000 DA, alors le coût de développement sera estimé à 2 Million DA.

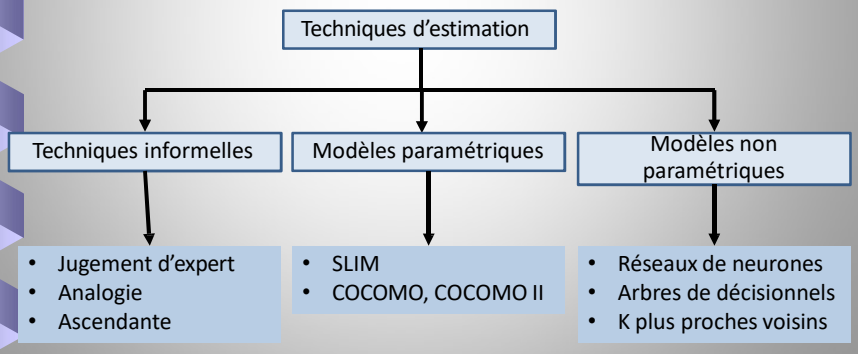
2.ETAPES D'ESTIMATION

Remarques importante

- Le coût total d'un projet logiciel comprend:
  - Le coût de développement du logiciel
  - Le coût du matériel
  - Le coût des déplacements…etc.
- Le coût de développement du logiciel correspond au temps passé à développer celui-ci par les développeurs dont on connaît les frais salariaux.
- Dans les projets logiciels, le coût de développement dépasse 80% du coût total du projet.

3. TECHNIQUES D'ESTIMATION

- **3 catégories des techniques d'estimation:** (1) Techniques informelles, (2) Modèles paramétriques, et (3) Modèles non paramétriques



3. TECHNIQUES D'ESTIMATION

1. Techniques informelles

- S'appuient sur l'expérience de l'historique des projets similaires,
- Conduites par une ou plusieurs personnes dites expertes dans le domaine de l'estimation.

- **Avantage**
  - Rapides et peuvent être utilisées assez tôt dans le cycle de développement logiciel.
- **Inconvénient**
  - Nécessite la disponibilité d'experts,
  - Très subjectives, et manquent d'argumentation analytique.

- Parmi ces techniques, nous pouvons citer: *le jugement d'expert*, *l'analogie* et *l'estimation ascendante,*

3. TECHNIQUES D'ESTIMATION

3.1.1. Jugement de l'expert

- Consiste à consulter un ou plusieurs experts qui utilisent leurs expériences ainsi que leur compréhension du projet afin de fournir une estimation à son coût.
- La ***méthode Delphi*** propose une démarche pour combiner les différents jugements d'experts :
  1) Chaque expert propose une estimation en utilisant sa propre expérience
  2) Tous les jugements son rendus publics, mais restent anonymes. Chaque expert peut alors modifier sa propre estimation ou la confirmer: Un expert doit se poser des questions si ses estimations sont très éloignées de celles des autres.
  3) Les estimations sont dévoilées et chacun peut justifier son propre jugement.
  4) Chacun propose une révision de son estimation.
  5) Ce processus est réitéré jusqu'à l'adoption d'une estimation par tout le groupe.

3. TECHNIQUES D'ESTIMATION

2. Analogie

- Si on dispose d'un projet similaire déjà achevé, dont on connait la taille, il est possible d'estimerez chaque partie principale du nouveau projet comme un pourcentage de la taille de la partie similaire du précédent projet.
- Avantage
  - Rapide à mettre en œuvre et peut être utilisée durant tout le cycle de vie du logiciel.
- Inconvénients
  - Nécessité d'avoir une historique sur lequel on peut se baser afin d'établir des analogies
  - Difficulté de trouver dans l'ensemble des tâches réalisées, une tâche suffisamment similaire à la tâche à estime

3. TECHNIQUES D'ESTIMATION

3. Estimation ascendante (bottom-up)

- Le projet logiciel est décomposé en plusieurs tâches constituant une arborescence (WBS).
- On estime l'effort nécessaire pour chaque tâche du plus bas niveau dans l'arborescence (généralement par jugement d'expert);
- On détermine progressivement l'effort des autres tâches, se retrouvant dans un niveau supérieur dans l'arborescence, en combinant les efforts nécessaires associés aux sous-tâches.
- L'effort de tout le projet est alors la somme de l'effort des estimations de chaque tâche du projet, éventuellement avec l'ajout d'une quantité d'effort pour couvrir les activités et les événements imprévus,

Inconvénient

- Le découpage du projet en tâches nécessite une connaissance détaillée du projet, elle ne peut en conséquence être appliquée que dans les phases aval du projet.

3. TECHNIQUES D'ESTIMATION

3.2. Modèles paramétriques

- Développés pour pallier aux inconvénients des techniques informelles se basant essentiellement sur la disponibilité des experts.
- S'appuient sur la modélisation statistique pour exprimer la relation qui existent entre l'effort et les variables considérées déterminantes de l'effort et appelées « facteurs d'effort » ou " « *cost drivers* ».
- L'élaboration des modèles d'estimation se fait deux étapes majeures:
  1. Identification des facteurs influençant l'effort de développement du logiciels « *cost drivers* ».:
     - La taille du projet logiciel est le facteur d'effort principal.
     - D'autres facteurs peuvent être prise en compte: Expérience du personnel (analystes, concepteurs, programmeurs, etc.), Technologie utilisée,
  2. Identification de la nature de la relation exprimant l'effort en fonction de ces facteurs.

3. TECHNIQUES D'ESTIMATION

2. Modèles paramétriques

– La relation entre l'effort et ses facteurs sera représentée par une fonction *f*:

$$Effort = f\ (X_1\ , X_2, \ldots, X_m)$$

Où $X_1\ , X_2, \ldots, X_m$ sont les facteurs affectant l'effort; f est la relation exprimant l'effort en fonction des facteurs X_i .

– L'élaboration de la fonction f est souvent basée sur l'analyse des données historiques collectées sur les projets logiciels déjà achevés.

– Ces données historiques peuvent être représentées par une matrice *N*x(*M*+1):

	Effort réel	X_1	X_2	…	X_M
Projet 1	E_1	x_{11}	x_{12}	…	x_{1M}
Projet 2	E_2	x_{21}	x_{22}	…	x_{2M}
Projet 3	E_3	x_{31}	x_{32}	…	x_{3M}
…	…	…	…	…	…
Projet N	E_N	x_{N1}	x_{N2}	…	x_{NM}

3. TECHNIQUES D'ESTIMATION

2. Modèles paramétriques

– *N* représente le nombre de projets logiciels déjà achevés et *M* le nombre de facteurs affectant l'effort.

– La colonne *Effort réel* ($E_1\ , E_2, \ldots, E_N$) représente l'effort réel de chaque projet logiciel déjà achevé.

	Effort réel	X_1	X_2	…	X_M
Projet 1	E_1	x_{11}	x_{12}	…	x_{1M}
Projet 2	E_2	x_{21}	x_{22}	…	x_{2M}
Projet 3	E_3	x_{31}	x_{32}	…	x_{3M}
…	…	…	…	…	…
Projet N	E_N	x_{N1}	x_{N2}	…	x_{NM}

– Le défi est de reconstruire cette relation f à partir de ces données afin qu'elle soit utilisée pour prédire le coût des nouveaux projets logiciels.

– Il existe plusieurs techniques qui permettent la reconstruction de la fonction *f* à partir d'un ensemble d'exemples.

3. TECHNIQUES D'ESTIMATION

3.2. Modèles paramétriques

- Le défi est de reconstruire cette relation ***f*** à partir de ces données afin qu'elle soit utilisée pour prédire le coût des nouveaux projets logiciels.

- Plusieurs techniques qui permettent la reconstruction de la fonction *f* à partir d'un ensemble d'exemples:

1. **Régression simple**
   - L'application de la régression linéaire simple, considère la taille du logiciel comme étant le facteur le plus significatif pour la prédiction de l'effort:

$$Effort = A + B \times taille$$

Où *A* et *B* sont des constantes.

   - La taille d'un logiciel est souvent mesurée par le nombre de lignes de son code source (Kilo Line Of Code -KLOC-) ou le nombre de points de fonctions.

3. TECHNIQUES D'ESTIMATION

3.2. Modèles paramétriques

3.2.2. Transformation logarithmique

- La transformation logarithmique est la plus utilisée en estimation des coûts du fait que les modèles adoptent souvent une équation centrale de la forme:

$$Effort = B \times taille^{\,C}$$

Pour rendre linéaire ce genre de modèles, on transforme les deux variables *effort* et *taille* en *log(effort)* et *log(taille)*. Ainsi, l'équation centrale transformée du modèle s'écrit comme suit:

$$\log Effort = \log B + C \times \log taille$$

On utilise ensuite la régression linéaire pour déterminer les deux constantes $\log B$ et C .

3. TECHNIQUES D'ESTIMATION

2. Modèles paramétriques

3. Régression linéaire multiple

– Dans le cas où le nombre de ces facteurs est supérieur ou égal à deux, la mise au point du modèle fait appel à la régression linéaire multiple.

– Des exemples de facteurs affectant le coût, autres que la taille, sont l'expérience du personnel impliqué dans le développement, la complexité de l'application et la méthodologie de développement.

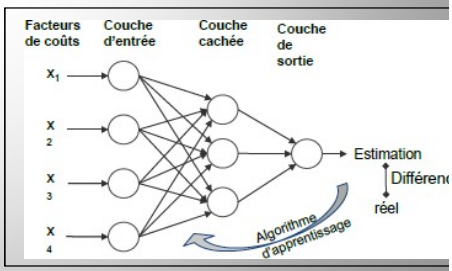
$$Effort = a_0 + a_1 x_1 + a_2 x_2 + \cdots a_m x_m$$

Où les a_i sont les facteurs affectant l'effort et les x_i sont des coefficients choisis pour fournir le meilleur ajustement à l'ensemble des données historiques observées.

3. TECHNIQUES D'ESTIMATION

3.3. Modèles non paramétriques

- Les modèles non paramétriques s'appuient sur moins d'hypothèses concernant la distribution des données et supposent que la forme de la fonction n'est pas définie *à priori*.
- Les modèles non paramétriques s'appuient aux algorithmes d'apprentissage automatique:
  - Les réseaux de neurones
  - Les arbres de décision,
  - La méthode des k plus poches voisins



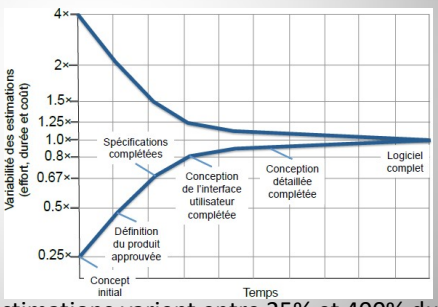
Structure d'un réseau neurones

4.INCERTITUDE DES ESTIMATIONS

- Les spécifications du projet peuvent changer au cours du projet ce qui rend nécessaire la **ré-estimation** de l'effort tout au long du cycle de vie du projet.
- Au début du projet les spécifications sont souvent incomplètes et imprécises, par conséquent, la marge d'erreur des estimations est considérable.
- Au fur et à mesure que les spécifications deviennent plus détaillée, ainsi, la marge d'erreur diminue et les estimations se rapprochent du coût réel.

4.INCERTITUDE DES ESTIMATIONS

- Boehm (1981) et McConnell (2006) illustrent l'évolution de l'erreur des estimations durant le cycle de vie du projet à travers le « Cône d'incertitude»:



- Au début du projet, les estimations varient entre 25% et 400% du couts réel du projet.
- À la phase de conception détaillée, les estimations passent au-dessous de 120% du couts réel.

5.LE MODELE COCOMO

- Le modèle COCOMO est le modèle le plus connu et le mieux documenté dans toute la littérature d'estimation des coûts.
- Il a été construit par Boehm en 1981 à partir d'une analyse des données par régression pratiquée sur 63 projets logiciels de 2.000 à 100.000 lignes de codes dans l'entreprise TRW Int.
- COCOMO est un acronyme pour COnstructive COst MOdel (Modèle de construction de coût).
- Ce modèle général se compose de trois modèles différents: le *modèle de base*, le *modèle intermédiaire*, et le *modèle détaillé.*
- Une nouvelle version du modèle appelé COCOMO II a été développé en 2000. COCOMO II est réputé être mieux adapté pour estimer des projets de développement de logiciels modernes.

5.LE MODELE COCOMO

- **5.1.Le modèle COCOMO de base :** Le modèle COCOMO s'appuis sur une estimation de la charge basée sur les formules

$$Charge = a \times KLOC^{b}$$
$$Durée = c \times Charge^{d}$$

- Charge (ou effort): est l'effort de développement exprimé en *homme-mois* (HM).
  - Un homme-mois correspond à 152 heures (19 jours) de travail effectif. Ce chiffre tient compte des absences pour formation, vacances, arrêts maladie…
- KLOC représente la taille du logiciel en milliers de lignes de code (instructions) (KLOC = Kilo Lines Of Code).
- *Durée* : est la durée de développement en *Mois*

5.LE MODELE COCOMO

1. Le modèle COCOMO de base

- Les paramètres *a*,*b*,*c* et *d* prennent des valeurs différentes selon le type du projet.
- Les 3 types de projets dans COCOMO sont:
  - **Simple** *(Organique)*: *projet qui* peut être réalisé par une équipe de petite taille (2 à 8 personnes), travaillant dans un domaine qu'ils connaissent (ex: petite gestion, système de notes dans une école, traducteurs).
  - **Moyen** *(Semi-détaché)*: projet qui présente un degré de difficulté moyen, l'équipe a une expérience limitée du type d'application (ex: Compilateurs, système bancaire interactif).
  - **Complexe** *(Intégré)* : projet complexe qui présente des contraintes fortes (contraintes de type temps réel, sécurité, support matériel et logiciel complexes). Le coût de changement d'une contrainte est très élevé. (Exemples: Gros système d'exploitation, système de contrôle aérospatial..)

Type de projet	*a*	*b*	*c*	*d*
Simple	2.4	1.05	2,5	0,38
Moyen	3.0	1.12	2,5	0,35
Complexe	3.6	1.20	2,5	0,32

5.LE MODELE COCOMO

5.1. Le modèle COCOMO de base

- A partir des valeurs obtenues pour la charge et la durée, on peut déduire :
  - **Le Staffing Moyen** : Le nombre de personnes requises pour réaliser le projet dans la durée estimée, exprimé en FSP (Full Time Equivalent Software Personnel):

Staffing Moyen =Charge/Durée

  - **La productivité moyenne** de l'équipe , exprimée en LOC/HM (lignes de code par homme-mois):

Productivité= Taille (LOC) /Charge

5.LE MODELE COCOMO

5.1. Le modèle COCOMO de base

- On peut ensuite calculer la distribution de la charge et de la durée de développement par phases (en %), selon le type et la taille du logiciel.

Type de projet	Phase	2 KLOC	8 KLOC	32 KLOC	128 KLOC	512 KLOC
Simple	Conception général	16	16	16	16	
	Conception détaillée	26	25	24	23	
	Programmation et tests unitaires	42	40	38	36	
	Intégration et test d'intégration	16	19	22	25	
Moyen	Conception général	17	17	17	17	17
	Conception détaillée	27	26	25	24	23
	Programmation et tests unitaires	37	35	33	31	29
	Intégration et test d'intégration	19	22	25	28	31
Complexe	Conception général	18	18	18	18	18
	Conception détaillée	28	27	26	25	24
	Programmation et tests unitaires	32	30	28	26	24
	Intégration et test d'intégration	22	25	28	31	34

Distribution de la charge par phase en pourcentage

5.LE MODELE COCOMO

5.1. Le modèle COCOMO de base

Type du projet	Phase	2 KLOC	8 KLOC	32 KLOC	128 KLOC	512 KLOC
Simple	Conception générale	19	19	19	19	
	Conception détaillée et Programmation	63	59	55	51	
	Tests et intégration	18	22	26	30	
Moyen	Conception générale	24	25	26	27	28
	Conception détaillée et Programmation	56	52	48	44	40
	Tests et intégration	20	23	26	29	32
Complexe	Conception générale	30	32	34	36	38
	Conception détaillée et Programmation	48	44	40	36	32
	Tests et intégration	22	24	26	28	30

Distribution de la durée par phase en pourcentage

5.LE MODELE COCOMO

5.1. Le modèle COCOMO de base

EXEMPLE : Un projet de type *simple* ayant une taille estimée à 32000 lignes de code.

- Mois

- Charge = 2.4 * $(32)^{1.05}$ = 91 HM (Homme-Mois)
- Durée = 2.5 * $(91)^{0.38}$ = 14 Mois
- Productivité= 32000 LOC/91 HM = 352 LOC/HM
- Staffing Moyen = 91 HM / 14 MOIS = 6.5 FSP
- **Distribution de la charge :**
  - Phase de conception générale : 0.16 * 91 = 14.6 HM
  - Phase de conception détaillée : 0.24 * 91 = 21.9 HM
  - Phase de programmation : 0.38* 91=34.6 HM
  - Phase d'intégration : 0.22*91= 20HM
- **Distribution de la durée :**
  - Phase de conception : 0.19*14=2.6 Mois
  - Phase de conception et de la programmation: 0.55 * 14 = 7.7 Mois
  - Phase d'intégration : 0.26 * 14 = 3.7 Mois

5.LE MODELE COCOMO

5.2. Le modèle COCOMO intermédiaire

- Le modèle COCOMO de base ne prend en compte que la taille et le type du logiciel lors de l'estimation de l'effort, toutefois, Il existe d'autres facteurs qui influencent l'effort.
- Le modèle COCOMO Intermédiaire est une extension du modèle COCOMO de base. Il prend en compte 15 facteurs, en plus de la taille et le type du logiciel
- Les équations de l'effort et de la durée du modèle COCOMO Intermédiaire:

$$Charge = a \times KLOC^{b} \times \prod_{i=1}^{15} C_i$$

$$Durée = c \times Charge^{d}$$

- Les paramètres *a*, *b*, *c*, et *d* du modèle COCOMO intermédiaire:

Type de projet	*a*	*b*	*c*	*d*
Simple	3,20	1,05	2,50	0,38
Moyen	3,00	1,12	2,50	0,35
Complexe	2,80	1,20	2,50	0,32

5.LE MODELE COCOMO

5.2. Le modèle COCOMO intermédiaire Les 15 facteurs d'effort C_i du modèle COCOMO intermédiaire:

Attributs du Produit
RELY (*Required Software Reliability*): Fiabilité requise
DATA (*Data Base Size*): Taille de la base de données
CPLX (*Product Complexity*): Complexité du logiciel
Attributs du Matériel
TIME (*Execution Time Contraint*): Contrainte du temps d'exécution
STOR (*Main storage Contraint*) Contrainte de la taille mémoire
VIRT (*Virtual machine Volatility*): Instabilité de la plateforme
TURN (*Computer Turnaround Time*): Temps de restitution de l'ordinateur
Attributs du Personnel
ACAP (*Analyst Capability*): Compétence des analystes
PCAP (*Programmer Capability*): Compétence des programmeurs
AEXP (*Application Experience*): Expérience du domaine d'application
VEXP (*Virtual Machine Experience*): Expérience dans la plateforme
LEXP (*Programming Language Experience*): Expérience du langage de programmation
Attributs du Projet
MODP (*Modern Programming Practices*): Pratiques des méthodes de programmation
TOOL (*Use of Software Tools*): Utilisation d'outils logiciels
SCED (*Required Development Schedule*): Contraintes de plannification

5.LE MODELE COCOMO

5.2. Le modèle COCOMO intermédiaire

- Chaque facteur est évalué avec une note, ensuite, la note est converti à une valeur.
- Les notes qu'un facteur d'effort peut prendre sont : **Très faible, Faible, Moyen, Élevé, Très élevé, et Extra-élevé**

Facteur	Evaluation					
	Très faible	Faible	Moyen	Elevé	Très élevé	Extra-élevé
RELY	0,75	0,88	1	1,15	1,4	
DATA		0,94	1	1,08	1,16	
CPLX	0,7	0,85	1	1,15	1,3	1,65
TIME			1	1,11	1,3	1,66
STOR			1	1,06	1,21	1,56
VIRT		0,87	1	1,15	1,3	
TURN		0,87	1	1,07	1,15	
ACAP	1,46	1,19	1	0,86	0,71	
AEXP	1,29	1,13	1	0,91	0,82	
PCAP	1,42	1,17	1	0,86	0,7	
VEXP	1,21	1,1	1	0,9		
LEXP	1,14	1,07	1	0,95		
MODP	1,24	1,1	1	0,91	0,82	
TOOL	1,24	1,1	1	0,91	0,83	
SCED	1,23	1,08	1	1,04	1,1	

5.LE MODELE COCOMO

5.2. Le modèle COCOMO intermédiaire

EXEMPLE : Considérons un projet de type *simple* ayant une taille estimée à 32000 lignes de code.

- **1ier Cas**: Tous les facteurs ont tous la valeur « Moyen »:
  Charge = 3.2 * $(32)^{1.05}$ *1= 122 H-M
- **2éme Cas**: La fiabilité est «Très élevée », et le reste des facteurs ont la valeur « Moyen » :
  Charge = 3.2 * $(32)^{1.05}$1.4 = 170.5 H-M
- **3éme Cas**: La fiabilité est «Très faible », et le reste des facteurs ont la valeur « Moyen » :
  Charge = 3.2 * $(32)^{1.05}$0.75 = 91.3 H-M