Square des Utilisateurs

Estimation des charges d'un projet xNet

Méthodes d'estimation dans le cadre d'un projet xNet

D'après une thèse professionnelle réalisée à la DSI de la Caisse des Dépôts et Consignations (Établissement Public), dans le cadre de la formation du Mastère Spécialisé « Management des Systèmes d'Information et des Technologies », conduite conjointement par l'École des Hautes Études Commerciales (HEC) et l'École Nationale Supérieure des Mines de Paris (ENSMP), sous la supervision d'Alain Berdugo (Professeur à HEC, Directeur du Mastère MSIT) et d'André Probst (Responsable Méthodes de la DSI CDC - Établissement Public)

Thèse consultable dans son intégralité sur :
http://www.gezzed.net/these/These-MethodesEvaluationsChargesXnet.pdf

Introduction

Lorsque l'on parle d'« Estimation des charges » de projets informatiques à une Maîtrise d'œuvre ou à une Maîtrise d'ouvrage, chacun a déjà plus ou moins une idée sur la question… sa propre méthode qu'il utilise, ou une vieille formule gribouillée sur une feuille (dont, d'ailleurs, on ne sait plus très bien d'où elle vient), mais qui a fait ses preuves, et que l'on continue à utiliser… ou alors, la bonne vieille méthode du « nez » du chef de projet, que l'on retrouve tout aussi fréquemment lors de l'estimation de charges de certains projets.

Évidemment, une entreprise qui a déjà fait un certain nombre de projets informatiques, connaît plus ou moins les charges et les délais dont elle aura besoin pour faire un projet, dont la plate-forme technique et les règles fonctionnelles lui sont familières. Ces « méthodes » d'une efficacité relative, n'en restent pas moins intuitives, et par là, perdent une certaine crédibilité lors d'une justification officielle de l'estimation.

Il existe pourtant des méthodes reconnues et efficientes d'estimation de charges de projet informatiques. Ces méthodes ont eu le temps de faire leur preuve dans divers domaines. La plupart d'entre elles se basent sur l'expérience d'un certain nombre de projets-types analysés.

Jusqu'à présent, ces méthodes étaient relativement fiables, vis-à-vis des projets informatiques « classiques », qui pouvaient se faire jusqu'au milieu des années 1990 – en l'occurrence, les applications monopostes, ou en client-serveur.

Mais depuis moins d'une dizaine d'années, et avec l'évolution exponentielle des technologies et l'apparition des xNet, les règles du jeu ont changé, une fois encore…. Toutes les entreprises ont aujourd'hui leur internet, tous les grands comptes, leur intranet, et la majorité de commerces, leur extranet.

La mesure des charges de ces xNet est une « science » bien trop récente pour pouvoir en parler avec certitude, et sans risquer de se tromper. Mais, cette mesure devient aujourd'hui plus qu'indispensable, et l'utilisation d'une méthode sûre, et reconnue dans le cadre des xNet, pourrait aider à poser des bases concrètes de comparaison, pour les maîtres d'ouvrage et les maîtres d'œuvre et, pourquoi pas, les SSII.

L'évaluation de charges d'un projet Informatique

Pourquoi évaluer les charges ?

Dans le cadre des projets de la nouvelle économie, le concept de l'xNet est devenu quasi-incontournable : toutes les grandes entreprises possèdent aujourd'hui un intranet, et la conception (et le développement) de modules s'y greffant est le lot quotidien des responsables informatiques de ces entreprises.

Les diverses étapes de spécification, conception, recette, etc., prennent du temps, aussi bien du côté de la maîtrise d'œuvre que celui de la maîtrise d'ouvrage, et même de l'exploitation… et parfois beaucoup plus (ou beaucoup moins) que les métiers et la maîtrise d'ouvrage ne pourraient l'estimer.

*Le concept de « Jour*Homme »*

Le « Jour*Homme » est l'unité de mesure de la charge de travail dans le contexte d'un projet.

Concrètement, un projet estimé à « cent jours pour une personne à plein temps » pour arriver à son terme, on pourra considérer qu'il s'agit d'un projet estimé à « 100 jours hommes ». Cependant, nous pouvons nous permettre de supposer que tout projet, à un certain niveau, est « scindable » : c'est-à-dire que l'on va pouvoir le décomposer en un ensemble de tâches.

Si ce projet est partitionnable (et ce sera quasiment toujours le cas), il est raisonnablement possible d'effectuer plusieurs tâches en parallèle – donc de les faire effectuer par plusieurs hommes simultanément.

Partant de ce postulat, et dans un cadre idéal, on peut donc considérer qu'un projet de 36 J*H (planifié pour 12 hommes sur 3 jours) pourra être effectué en 9 jours par 4 hommes.

Cette répartition idéale suit une parabole. Nous verrons cependant un peu plus tard que cette théorie est loin d'être exacte dans le domaine qui nous intéresse, c'est-à-dire celui des projets informatiques de la nouvelle économie.

Les pièges de l'évaluation théorique

Il existe plusieurs pièges liés au fort désir des hommes à vouloir mettre au préalable une durée sur une tâche.

Par exemple, une des lois de C.N.Parkinson stipule que « les programmes sont comme les gaz parfaits : ils prennent toute la place qu'on leur donne ». L'interprétation de ce postulat est extrêmement simple : si l'on donne à une équipe un projet de 50 Jours*Homme (valeur « réelle ») à faire sur 100 Jours*Homme, le projet sera fait en 100 Jours*Homme, et pas un de moins. Cela fonctionne également dans l'autre sens : il est possible de réduire sensiblement la durée théorique d'un projet, et obtenir un résultat dans le délai escompté. Ce n'est cependant pas une méthode recommandée par la plupart des manuels de gestion de projet, et par les maîtres d'œuvre, eux-mêmes.

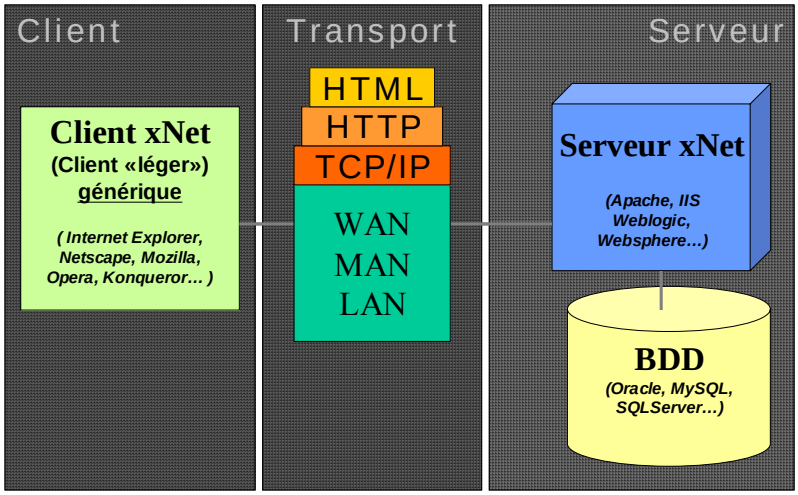
De plus, l'unité utilisée pour effectuer l'estimation de la durée est d'une fiabilité toute relative : reprenons l'exemple du projet de 100 J*H : selon la théorie, ce projet pourrait être effectué par 200 Hommes en une demi-journée. Bien que ce soit mathématiquement exact, c'est une aberration évidente : Les projets de l'informatique, en général, ne peuvent pas êtres scindés ainsi.

À l'extrême opposé, il existe également des projets qui ne sont pas partitionnables.

L'exemple revenant le plus souvent est celui –il est vrai, assez simpliste, mais très illustratif- de la femme enceinte, qui va mettre neuf mois à mettre au monde un enfant… Il est, de façon évidente impossible de partitionner, en plus d'une tâche. – Même trois femmes dotées de la meilleure volonté du monde ne pourront faire un enfant en trois mois.

Il faudra donc faire preuve de prudence lorsque l'on présentera une estimation de charge d'un projet, en tenant compte de tous les pièges tendus, amenés par le désir (bien compréhensible) des maîtres d'œuvre de mettre une durée sur une tâche ou un projet.

Le xNet



Le « xNet » (ou *Net) est une appellation générique du Système d'Information de front-office de l'entreprise. Il peut désigner un Internet, un Extranet, ou encore un Intranet, voire une combinaison de ces modèles de réseaux.

La différence fondamentale entre ces Internet, Extranet, et Intranet, est principalement liée au public (aux « visiteurs ») visé par ces réseaux, et par voie de conséquence au type d'accès de ces utilisateurs à ce que nous appellerons le « moteur de distribution de contenu ».

La plupart du temps, la technologie utilisée dans les xNet est la suivante :

Un serveur HTTP fournissant du contenu HTML, indifféremment statique ou dynamique, et interactif. À cela s'ajoutent divers services tournant derrière les serveurs HTTP, tels que des serveurs de données, ou de fichiers, et plus généralement quasiment n'importe quelle application pouvant potentiellement être assurée en back-office par le système d'information.

La frontière entre les trois concepts d'Intranet, d'Internet et d'Extranet s'avérant parfois extrêmement ténue, certains professionnels préfèrent utiliser le terme d'"Application xNet" pour désigner un système de « site » interactif mêlant indifféremment un ou plusieurs des trois concepts.

En général, on aura d'un côté, un ou plusieurs « clients légers » sur lesquels s'affichent une interface utilisateur, très proche du principe des interfaces standard (Windows ou MacOS), et dans lesquels le client verra s'afficher des objets graphiques lui permettant d'effectuer des opérations. Le résultat des opérations effectuées s'affichera de même dans cette fenêtre de « client léger"

De l'autre côté, un ou plusieurs « serveurs », qui auront pour tâche de récupérer les ordres envoyés par le client, de les traiter, et de renvoyer au « client léger" le résultat des opérations.

- Un site dit « Internet » aura pour vocation de toucher toutes les personnes capables de s'y connecter physiquement.
- Un site « Intranet »" est lié à un accès extrêmement restreint : le nombre de visiteurs potentiels est parfaitement connu, ainsi que l'identité de ces derniers. Ce type de site requiert en général des technologies dynamiques, et apporte des services conséquents à ses visiteurs, dans le cadre, par exemple, d'une organisation.

- Un site « Extranet », plus restreint qu'un site Internet, sera destiné à un public possédant un mot de passe, ou une clef permettant de consulter ses données. Le plus souvent, ce type de site est dynamique, et les visiteurs sont a priori connus, ou, en tout cas, répertoriés.

Caractéristiques intrinsèques

Grâce au modèle HTML et aux liens hypertextes (inventés par Théodore Nelson en 1965), on peut, de façon extrêmement simple, modéliser n'importe quel xNet sous forme d'un ensemble de « pages » liées entre elles par ces liens. On trouve par ailleurs souvent dans les xNet un module de navigation (intimement lié à un module d'identification), permettant au xNaute de naviguer suivant leur profil.

Suivant le type de xNet visité, et suivant le contexte dans lequel on se place, tous les utilisateurs n'auront pas accès de la même façon au site… (ces « droits d'accès » ayant une répercussion directe sur le module de navigation). Ceci est parfaitement compréhensible si l'on considère que derrière chaque page visitée, il y a une information (pouvant être à caractère privé), ou un traitement (pouvant être à caractère sensible) qui peut être uniquement dédié à un groupe d'utilisateurs, voire à un seul utilisateur.

Exemples : authentification d'un site Web d'une banque (Extranet), administration distante d'un site, etc.

Gardons en tête qu'un xNet n'est potentiellement pas un simple « site web ». C'est une application à part entière : en plus de l'interface utilisateur, chaque xNet possède son traitement et ses données.

La partie « visible » du xNet, l'interface, est en fait un contenu dynamique affiché sur un client… ce contenu est le résultat d'une suite d'opérations effectuées sur un ou plusieurs serveurs distants, grâce à des données situées elles aussi, sur un ou plusieurs serveurs distants. La seule différence significative par rapport au Client-Serveur, est que l'interface client est normalisée, et que les transactions liant l'interface aux traitements le sont aussi.

Il est donc tout à fait légitime, dans une certaine mesure, de considérer le xNet comme une application Client-Serveur, et – en terme de charges - de l'évaluer de façon similaire.

Les différentes méthodes que nous allons analyser ne prendront pas en compte le contenu textuel statique du site, qui devra faire l'objet d'une spécification à part, du côté de l'utilisateur.

État de l'art des méthodes d'évaluations

Méthode des points de Fonctions

La méthode des Points de Fonctions est destinée à évaluer la taille d'un projet indépendamment de la technologie utilisée pour le réaliser. Son avantage réside principalement dans le fait qu'il est possible d'effectuer une évaluation grâce à cette méthode très en amont.

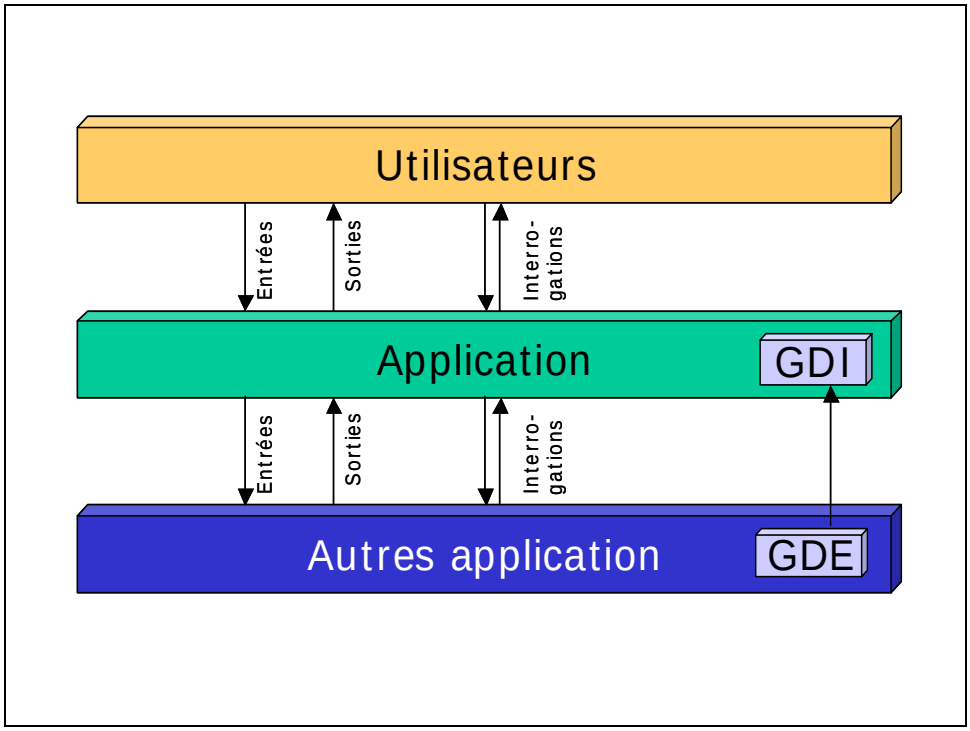
On s'affranchit, de plus, du comptage de lignes de code – comme pour la méthode COCOMO, qui devient d'une fiabilité tout à fait relative en fonction du langage utilisé… Ceci étant d'autant plus vrai dans le domaine des xNet ou une grosse partie des lignes de « code" définit l'interface.

Les points de fonctions reposaient, à l'origine, sur un calcul basé sur quatre entités (entrée, sortie, interrogation, fichiers), et ce, sans catégorisation de complexité. (Albrecht, 1979).

Depuis le milieu des années 80, avec l'IFPUG (International Function Points User Group), et la normalisation AFNOR, le comptage des points de fonctions se fait à partir des entités suivantes :

Données internes (GDI) : Groupe de données logiquement liées, ou de groupe de paramètres de contrôle, et identifiables par l'utilisateur. Ces données sont mises à jour et utilisés à l'intérieur de la frontière de l'application. Les entités reliées par une cardinalité (1,1) forment un seul et même GDI.

Données externes (GDE) : Groupe de données logiquement liées, ou groupe de paramètre de contrôle, et identifiables par l'utilisateur. Ces données sont utilisées par l'application, mais mises à jour par une autre application. Le GDE est un GDI dans un autre domaine.



Entrées (ENT) : Données, ou les paramètres de contrôle, qui entrent dans l'application considérée. Ces entrées maintiennent un ou plusieurs GDI, initialisent ou contrôlent un traitement, et font l'objet d'un traitement unique. Une Entrée correspond donc à un écran de saisie, ou à une réception de données. A chaque GDI doit correspondre au moins une entrée, permettant sa mise à jour.

Sorties (SOR) : Données, ou les paramètres de contrôle qui sortent de l'application. Ces sorties sont le résultat d'un traitement unique (différent d'une simple extraction de données). Ce peut être un écran de visualisation, ou un message vers une autre application.

Interrogations (INT) : Ce sont des données élémentaires qui correspondent à une extraction de données. L'INT ne met à jour aucun GDI.

On associe à ces cinq entités des paramètres supplémentaires, qui vont permettre de déterminer par la suite le niveau de complexité de chaque élément.

Données élémentaires (DE) : Chaque GDI ou GDE est composé de données élémentaires. Une DE équivaut à un champ de données. On compte un seul DE par champ répétitif dans les entrées, les sorties, et les interrogations.

Sous-ensemble logique de données (SLD) : Groupements logiques de GDI ou de GDE qui sont traitées simultanément dans l'application.

Groupe de données référencées (GDR)

Groupements logiques de GDI ou de GDE qui sont mis à jour, ou consultés simultanément par les différentes ENT, SOR ou INT.

Points de fonctions				
		1 à 19 DE	20 à 50 DE	> 50 DE
GDI	1 SLD	7	7	10
	2 à 5 SLD	7	10	15
	>5	10	15	15
		1 à 19 DE	20 à 50 DE	> 50 DE
GDE	1 SLD	5	5	7
	2 à 5 SLD	5	7	10
	>5	7	10	10
		1 à 4 DE	5 à 15 DE	> 10 DE
ENT	0 ou 1 GDR	3	3	4
	2 GDR	3	4	6
	> 2 GDR	4	6	6
		1 à 5 DE	6 à 19 DE	> 19 DE
SOR	0 ou 1 GDR	4	4	5
	2 ou 3 GDR	4	5	7
	> 3 GDR	5	7	7
		1 à 5 DE	6 à 19 DE	> 19 DE
INT	0 ou 1 GDR	3	3	4
	2 ou 3 GDR	3	4	6
	> 3 GDR	4	6	6

Les points de fonctions sont calculés en fonction du tableau ci-contre, selon le nombre d'entités et de leur complexité.

L'étape finale consistera à ajuster ces points de fonctions grâce à un ensemble de 14 degrés d'influence. (Communication, distribution des données ou des traitements, performance, intensité d'utilisation de la configuration matérielle, taux de transition, de transaction, saisie interactive, convivialité, mise à jour en temps réel des GDI, complexité des traitements, réutilisation, facilité d'installation, facilité d'exploitation, portabilité, facilité d'adaptation).

Ce Facteur d'Ajustement va nous permettre de corriger le nombre de points de fonctions bruts avec un facteur allant de 65% à 135% (donc avec un résultat allant potentiellement du simple au double).

Par la suite, la charge se calcule grâce à un facteur multiplicatif, d'environ 3 Jours*Homme par Point de Fonction en moyenne (2 J*H si le projet est petit, 4 s'il s'agit d'un grand projet)

À la fin de l'étude détaillée, ce facteur peut varier de 0,1 (pour les projets en L4G, dans le meilleur des cas) jusqu'à 2 pour les projets plus complexes, en passant par un facteur 0,5 pour les projets de type RAD, où la productivité est plus élevée.

Pour synthétiser, le comptage subjectif, difficile à automatiser, et même si l'estimation de l'effort est juste, l'estimation de la charge (sujette à une multiplication par une valeur subjective) est d'une exactitude toute relative. Cependant, retenons que l'estimation est d'une part disponible assez tôt dans le projet, et surtout, est indépendante du langage, de la plate-forme, et des diverses technologies utilisées.

On peut, avec l'expérience et les statistiques, adapter facilement le modèle à l'estimation des xNet.

Méthode COCOMO (Constructive Cost Model)

« Pour estimer la charge d'un projet, il faut en estimer la taille ». La principale hypothèse de cette méthode est qu'un informaticien saura mieux évaluer la « taille » d'un projet informatique que l' « effort » à fournir pour le développer.

COCOMO a été défini par B.W.Boehm sous la forme d'équations simples, admettant quatre paramètres.

Ces divers coefficients ont été trouvés de façon expérimentale, Boehm se basant sur l'estimation et la réalisation de plus d'une soixantaine de projets divers d'informatique classique. (chaque projet ayant une taille située entre 2000 et 100.000 lignes de code) La probabilité d'obtenir un résultat proche de la réalité grâce à COCOMO dépend du fait que le projet que l'on analyse est proche ou non du panel de projets analysés par Boehm.

Le paramètre principal est la « ligne de code ». Le nombre de milliers de ligne de code représente la « taille » du projet. Selon N.E.Fenton (Software Metrics: A Rigorous and Practical Data, 1998) : « Une ligne de code est toute ligne du texte d'un programme qui n'est pas une ligne de commentaire, ou une ligne blanche, sans considération du nombre d'instructions ou de fragments d'instructions dans la ligne. Sont incluses toutes les lignes contenant des en-têtes de programmes, des déclarations, et des instructions exécutables et non exécutables. ».

Cependant, dans le contexte des xNet, contrairement aux langages modernes, orientés GUI, on est en droit de se demander si le concept de « nombre de lignes de code » reste fiable, en raison de la nécessité d'utilisation d'un grand nombre d'entre elles pour définir uniquement l'interface de l'application xNet. (C'est typiquement le cas du HTML).

La taille d'un projet est estimée en milliers de lignes de code. On parle généralement de KDSI (Kilo Delivered Source Instruction), ou de KLOC. (Kilo Lines Of Code)

Il est possible, théoriquement, d'évaluer la taille (le nombre de KDSI) en fonction du nombre de points de fonctions du projet, grâce à la formule T = 0.1187.PFA-6.49.

Une autre méthode d'estimation de T peut être déterminée grâce aux facteurs suivants, en fonction du langage utilisé : (Assembleur : 0,32 KDSI par Point de Fonction, C : 0,15 ; COBOL : 0,106 ; Pascal : 0,091 ; Prolog : 0,064 ; Basic : 0,064 ; SQL : 0,040 ; Smalltalk : 0,021)

La première chose à faire lorsque l'on décide d'utiliser COCOMO pour évaluer le projet, est donc de déterminer à quel « type » de projet l'on a affaire. Boehm propose trois types de projet:

Projet Organique : Projet simple, ou de routine, effectué par une équipe ayant déjà travaillé ensemble, dans lequel il y a peu de « surprises » et où une bonne anticipation est possible.

Projet Semi-Détaché : Entre organique et imbriqué. Le Projet n'est ni trop simple, ni trop complexe, l'équipe de développement se connaît un peu, et les technologies peuvent être mal connues, mais pas d'une grande difficulté d'appréhension.

Projet Imbriqué : Techniques innovantes, organisation complexe, beaucoup d'interactions. Projet difficile, ou dans un domaine inconnu par l'entreprise, équipe de développement n'ayant pas encore travaillé ensemble, ou projet impliquant des technologies encore peu connues des développeurs.

Les calculs se font grâce aux équations suivantes :

COCOMO Intermédiaire	a	b	c	d
Organique	3,2	1,05	2,5	0,38
Semi-détaché	3,0	1,12	2,5	0,35
Imbriqué	2,8	1,20	2,5	0,32

$$\textbf{Effort} = 21 * a * T^{b} * \textbf{Facteur d'Ajustement}$$

$$\textbf{Durée} = 21 * c * (\textbf{Effort}/21)^{d}$$

$$\textbf{Effectifs} = \textbf{Effort}/\textbf{Durée}$$

Le facteur d'ajustement total est égal au produit de 15 facteurs d'ajustement, en quatre catégories :

- Attributs du produit (Fiabilité Requise, Taille de la Base de Données, Complexité du produit)
- Attributs du Matériel (Contraintes de temps d'exécution, Contraintes de taille mémoire principale, Instabilité de la Machine Virtuelle, Temps de Retournement)
- Attributs de l'équipe (Compétence des Analystes, Expérience du domaine d'application, Compétence des Programmeurs, Expérience de la Machine Virtuelle, Expérience du langage)
- Méthodes et Outils (Pratique des Méthodes, Utilisation d'outils logiciels, Contraintes de planning)

Notons que le facteur d'ajustement peut varier de 0.09 à 72.38 (un rapport de plus de 800 !!!), sa valeur étant égale à 1 pour un projet normal.

Pour synthétiser : il est assez ardu de calculer de façon fiable la « taille » du projet (en KDSI) lorsque l'on est peu avancé dans le projet, et la fiabilité du résultat obtenu dépend énormément du fait que le projet à réaliser se rapproche fortement de ceux qui ont servi à créer le modèle.

Le fait d'utiliser la « ligne de code" en tant qu'unité de calcul amène quelques failles dans le système : on ne tient pas compte, par exemple, des commentaires dans le code (Qui, pour la maintenance ou le déboguage s'avèrent primordiaux).

COCOMO présente toutefois l'avantage d'être transparent : il est facile d'ajuster ses paramètres et de recréer son propre modèle.

ÉVALUATEUR-RAD (J-P.Vickoff)

ÉVALUATEUR-RAD est un logiciel de Jean-Pierre Vickoff qui se base, selon l'auteur, sur une évolution des travaux de Albrecht, Ashby, Boehm, Clark, Egyed, Gacek, Hoh, Lehman, Martin, Maxwell, Pareto, Parkinson, Rayleigh et Shannon… autant dire une combinaison de la plupart des méthodes d'évaluation et de statistiques du domaines. Ce n'est donc pas, à proprement parler, une méthode. Cependant, ce logiciel (disponible sur http://mapage.noos.fr/rad/eval2000.htm) reste néanmoins très intéressant dans le contexte qui nous intéresse, dans la mesure où Évaluateur-RAD prend en compte plusieurs paramètres, dont la nature a totalement été occultée dans les autres méthodes analysées ici. De plus, il s'agit bel et bien du produit la plus « adapté » aux xNet, les autres méthodes répondant aux demandes d'évaluation, pour la plupart, du client-serveur, ou bien de l'informatique classique.

Après plusieurs tests approfondis du dit logiciel, plusieurs personnes estiment que la charge jaugée en fin de remplissage des écrans était légèrement supérieure à la réalisation réelle.

Nous vous conseillons d'essayer vous-même ce logiciel. Son utilisation est intuitive, et on obtient un résultat rapidement.

Cependant, il semble, au final, donner des chiffres trop élevés pour des petits projets. De plus, certains coefficients de pondération sont difficilement explicables ou justifiables (ex: taille de l'écran de l'application, etc...), qui amènent, par exemple, pour un projet nominal de 100 Jours*Homme, une variation de 8% à plus de 35 000% de la taille prévue.

Méthode Interne CDC (A.Probst)

Cette méthode a été mise au point par André Probst, Responsable Méthodes de la Direction des Systèmes d'Information de la Caisse des Dépôts et Consignations. La méthode correspond, à l'origine, à l'évaluation de projets de type client-serveur.

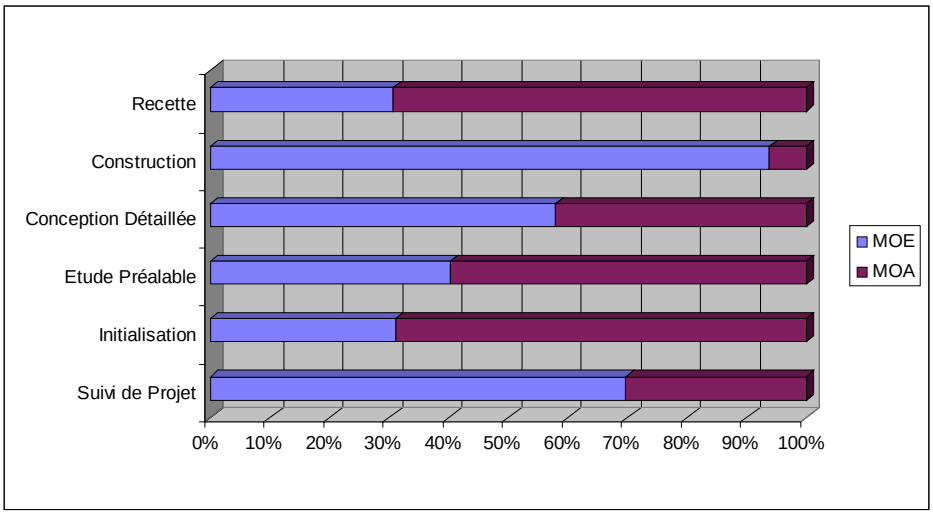
Les coefficients ont été obtenus grâce à l'expérience, sur des essais fait sur plus d'une dizaine de gros projets de la Caisse des Dépôts et Consignations. Cette méthode a l'avantage d'évaluer à la fois, la charge nécessaire en terme de maîtrise d'œuvre et d'ouvrage, en rentrant dans tous les détails de la construction de l'application.

La première étape consiste à décomposer le projet en macro-fonctions. Chacune de ces macro-fonctions correspond à une fonctionnalité globale de l'application. Par la suite, il s'agira de décomposer chaque macro-fonction en un ensemble de micro-fonctions, ces dernières étant pondérées par leur niveau de complexité que leur attribue le chef de projet grâce à sa propre expérience.

Puis il convient de calculer un « Nombre d'Objets/Relations ». Sa valeur correspond, dans le MCD, à la somme du nombre de tables « significatives » du modèle de données, et du nombre de relations les liant.

On obtient donc dans un premier temps un « Effort » (il s'agit de la métrique de référence) à fournir dans le cadre de la réalisation d'un projet, à partir d'une étape avancée dans le temps, correspondant approximativement au milieu de l'étude préalable. (La méthode d'origine permettant de calculer les charges en fonction de ces informations était basée sur une durée de projet, évaluée à partir de cette période).

Cet effort est décomposé grâce à un certain nombre de coefficients en un certain nombre de charges.



La répartition des charges du projet se fait pour la maîtrise d'œuvre et la maîtrise d'ouvrage en fonction de diverses pondérations de la charge, suivant des coefficients affectés à certaines tâches du projet.

Il faut garder en tête que les étapes et les coefficients calculés ici sont adaptés au Client-Serveur ; pour pouvoir être en adéquation avec les xNet, A.Probst propose d'éventuellement diminuer la charge du projet de 20% par rapport à sa valeur d'origine (pour avoir un résultat rapide), ou mieux, ajouter des étapes, et adapter les différents coefficients.

Par la suite, certaines charges calculées précédemment sont directement liées au niveau d'expertise de l'équipe de développement. On va ainsi pouvoir les pondérer. On compte parmi ces tâches le Maquettage, l'Architecture Applicative, les Spécifications Techniques, le Codage des Composants, et les tests.

On applique à chacune de ces charges un coefficient de pondération en fonction du niveau d'expertise du développeur.

Puis, on va estimer un besoin « transverse » : un surcoût lié à des structures transverses, s'impliquant principalement dans le projet lors de réunions, pour des besoins de suivi de projet en terme d'architecture, de qualité, et de sécurité.

On pourra ainsi calculer la charge du projet en extrayant les étapes correspondant à la conception, la construction du projet, et sa recette.

Méthode interne ICDC (ESTIMAT)

ESTIMAT est un outil mis au point au début des années 1990 par une cellule d'Informatique CDC.

Cette méthode se base principalement sur la méthode Algorithmique : le principe consiste à utiliser des formules mathématiques pour exprimer l'effort en fonction d'éléments mesurés, ou estimés.

Par la suite, on leur applique un coefficient d'ajustement se basant sur l'environnement. (Le principe de ces coefficients est relativement similaire au modèle COCOMO). Ce sont des données issues de l'expérience qui vont servir à déterminer ces coefficients.

Le but de la première opération est de calculer un effort tenant compte, à la fois de la complexité des règles de gestion et des règles de mise à jour de la base, mais aussi du volume des travaux à effectuer.

On distingue ainsi deux types de traitements :

- Le traitement Conversationnel : (ou Transactionnel) - Nombre d'écrans dédiés à l'utilisateur
- Le traitement par lots : Nombres d'états – Traitement automatisé

ESTIMAT ***Estimation détaillée***				
JH		Simple	Moyen	Difficile
Traitements conversationnels	Fixe	5	10	15
	Variable	3	4	5
Traitements par lots	Fixe	4	8	12
	Variable	4	6	8

À l'intérieur de ces traitements, on va distinguer deux parties :

- Partie Variable : écrans, ou traitements.
- Partie Fixe : Traitement de la partie variable, Processus. (Coût de la séquence entre les parties variables)

L'effort est calculé en sommant les valeurs associées aux types et à la complexité des divers traitements suivant le tableau ci-dessus. On pondèrera cet effort grâce à un coefficient résultant d'ensemble de facteurs, similaires à la méthode COCOMO. Ce coefficient peut varier de 0.12 à 7.5 (un rapport de plus de 60… ce qui reste très raisonnable par rapport à COCOMO, mais reste extrêmement élevé dans l'absolu), sa valeur étant égale à 1 pour un projet normal.

Cette méthode ESTIMAT est visiblement inspirée d'une métrique de type « Points de Fonction » pour l'estimation nominale de la charge, et de COCOMO pour la pondération. Elle a l'avantage indéniable d'affecter à la charge nominale des coefficients permettant d'estimer la charge totale, des spécifications à la mise en production. La méthode est, a priori, bien adaptée à tous les projets en général, puisque ne dispose pas de paramètres spécifiques au type d'application développée. Il est donc aisé, une fois de plus, d'adapter les coefficients de la méthode pour obtenir une estimation propre aux xNet.

Évaluation d'un xNet

Vue d'ensemble

Les méthodes que nous avons analysées jusqu'à présent (hormis celle de J-P.Vickoff) ne sont pas, à proprement parler, dédiées à l'évaluation des xNet. C'est d'autant plus cohérent, que ces méthodes datent d'avant le milieu des années 1990.

Il est relativement légitime de considérer le développement d'un xNet comme celui d'une application en client-serveur (et ce, pour des raisons techniques). Cependant, on notera tout de même quelques différences flagrantes entre ces deux mondes, qui ne sont pas sans avoir un impact important sur un paramètre de développement qui nous intéresse particulièrement ici, à savoir la charge nécessaire au développement.

Tout d'abord, il faut savoir que le client sur le xNet est universel (c'est le navigateur). Le code de l'application « client » s'en trouve de fait considérablement simplifié : grâce à un « méta-langage » à la fois simple, et ayant un formidable potentiel en terme d'interface, il n'est plus nécessaire d'avoir un code lourd du côté client… sans compter que la simplicité du HTML permet une capacité de modification du code client sans comparaison. De même, grâce à la standardisation du CGI et du HTTP, on s'affranchit des protocoles complexes de communication (Sockets, RMI, RPC, CORBA, SOAP, COM, etc.) inhérents au client-serveur.

Ensuite, nous avons pu voir dans certains méthodes que les applications de type « RAD » pouvaient être développées beaucoup plus rapidement… pour dire la même chose d'une autre façon, si l'utilisateur peut donner le plus souvent possible un retour sur l'application en cours de développement, les chances de voir le projet dévier, seront amoindries (dans une certaine mesure, bien entendu), et le développement devrait en être d'autant plus rapide.

Dans la réalité, ce genre de fonctionnement n'est pas forcément du goût de la maîtrise d'œuvre… en effet, si ce levier est mal utilisé par la maîtrise d'ouvrage, la pression subie par les développeurs, et les changements d'opinions (inévitables) de la part des maîtrises d'ouvrage pourrait également faire perdre du temps de développement.

Il faut aussi bien voir que si une partie de la charge de la maîtrise d'œuvre est réduite, du fait d'une vérification assez assidue de la maîtrise d'ouvrage, ce gain de charge est contrebalancé de l'autre côté par une trop grande implication et une perte de temps pour la maîtrise d'ouvrage… ce qui est également coûteux.

Enfin, il existe divers problèmes de « sécurité » dus au fait que le client soit standardisé… ceci pouvant nécessiter une charge supplémentaire, assignée au contrôle du déroulement correct du processus, aux vérifications des authentifications, au comblement des trous de sécurité, etc.

En bref, le type d'application xNet peut aussi bien se trouver être largement plus simplifiée en terme de développement que le client-serveur, ou bien nécessiter à peu près la même charge, voire complètement dépasser les délais prévus, et ce, en fonction principalement, de la complexité de l'application… et des contraintes annexes.

Modules xNet

Il faut savoir, par la suite, que le développement d'un xNet connaît plusieurs étapes potentielles. La première, est à proprement parler, celle de la « création », et la seconde, celle de l'« ajout de modules ».

La création d'un xNet est l'étape qui a lieu avant toutes les autres. (les autres étant des « ajouts de modules », si le cas se présente).

Il s'agit en fait, de la toute première réalisation, de la « création » à proprement parler. Lors de cette étape, sont établies toutes les bases de l'application : framework applicatif, module d'authentification, module de navigation, charte graphique, etc. et, bien évidemment, les modules métiers, pour lesquels le xNet a été mis en place.

La seconde étape pourrait être comparée à une simple « maintenance évolutive » : il suffit de greffer à l'existant une application possédant ses propres règles fonctionnelles. Le terme « greffer » est tout à fait adapté : il s'agit en effet de se baser sur les modules d'identification et de navigation existants (et au besoin, les modifier) pour ajouter une application à l'existant.

Ce type de « projet » est beaucoup moins spécifique que lors d'une création, et s'apparente, dans ce cas, plus à de l'informatique classique.

Il faut savoir que les solutions techniques de créations de xNet sont multiples. Celles-ci peuvent aller de la simple page en HTML sans base de données, à des technologies objets métiers (Beans, EJB,

Java…), des procédures stockées en base (PL/SQL…), en passant par des langages pré-compilés (CGI en général…), ou interprétés côté serveur (Perl, ASP, PHP…).

Bref, la maîtrise d'œuvre dispose d'un panel de choix technologique on ne peut plus important. Le choix en est d'autant plus difficile qu'un même résultat peut être obtenu en utilisant des technologies tout à fait différentes, parfois de la plus simple à la plus complexe.

En fait, tout dépend du niveau de réutilisabilité de l'application, du besoin en maintenance, de la performance désirée. Il va falloir, quels que soient les délais imposés, trouver le meilleur compromis entre le temps de développement, les contraintes de plate-forme technologique, le besoin en réutilisabilité et en maintenance, etc.

D'autant qu'il est de plus en plus nécessaire de revenir par la suite sur les applications, pour des besoins en terme d'ergonomie, ou d'évolutivité du produit. Le temps passé en développement doit être considéré comme un « investissement" sur les maintenances à venir. (Comme dans tous les projets informatiques, d'ailleurs).

Alors comment évaluer le xNet ? La méthode idéale pour évaluer un xNet serait, de façon intuitive, d'adapter un modèle existant (La méthode Probst, par exemple, étant la plus transparente, est très adaptée) en modifiant les coefficients attribués.

Un exemple concret

Le projet RPI

Le projet RPI est un projet interne de la DSI de la Caisse des Dépôts et Consignations, qui a pour but de mettre en place un certain nombre de tableaux de bord concernant les projets informatiques de la CDC. Ces tableaux font apparaître aux décideurs des informations pertinentes sur les projets informatiques qui porteront sur les coûts, les délais et les risques. L'objectif consistait à définir et mettre en place un ensemble de modules se greffant sur l'intranet de la Caisse des Dépôts. Ces modules constituent des interfaces d'alimentation et de consultation de la base de données de l'application RPI.

Estimations théoriques RPI

Grâce aux méthodes analysées, cette application RPI (ou tout du moins son premier lot) a pu faire apparaître les données suivantes :

Points de Fonctions Bruts : 73 ; Points de Fonctions Ajustés : 75.

Dans le cas d'un projet « simple », la méthode conseille d'utiliser une estimation 2 Jours*Homme par Points de fonctions. Soit une charge de 2 * 75 = 150 J*H

Méthode	Valeur nominale	Valeur pondérée
Réel	48	55
Probst	60	60
ESTIMAT	70	72
RAD - Evaluateur	66	96
Points de fonction	146	150
COCOMO	281	331,5

Taille réelle de l'application : 3.798 KLOC, dont 2.500 KDSI (Chiffre donné par la MOE)

Taille théorique grâce aux Points de fonction :

T = 0.1187*75-6.49 = 2412 KDSI

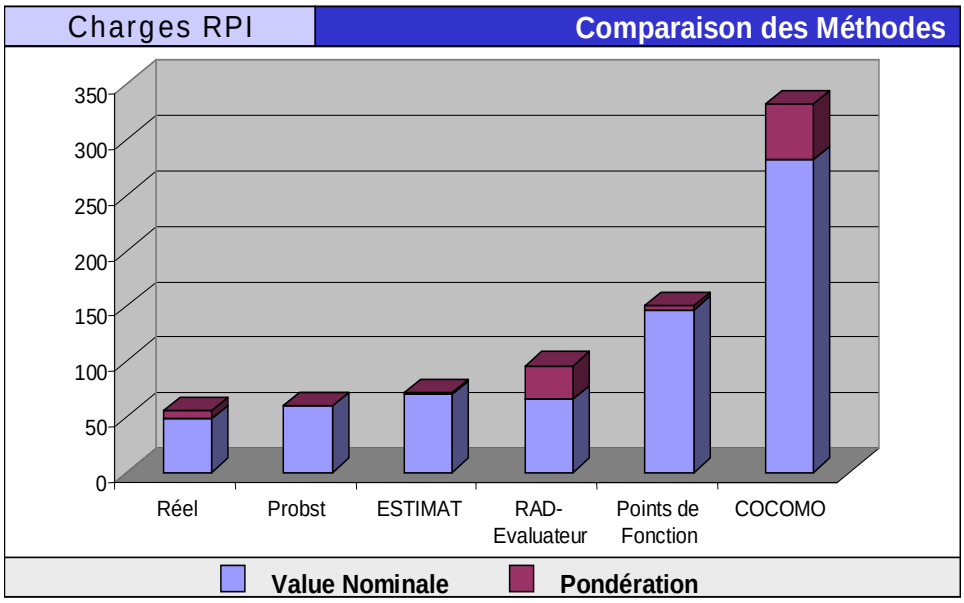
(Proche des 2500 réels)

COCOMO semi-détaché donne E = 331.5 J*H

Évaluateur-RAD donne une valeur nominale du projet est de 66 J*H, ajustés à 96 J*H .

La méthode Probst fournit un résultat de 60 J*H

La méthode ESTIMAT fournit un résultat de 72 J*H



Le lot 1.1 de l'application RPI avait été estimé à 48 J*H. Sa réalisation en a consommé 55.

On peut tirer de ces résultats les conclusions suivantes :

La méthode COCOMO ne semble pas être adaptée : si l'on considère réellement les KDSI ou les KLOC, en fonction du langage utilisé, pour la même application, on obtiendra des résultats réellement disproportionnés. D'autant que la méthode COCOMO ne tient pas du tout compte de l'application du côté client et de la complexité de l'ergonomie.

La méthode des points de fonction semble être excellente pour mesurer la « taille » d'une application, en terme de KDSI ou KLOC, mais le facteur de 2 JH par Point de Fonction est beaucoup trop élevé. Il aurait fallu, pour avoir un résultat exact, faire le calcul avec 0.75 JH par Point de Fonction.

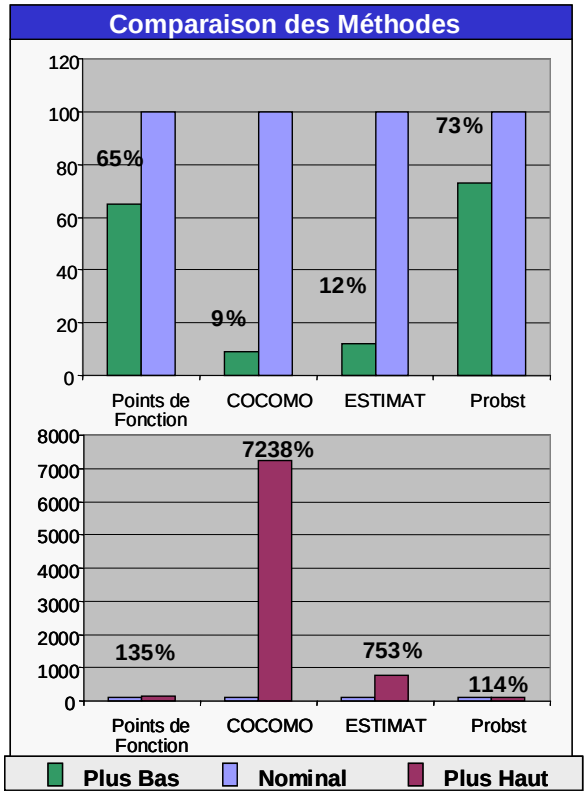
La méthode RAD-ÉVALUATEUR donne un résultat nominal (métrique d'origine) très acceptable, mais les pondérations opaques faites par le logiciel font perdre de la crédibilité à la méthode, ce qui est d'autant plus dommage que le logiciel a été développé spécifiquement pour estimer la charge des xNet.

La méthode ESTIMAT est proche du résultat final. Il faudrait probablement pour être proche de la réalité, y ajouter certaines données relatives aux modules d'identification et de navigation, et modifier les coefficients de complexité des composants.

C'est la méthode Probst qui se rapproche le plus de la réalité (moins de 10% d'erreur sur ce projet). Comme suggéré lors de l'analyse préalable, il faudrait là aussi, ajouter des données relatives aux modules d'identification et de navigation, et les charges d'opérations transverses, pour obtenir un résultat réellement fiable.

Conclusion

Au bilan, cinq méthodes plus ou moins différentes les unes des autres. Chacune a ses propres paramètres, chacune donne ses propres résultats.



Et la fourchette des paramètres entrés étant réellement large, on ne peut pas réellement faire ressortir les paramètres de métriques principaux. De plus, les variables de sortie étant elles mêmes peu cloisonnées, on ne sait jamais exactement ce que l'on est en train d'estimer (les tests unitaires et la recette sont-ils compris dans les charges de la réalisation, etc.)

En bref, vouloir faire ressortir un résultat précis et bien délimité, à partir d'une science non exacte prenant en compte toute sorte de paramètres, dont certains sont plus ou moins farfelus, se révèle être, au final plus un exercice de style, qu'une méthode fiable : on jongle avec beaucoup de choses, et il est on ne peut plus aisé de faire rendre à la méthode les chiffres que l'on veut.

Cependant tout n'est pas perdu : le fait de « croiser » toutes ces méthodes peut permettre de donner une estimation plus ou moins réaliste (à un facteur près, puisque toutes les évaluations se révèlent être trop élevées.)

Ainsi, une bonne solution pourrait être, si l'on désire effectuer rapidement une évaluation d'un projet xNet, de créer une sorte de « Tableau de Bord », destiné à la maîtrise d'ouvrage, qui serait également fourni à la maîtrise d'œuvre, et contenant un ensemble de points sur lesquels ils pourraient se mettre d'accord, en l'occurrence sur les paramètres d'entrée... ceci nécessitant une demi-journée, tout au plus.

La maîtrise d'ouvrage pourrait rapidement avoir un aperçu global des estimations ainsi faites.

Ce tableau de bord pourrait contenir les informations suivantes :

Estimation de la maîtrise d'œuvre, métriques des cinq méthodes (Nombre et complexité d'écrans, de tables, de fonctions, etc.), estimations faites avec les Points de Fonctions, COCOMO, RAD-Evaluateur, Probst, et ESTIMAT.

Présentation des marges d'inexactitude des pondérations en fonction des méthodes de la figure ci-contre. En effet, ces pondérations font varier la taille nominale du projet de façon inquiétante. La maîtrise d'ouvrage, en voyant ces chiffres, aura probablement le réflexe d'utiliser la méthode où l'erreur due à la pondération est la plus faible.

... cette solution amènerait la maîtrise d'ouvrage à juger elle-même du chiffre d'estimation des charges qu'elle peut obtenir, tout en lui mettant en avant qu'il est évident que le résultat obtenu sera d'une fiabilité relative, en raison des marges d'erreurs, et de la difficulté à comparer ces méthodes hétéroclites... Le plus important de ce « tableau de bord» étant probablement d'apporter des arguments nécessaires à une discussion saine et constructive, en matière d'élaboration de planning, entre les maîtrises d'œuvre et d'ouvrage.

Georges Zadrozynski
gz@msit.org
http://www.gezzed.net
Ingénieur EISTI
Mastère Management des S.I., HEC & ENSMP

Bibliographie

La mesure du logiciel (Henri Habrias) 2ème édition revue, corrigée et augmentée, Teknea (ISBN : 2877170454)

Piloter les projets informatiques de la nouvelle économie (Jean-Pierre Vickoff), Éditions d'Organisation (ISBN : 2708124870)

Project Management: a Managerial Approach (Jack R. Meredith, Samuel J. Mantel Jr.), John Wiley and Sons (ISBN: 0471434620)

Web Site Project Management (Ashley Friedlein), Morgan Kaufmann (ISBN: 1558606785)

Gestion d'un projet système d'information - Principes, techniques et mise en œuvre (Chantal Morley), Dunod (ISBN: 2100059734)

Précis de Conduite de Projet Informatique (Cyrille Chartier-Kastler), Éditions d'Organisation (ISBN : 2708118145)

Conduite de projets informatiques : principes et techniques (José Moréjon, Jean-René James), Interéditions (ISBN : 272960457X)

Le management d'un projet (H.-P. Maders), Éditions d'Organisation (ISBN : 2708117947)

Que Sais-je : Le management de projet (Olivier Badot, Jean-Marie Hazebroucq), Presses Universitaires de France (ISBN : 2130474179)

Le maître d'ouvrage du système d'information (Alain Berdugo), Hermès Sciences (ISBN : 2866016211)

Guide du Management des Systèmes d'Informations - Thèmes & Termes Essentiels (Alain Berdugo, Robert Mahl, Gérard Jean, HEC/ENSMP MSIT 2002), Hermès Sciences (ISBN : 2746205246)

De la Stratégie aux Systèmes d'Information (ALTIME) Formation ALTIME/HEC/ENSMP, février 2002

[ADELI] Estimations de charges - Les orientations de la commission (Alain Coulon)
Lettre ADELI n°39 - Square des Utilisateurs - avril 2000

[ADELI] Estimation de projets informatiques (Kathleen Peters, traduit par Alain Coulon)
Lettre ADELI n°41 - Square des Utilisateurs - octobre 2000

[ADELI] Calibrage et étalonnage (Jean Joskowicz)
Lettre ADELI n°42 - Square des Utilisateurs - janvier 2001

[ADELI] Science ou magie ? - Le point sur les estimations des projets logiciels (Nicolas Trèves et Alain Coulon)
Lettre ADELI n°45 - Square des Utilisateurs - octobre 2001

La méthode des Points de Fonction (Olivier Denel), Thèse MSIT (HEC/ENSMP) - http://www.denel.com/pdf/index.php

Rapport CIGREF : Réseaux Internet/Intranet, Rapport CIGREF, septembre 1997

COCOMO: Resource Estimation (David Stotts, Associate Professor, Dept. of Computer Science) University of North Carolina - http://www.cs.unc.edu/~stotts/COMP145/cocomo.html

IFT3902 : Développement, maintenance de logiciels (Université de Montréal : Faculté des Arts et des Sciences)
Informatique et recherche opérationnelle,
http://www.iro.umontreal.ca/~pift3902/Cours2001/Planification.pdf

Estimation des coûts et délais par la méthode COCOMO (Tina Wilhelm, Évelyne Parthenay, Hugues Am)
École Supérieure en Sciences Informatiques,
http://www.essi.fr/~hugues/GL/COCOMO/cocomo.html

Algorithmics Costs Model (Dan Snell), Bournemouth University - http://www.ecfc.u-net.com/cost/models.htm

Évaluation, la théorie (Jean-Pierre Vickoff), http://mapage.noos.fr/rad/boudeval.htm

Principes d'Évaluation (Jean-Pierre Vickoff), http://mapage.noos.fr/rad/evalint.htm

Méthode Interne CDC – Feuille de Calcul d'Estimation de Projets (André Probst)

Méthode ESTIMAT (Informatique CDC, Méthodes Ingénierie Progiciels)