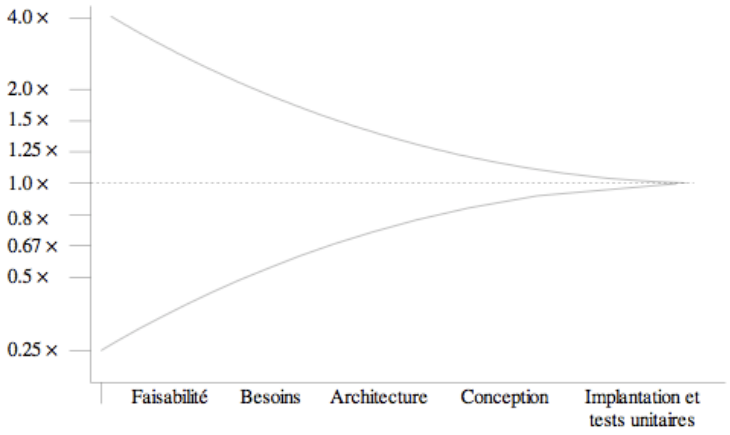
IFT3912 – Développement et maintenance de logiciels

Gestion de projet - Estimation

Bruno Dufour
Université de Montréal
dufour@iro.umontreal.ca

Incertitude et estimation



11

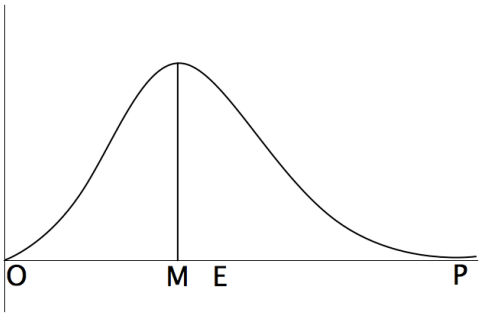
Estimation de la durée

- Estimation par analogie
  - Adaptation des estimations pour des projets similaires réalisés dans le passé
  - Au niveau du projet ou de parties du projet
- Estimation par étude de d'historiques
- Estimation par avis d'experts

12

Estimation de la durée

- Estimation par méthode des trois points
  - O: Estimé optimiste
  - P: Estimé pessimiste
  - M: Estimé le plus probable
  - E = (O + 4M + P) / 6



13

Estimation de la durée

- Méthode de Delphes
  - Consensus d'un groupe d'experts obtenu par itération

1. Chaque expert donne un estimé
2. Les résultats sont compilés
3. Les experts qui ont donné des estimés extrêmes doivent justifier leurs choix
4. Les étapes sont répétées jusqu'à consensus

  - Variation: chaque estimé utilise la méthode des trois points

14

Estimation de la taille

- *K(LOC)*
  - *Pas de standard*
  - *Dépend du langage utilisé*
- *Function points* (biaisés en faveur du traitement de données)
  - Entrées / sorties
  - Interactions avec l'usager
  - Interfaces externes
  - Fichiers utilisés

  - UFC = Σ_{type}(nombre d'éléments du type)(normalisation)

15

Estimation de la taille

- *Object Points*
  - Nombre d'écrans présentés
  - Nombre de rapports produits
  - Nombre de modules en langages de haut niveau

16

Estimations des coûts

- Plusieurs techniques possibles :
  - Estimation par analogie
  - Jugement d'expert(s)
  - Loi de Parkinson
    - La travail utilise tout le temps disponible
    - Par ex, un projet livrable dans 12 mois et disposant de 5 développeurs est estimé à 60 personnes-mois d'effort
  - Estimation pour obtenir le projet
    - Prix estimé pour emporter le développement du logiciel sur la concurrence
  - Modèles mathématiques / algorithmiques
    - ex. COCOMO (*COnstructive COst MOdel, Boehm 81)*

17

Modèles mathématiques

$$\text{Effort} = A \times \text{Taille}^B \times M$$

- A: constante
  - Type de logiciel
  - Organisation
- Taille: Taille du logiciel (peut être estimée)
  - (K)LOC
  - FP
  - OP
- B: entre 1 et 1.5
  - Reflète l'augmentation non-lineaire des coûts en fonction de la taille du projet
- M: multiplicateur

18

Constructive Cost Model (COCOMO)

- Développé par Barry Boehm
  - Première version en 1981
  - Révisé en 2000 pour l'adapter au développement moderne (p. ex., support pour la réutilisation de code)
- Modèle empirique basé sur des données provenant d'un grand nombre de projets

19

COCOMO 81 – de base

Nature du projet	Estimation de l'effort	Estimation de la durée
Simple (Organique)	PM = 2.4 (KLOC)$^{1.05}$	D = 2.5 (PM)$^{0.38}$
Modéré (Semi-détaché)	PM = 3.0 (KLOC)$^{1.12}$	D = 2.5 (PM)$^{0.35}$
Complexe (Embedded)	PM = 3.6 (KLOC)$^{1.20}$	D= 2.5 (PM)$^{0.32}$

KLOC: milliers de lignes de code
PM: effort en personnes-mois
D: durée en mois
P = PM/D: Nombres de personnes requises

20

COCOMO 81 – intermédiaire

- Ajoute un multiplicateur basé sur 4 « inducteurs de coûts »
  - Attributs du produit
  - Attributs du matériel
  - Attributs du personnel
  - Attributs du projet
- Ces facteurs peuvent faire varier la valeur de M à la hausse ou à la baisse (M = 1 dans le modèle de base)

Nature du projet	Estimation de l'effort
Simple (Organique)	PM = 3.2 (KLOC)$^{1.05}$ x M
Modéré (Semi-détaché)	PM = 3.0 (KLOC)$^{1.12}$ x M
Complexe (Embedded)	PM = 2.8 (KLOC)$^{1.20}$ x M

21

COCOMO 81 – détaillé

- Prend en compte les phases du développement
- Basé sur le processus de développement en cascade
  - Ce processus était populaire en 1981
- Ne correspond pas vraiment à la réalité du développement moderne
  - COCOMO de base et intermédiaire peuvent être utilisés comme estimés simples et rapides

22

COCOMO II

- Ajoute le support pour la réutilisation de code
- Trois niveaux de formules, en fonction de la quantité d'information connue :
  - *Application-composition model*
    - Utilisé pour les prototypes ou pour les projets développés par composition de composants existants
  - *Early design model*
    - Utilisé avant la phase de conception détaillée
  - *Post-architecture model*
    - Utilisé lorsque l'architecture du logiciel développé est connue

23

COCOMO II – Application Composition Model

$$PM = OP \times (1 - \%reutilisation) / PROD$$

- PM: efforts en personnes-mois
- OP: *Object Points*
- $0 \leq \%reutilisation < 1$
- PROD : facteur de productivité
  - 4-50 d'après l'expérience des développeurs et la maturité du processus

24

COCOMO II – Early Design Model

$$PM = A \times Taille^{B} \times M$$

- PM: efforts en personnes-mois
- A = 2.94
- Taille: en KSLOC
- B: varie de 1.01 à 1.26 en fonction de plusieurs facteurs
- M: un facteur d'ajustement qui dépend de 7 attributs du projet et du processus

25

COCOMO II – Calcul de l'exposant (B)

Facteur	Explication
Familiarité	Reflète l'expérience dans ce type de projet
Flexibilité	Reflète de degré de flexibilité dans le processus de développement (haut = le client ne spécifie que des buts généraux, bas = un processus prescrit est suivi)
Résolution des risques	Reflète le degré d'analyse de risques effectuée.
Cohésion de l'équipe	Reflète la facilité des membres de l'équipe à travailler ensemble.
Maturité du processus	Reflète la maturité du processus dans l'entreprise (calculé à partir d'un questionnaire, mais peut être estimé).

- Chaque facteur est évalué sur une échelle de très bas (5) à extrêmement élevé (0)
- B = (somme des scores) / 100 + 1.01

26

COCOMO II - Multiplicateurs

- Les multiplicateurs reflètent les aptitudes des développeurs, les besoins non-fonctinnels, etc.
  - RCPX - product reliability and complexity
  - RUSE - the reuse required
  - PDIF - platform difficulty
  - PREX - personnel experience
  - PERS - personnel capability
  - SCED - required schedule
  - FCIL - team support facilities
- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED

27

COCOMO II – Génération de code

$$PM_{auto} = (ASLOC \times AT\%) / ATPROD$$

- ASLOC: nombre de lignes de code réutilisées ou générées automatiquement
- AT%: pourcentage du code réutilisé qui est généré automatiquement
- ATPROD: productivité à intégrer le code généré (=~2400 lignes / mois)

28

COCOMO II – Réutilisation

$$ESLOC = ASLOC \times (1 - AT\%) \times AAM$$

- ESLOC: nombre de lignes de nouveau code dues à la réutilisation
- AAM: représente la quantité de travail requise pour la réutilisation de code
  - Coût des changements au code réutilisé
  - Coût de compréhension du code réutilisé (varie de 10 à 50)
  - Coût associé à décider si un composant peut être réutilisé (varie de 0 à 8)

29

COCOMO II – Post-architecture model

$$PM = A \times Taille^{B} \times M$$

- Même formule que précédemment
- Utilise 17 types de multiplicateurs plutôt que 7
- La taille du code est estimée en utilisant :
  - Le nombre de lignes de code à développer (SLOC)
  - Le nombre de lignes de code à changer dû à la réutilisation (ESLOC)
  - Le nombre de lignes de code qui sont susceptibles d'être modifiées suite à des changements dans les besoins du système

30

COCOMO II – Estimation du calendrier

$$TDEV = 3 \times PM^{(0.33 + 0.2(B - 1.01))}$$

- TDEV: nombre de mois de développement
  - NB: TDEV est indépendant de la quantité de personnel
- PM: effort calculé à l'aide du modèle COCOMO
- B: exposant

31

Estimation du personnel

- La quantité de personnel requis ne peut pas être calculée en divisant le temps de développement par l'échéancier
  - Le nombre de participants à un projet varie selon la phase de développement
  - L'effort requis augmente généralement avec la taille de l'équipe
  - Ajouter du personnel réduit la productivité de l'équipe existante (communication, interfaces, etc.)
    - Ajouter une personne apporte un gain net de < 1 personne au projet

32