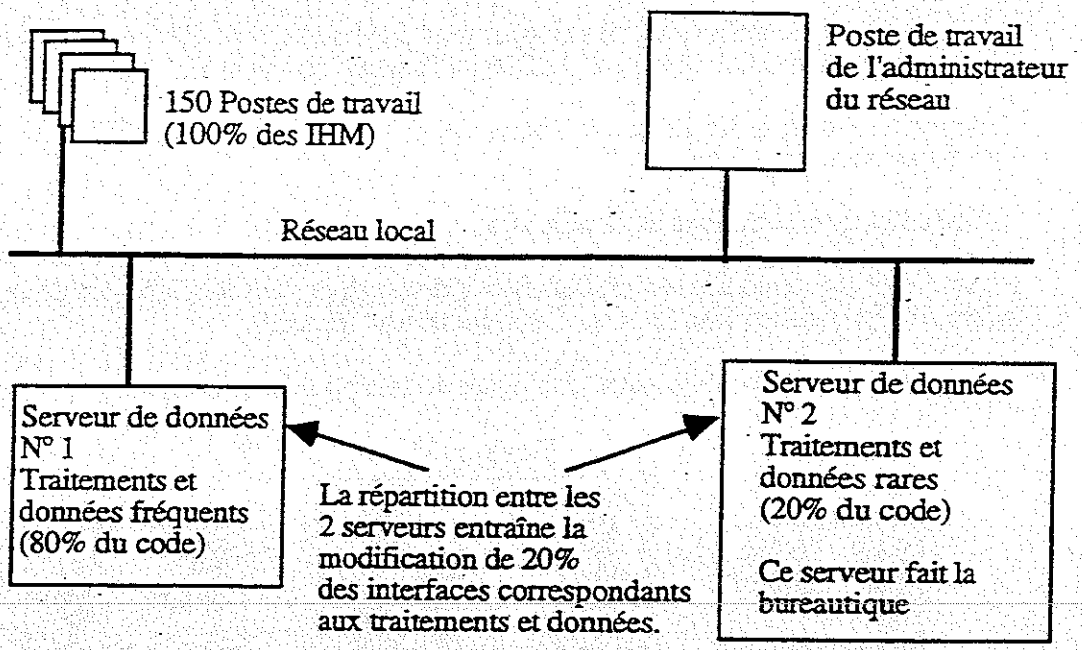
Nouvelle configuration client-serveur:



Questions:

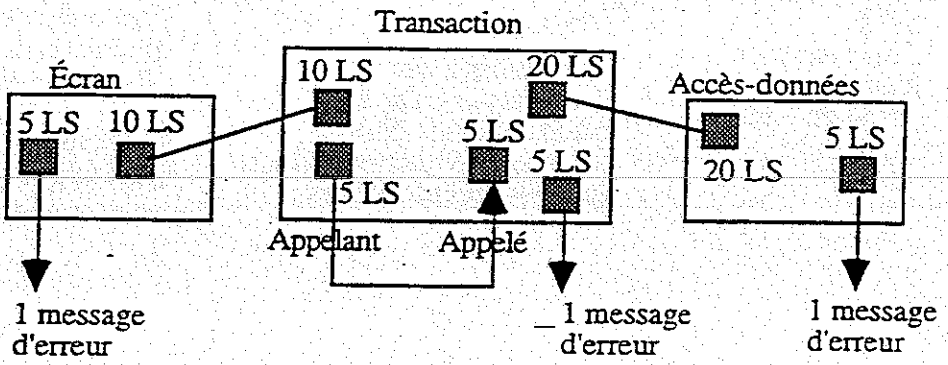
1. Reconstituer le dossier de développement de l'application GP sur le main-frame en vous servant du modèle de base COCOMO (Tableaux 6-1 et 6-8).

Donner les valeurs de: Effort et Temps de développement total, Effort et Temps de développement par phase.

2. Faire l'analyse technique de la migration:

- nombre de lignes source à modifier et/ou à développer dans les IHM, les traitements, les accès aux données et dans le poste d'administration réseau.

Mode de calcul des lignes source pour les interfaces:



Exemple d'emploi:

1 transaction qui appelle 3 écrans, émet 7 messages d'erreurs, appelle 4 autres transactions, et accède à 2 interfaces d'accès aux données a comme volume de code d'interfaces: 1×5LS + 3×10LS + 7×5LS + 4×5LS + 2×20LS = 130 LS

- types des programmes résultant de ces modifications (la bureautique du serveur N°2 est absolument prioritaire et ne doit pas être perturbées; les messages d'erreurs sont remontés sur le poste d'administration réseau qui doit maintenant gérer l'ensemble; on estime qu'il faut 5 LS par message dans le poste d'administration).

- estimation de l'effort à fournir et du temps nécessaire à cette migration

3. Comment renforcer l'équipe de développement; en vous servant du tableau 8-2 du modèle intermédiaire, analyser les conséquences prévisibles de ce renfort.

- NB: ACAP indique l'expérience de l'analyste (l'architecte de l'application), AEXP indique l'expérience acquise sur l'application, PCAP indique l'expérience du programmeur (voir tableau 8-3).

Correction de la question de cours

1ère Question

On arrondit les calculs à l'unité.

Reconstitution du projet:

Programme	Taille en KLS	Type	Poids	Effort par programme	TDev par programme	Effort pondéré	TDev pondéré (Durée)
IHM	15	E	0.125	$3.6(15)^{1.2} = 93$	11	1125×0.125=141	24×0.125=3
Traitements	75	S	0.625	$2.4(75)^{1.05} = 223$	20	366×0.625=229	24×0.625=15
Données	30	P	0.250	$3.0(30)^{1.12} = 135$	14	639×0.25=160	24×0.25=6
	120			(*1) 451HM	(*2)	**530HM**	**24Mois**

*1: ce total ne tient pas compte de l'effort de fractionnement et d'intégration; c'est un minorant de l'effort.
*2: Cette somme de TDev n'a pas de sens, car les 3 programmes sont menés en parallèles. La seule chose que l'on puisse dire est que le TDev global est > 20.

Ventilation par phase:
On utilise la table 6-8, en prenant la colonne large (128KDSI). Le mode P (semidetached) est une très bonne approximation, mais on pourrait également faire une moyenne pondérée (ce qui triplerait les calculs).

Ventilation	Effort	Durée
Conception générale	530×0.17=90 HM	24×0.27=6.5 mois
Programmation • Conception détaillée • Codage/tests unitaires	530×0.55=291 HM 530×0.24=127 HM 530×0.31=164 HM	24×0.44=11 mois
Intégration et tests	530×0.28=148 HM	24×0.29=7 mois

2ème Question: décompte des lignes de code à modifier/développer

IHM

Toutes les émissions de messages d'erreurs sont a refaire ainsi que l'interface d'appel de chaque écran, donc:

Interface	Décompte de l'interface	Volume des modifications
I_message	5mess×30écrans×5LS=750LS	100% -> 750
i_appelée	30écrans×10LS=300LS	100% -> 300
		Total: 1050LS

2D

Traitements

Toutes les interfaces (appels de transactions, erreurs, écrans, données) sont à revoir. mais compte tenu de la répartition adoptée entre les serveurs:

— il n'y aura que 20% du code des interfaces d'appels de transactions et d'accès aux données qui devra être refait,

— mais 100% du code des interfaces écrans et messages d'erreurs,

donc:

Interface	Décompte des interfaces	Volume des modifications
I_message	300×1×5LS=1500LS	100% -> 1500LS
I_écran	300×2×10LS=6000LS	100% -> 6000LS
I_données	300×5×20LS=30000LS	20% -> 6000LS
I-appelée	300×2×5LS=3000LS	20% -> 600LS
I_appelante	300×1×5LS=1500LS	20% -> 300LS
		Total: 14400LS

NB: Pour une transaction le volume de code interface est:

1[appelante]×5LS + 2[appelées]×5LS + 1[message]×5LS + 5[I_Données]×20LS + 2[I_écrans]×10LS= 140LS

Données

Interface	Décompte des interfaces	Volume des modifications
I_message	30×1×5LS=150LS	100%-> 150LS
I_appelant	30×1×20LS=600LS	20%-> 120LS
		Total: 270LS

Administration

Le poste d'administration nécessite un programme de gestion des messages entièrement nouveau.

Le nombre de messages reçus est:

30 écrans×5 + 300 Transactions×1 + 30 I_Données×1 = 480 messages

Le programme de gestion de messages a un volume de:

480×5LS = 2400LS

Soit un volume total de modifications/développement pour l'application:

1050LS + 14400LS + 270LS + 2400LS = 18120LS

Ce code est bien évidemment de **type E.**

Pour ce qui a été modifié:

Effort de développement: $3.6(18.12)^{1.2}$ =116HM

Durée de développement: $2.5(116)^{0.32}$ = 11 mois

Il faut refaire une intégration et les tests (car ils ont été perdus!). donc on doit rajouter une certaine fraction de l'effort d'intégration et test calculé ci-dessus, soit 148HM:

Pessimiste: 50% -> 74HM

Optimiste: 30% -> 44HM

Soit une fourchette d'effort:

Haute: 116 + 74 = 190HM, Basse: 116 + 44 = 160HM

On peut conserver une **durée de développement de 11mois**, car on peut paralléliser.

3ème Question: renfort de l'équipe

Le renfort de l'équipe pour la partie développement est donc:

116HM/11mois = 11personnes

Il faut donc **recruter 7 personnes**, en plus des 4 disponibles. Ces 7 personnes doivent bien connaître les contraintes de programmation réseau pour avoir un bon coefficient PCAP. Les personnes en place n'ont pas d'expérience réseau mais elles connaissent bien l'application et ses messages.

On peut prendre un équation d'effort du type:

$HM = 2.8 \times 0.86 \times 0.91 \times 0.86 (KLS)^{1.2}$ soit un coefficient = 1.88 (au lieu de 3.6)

ce qui est certainement optimiste.

Il y a un problème de cohabitation prévisible compte tenu du fait que les expérience des programmeurs sont très contrastées. L'architecte de l'application ne connaît pas les réseaux et les programmeurs recrutés ne connaissent pas l'application.

On peut tenter une organisation en 2 équipes

1 équipe de développement de la partie réseau,

1 équipe d'intégration et test qui refait la bibliothèque de test (c'est à dire les 74HM ou 44HM)

De toute façon, cette migration sera très délicate.

EXAMEN DE GENIE LOGICIEL

2ème session du
12 septembre1995

1. Exercice de conception

2 . Exercice de tests

3 . Question de cours

QUESTION DE COURS: Étude de la mise en place d'une politique qualité axée sur la prévention des défauts

Les corrections sont en *italiques*.

La société LOGICIEL-MICRO a décidé de revoir complètement sa politique qualité à l'occasion de la sortie de son nouveau produit Fenêtres-95. Le parc de machines auquel est destiné ce nouveau produit est estimé à 200 000 unités, dans un premier temps. Les ventes sont réalisées grâce à un réseau de revendeurs qui assurent la maintenance et le support technique de 1er niveau.

La collecte de statistiques sur les versions antérieures de ce logiciel fait apparaître les éléments suivants:

— Taille du logiciel: 1000 KLS.

— Taux d'erreurs par KLS: 3 (soit 3000 erreurs découvertes durant l'exploitation.

— Ratio RA / ERR: 5 (c'est à dire 5 Rapports d'Anomalies par erreur réelle, soit 15 000 RA collectés).

— Le support technique estime que chaque erreur détectée concerne 3 sites en moyenne (soit 9000 sites fautifs sur lesquels une intervention effective a été requise).

— La durée de vie d'une version est de 18 mois environ et il a été estimé que le flux d'erreurs décroissait de 50% tous les 6 mois (soit 1500 erreurs les 6 premiers mois, puis 750, puis 375, etc.).

Question 1: Établir le bilan économique (coût de Non-Qualité) de cette opération, en prenant comme facteurs de coûts les éléments suivants:

— Rédaction du RA par le support technique: 2 Heures

— Réception du RA et maintenance 1er niveau: 4 Heures

— Réception du RA et maintenance 2ème niveau: 8 Heures (on estime que la maintenance 2ème niveau ne reçoit que 2 RA sur les 5, les 3 autres ont été filtrés par le 1er niveau).

— Installation et test de la correction sur le site fautif: 4 Heures

NB: On prendra 1 journée = 8 Heures, 1 année = 220 journées.

On calculera le coût en HA (Homme-An).

Coût des 15000 RA:

2H=0.25j *0.25×15000=3750j*

4H=0.5j *0.5×15000=7500j*

8H=1j *1×6000=6000j (il n'y a que 6000 RA transmis au niveau 2)*

4H=0.5j *0.5×9000=45000j (il y a 9000 sites fautifs)*

Soit: 21750j = 21750/220 = 99HA

Question 2: Objectif d'amélioration et espérance de gain.

LOGICIEL-MICRO voudrait atteindre un taux d'erreur de 1 ERR par KLS (pour simplifier les calculs, on considère que la taille du logiciel est invariante à 1000KLS). L'étude plus fine des 3000 erreurs fait apparaître les éléments suivants qui sont ventilés par phase:

Nature de la phase	% ERR détectées	Nombre d'erreurs	%ERR espérées	Nombre d'erreurs prévisionnel
Expression de besoin - Spécification	10%	300	20%	*200*
Conception Préliminaire	30%	900	25%	*250*
Conception Détaillée	40%	1200	30%	*300*
Codage - Tests unitaires	20%	600	25%	*250*
Cumul	100%	3000	100%	1000

Le pourcentage d'erreurs espéré est en fait une distribution de référence pour un niveau de qualité considéré comme bon.

Que peut-on déduire de la comparaison de ces différents chiffres, en particulier de la distribution des erreurs détectées?

Le tableau dénote un assez fort décalage de la distribution des erreurs des phases de conception. Le taux d'erreur de ces phases est trop élevé.

Compléter le tableau et calculer l'économie réalisée si l'objectif espéré est atteint.

Il y aura 3 fois moins d'erreur à traiter, donc les coûts seront divisés par 3, soit 33 HA. L'économie réalisée est de 66HA.

Question 3: Chiffrage d'une politique de prévention et conséquences.

La ventilation des erreurs par phase et les phénomènes d'amplification des erreurs d'une phase à la suivante permettent de construire, pour les erreurs observées, le tableau suivant:

Nature de la phase	ERR Transmises T	ERR Amplifiées Axc	ERR Nouvelles N	Efficacité de la détection k	Cumul en sortie (T+A+N)×(1-k)
Expression de besoin - Spécification	—	—	**300**	0	300
Conception Préliminaire	300	—	**900**	0	1200
Conception Détaillée	800 (2/3)	400×1.5 (1/3)	**1200**	0	2600
Codage - Tests unitaires	850 (1/3)	1750×3.5 (2/3)	**600**	20%	6000

Intégration	6000	—	—	50%	**3000**

NB: Certaines valeurs ont été arrondies pour simplifier les calculs.

Explication du tableau:

— Les valeurs en gras résultent des observations; les autres valeurs sont des estimations considérées comme les plus vraisemblables.

— L'Expression de besoin - Spécification contient 300 erreurs. Il n'y a pas de mécanisme additionnel de détection d'erreurs, donc le coefficient d'efficacité k = 0. Les 300 erreurs sont transmises telles quelles à la conception préliminaire.

— La Conception Préliminaire rajoute les 900 erreurs imputables à cette phase, auxquelles on a rajouté les erreurs provenant de/des phases antérieures en les répartissant en T et A. Pour la conception préliminaire, il n'y a pas de phénomène d'amplification, donc les 300 erreurs s'imputent dans la colonne T.

— À partir de la Conception Détaillée, le phénomène d'amplification se manifeste de plus en plus fortement, soit 1/3 amplifiée pour la Conception Détaillée, et 2/3 pour le Codage. Les coefficients d'amplification étant respectivement de 1.5 et 3.5 (environ). Pour la phase de codage, il y a déjà une politique de prévention qui filtre 20% des erreurs, ce qui est considéré comme insuffisant.

— La phase d'intégration dont le but est essentiellement de faire des tests filtre 50% des erreurs, ce qui fait que les 3000 erreurs résiduelles découvertes en exploitation correspondaient à 6000 erreurs en fin de codage - tests unitaires. Ces 6000 erreurs sont le cumul des erreurs imputables aux différentes phases avec d'éventuelles amplifications.

— L'intégration est supposée ne pas rajouter d'erreur (ce qui est certainement faux, mais cela simplifie les calculs).

Ce tableau caractérise l'expérience et la maturité de l'organisation et des équipes de développement. On considère que cette maturité est stable (ce qui, en toute rigueur, n'est certainement pas vrai; mais cela nous permet de simplifier les calculs).

La politique qualité préconisée va donc se traduire par des coefficients d'efficacité k, qui correspondent à des imputations à effectuer pour les éléments V et S du paradigme ETVS. Ces imputations sont un coût qualité qui sera à comparer à l'économie réalisée si les objectifs sont atteints. On suppose que le produit Fenêtres-95 totalise 1000KLS.

Dans le tableau ci-dessous, calculer un jeu de coefficients k1, k2, k3 qui respecte les données constantes.

Nature de la phase	ERR Transmises T	ERR Amplifiées Axc	ERR Nouvelles N	Efficacité de la détection k	Cumul en sortie (T+A+N)×(1-k)
Expression de besoin - Spécification	—	—	**300**	k1	**=200**
Conception Préliminaire	200	—	**900**	k2	**≥450**
Conception Détaillée	?1 (2/3)	?2×**1.5** (1/3)	**1200**	k3	**≥750**

Codage - Tests unitaires	?3 (1/3)	?4×3.5 (2/3)	600	k4≥20%	2000
Intégration	2000	—	—	50%	=1000

NB: On a supposé que le nombre d'erreurs commises — colonne N — pendant le développement restait stable (en toute logique il devrait décroître pour tenir compte de l'expérience). La colonne "Cumul en sortie" fixe les contraintes imposées par les nouveaux objectifs. Sur les 300 erreurs de la 1ère phase, il faut en éliminer 100 pour tenir l'objectif de 200, d'où une certaine valeur de k qu'il faut calculer. Pour les phases de conception, il faudrait idéalement sortir avec exactement 450 et 750, mais une certaine tolérance, en excès, est accepté. Par contre l'objectif de sortie de la phase codage - tests unitaires est impérativement 2000 pour obtenir une sortie d'intégration avec 1000 erreurs résiduelles. Les valeurs k1, k2, k3, k4 corrèlent l'effort qu'il faut consacrer à la recherche des défauts.

On peut faire la mise en équation suivante:

$300\times(1-k1) = 200$ *soit:* $k1 = 33\%$

$(200+900)\times(1-k2) \geq 450$ *soit:* $k2 \leq 59\%$

$((2/3\times1100 + 1/3\times1100\times1.5)\times(1-k2) + 1200)\times(1-k3) \geq 750$

soit: $(1283\times(1-k2) + 1200)\times(1-k3) \geq 750$; $k3 \leq \dfrac{1733-1283\times k2}{2483-1283\times k2}$

1er Jeu: $k1 = 33\%$, $k2 = 59\%$, $k3 = 57\%$; *ce qui permet de calculer k4.*

$(750(1/3+2/3\times3.5) + 600)\times k4 = 2000$, *d'où* $k4 = 23\%$

2ème jeu: $k1 = 33\%$, $\overline{k2} = 0\%$, $k3 = 70\%$

Nature de la phase	ERR Transmises T	ERR Amplifiées Axc	ERR Nouvelles N	Efficacité de la détection k	Cumul en sortie (T+A+N)×(1-k)
Expression de besoin - Spécification	—	—	300	33%	=200
Conception Préliminaire	200	—	900	0%	1100
Conception Détaillée	733 (2/3)	367×1.5 (1/3)	1200	70%	745
Codage - Tests unitaires	248 (1/3)	497×3.5 (2/3)	600	23%	2000
Intégration	2000	—	—	50%	=1000

d'où k4 = 23%

Parmi les moyens à mettre en oeuvre suivants:

...

quels sont ceux qui vous paraissent les plus appropriés et les plus réalistes? Justifier votre réponse.

— Changer le personnel,
Très risqué, car cela risque de changer l'expérience globale de l'équipe et l'on est jamais sûr que les nouveaux soient réellement meilleurs que ceux qu'ils

remplacent. On peut envisager de changer les architectes car ils laissent passer beaucoup d'erreurs.

— Former le personnel,

Mesure certainement très recommandable.

— Faire des inspections et des audits,

Oui, c'est la composante essentielle de l'activité de validation (c'est le V de EVTS).

— Faire de nouveaux tests

Oui, avec la méthode des couvertures.

— Utiliser de nouveaux outils et/ou de nouvelles méthodes,

Faire des prototype et de la modélisation pour les 3 premières phases.

— Mettre en place une structure qualité de x? personnes,

Oui, en particulier pour réaliser les contrôles en entrée et sortie de tâches, ainsi que la bonne marche des inspections et audits de validation. Si l'on découpe le logiciel en modules de 1 KLS, soit 1000 modules et que l'on compte 2 jours par module pour faire toutes les vérifications (ce qui nécessite un personnel particulièrement compétent!), cela fait 2000 j de travail, soit 9HA. Si le travail dure 2 ans, il faudra 5 personnes. On peut également procéder par échantillonage si l'on connait les modules les plus critiques (il faut pour cela une excellent connaissance de l'architecture).

— Faire une prédiffusion chez des clients pilotes,

C'est certainement souhaitable, car la moitié des erreurs sont découvertes dans les 6 premiers mois. Une prédiffusion de 3 mois devrait donc être payante, si il y a assez de clients pilotes.

Quel est l'impact sur les délais des coefficients k1, k2, k3? Quel est l'impact sur la durée et le coût de l'intégration?

La durée des phases 1, 2, 3 devrait augmenter car il est certainement très difficile de faire tout cela en parallèle; de plus, le volume de tests augmente. Par contre la durée de l'intégration devrait être considérablement raccourcie car il y a 3 fois moins d'erreurs. Il faut s'arranger pour que le rallongement des phases amont soit compensé par un raccourcissement effectif de l'intégration. Ceci n'est possible que si le contrôle qualité est fait avec le plus grand sérieux, à l'aide de métrique ad'hoc.

Une valeur de k passant de 0 à >30% vous paraît-elle raisonnable? Justifier votre réponse.

C'est certainement tout à fait déraisonnable car cela implique de profonds changements dans le fonctionnement de l'organisation; de plus il faut faire fonctionner correctement le contrôle qualité du premier coup, ce qui est généralement très difficile.

Quelle pourrait être une politique alternative, si l'on se contentait dans un premier temps de k1 = k2 = k3 = 20%.

Nature de la phase	ERR Transmises T	ERR Amplifiées Axc	ERR Nouvelles N	Efficacité de la détection k	Cumul en sortie (T+A+N)×(1-k)
Expression de besoin - Spécification	—	—	300	*20%*	240
Conception Préliminaire	240	—	900	*20%*	912
Conception Détaillée	*608* (2/3)	*304*×1.5 (1/3)	1200	*20%*	1811
Codage - Tests unitaires	*603* (1/3)	*1208*×3.5 (2/3)	600	*63%*	2000
Intégration	2000	—	—	50%	=1000

Cette politique parait raisonnable pour les 3 premières phases où le coefficient 20% peut probablement être atteint. Mais cela présuppose un énorme effort au niveau de la phase de programmation et des tests unitaires. Il est certainement très difficile de passer de 20% à 63% sans un profond changement des méthodes et techniques de programmation.

Globalement l'objectif de passer en un seul coup de 3 ERR/KLS à 1 ERR/KLS n'est certainement pas réaliste. Si l'on prend 40% pour le codage et 60% pour l'intégration (ce qui est certainement déjà très difficile), on arrive à: (603+1208×3.5 + 600)×0.6×0.4 = 1304 erreurs résiduelles.

EXAMEN DE GÉNIE LOGICIEL

1ère session du 14 février 1996

CORRIGE DE LA QUESTION DE COURS

QUESTION DE COURS: Analyse d'une politique de tests.

Les corrections sont en *italiques*.

La société LOGICIEL-MICRO a décidé de revoir complètement sa politique de tests à l'occasion de la sortie de son nouveau produit Fenêtres-95. L'application qui sert de moyen d'essai à cette nouvelle politique totalise 120 KLS que l'on peut considérer comme de complexité moyenne (type P).

Question N° 1 : Reconstituer, en vous servant du modèle COCOMO joint en annexe, l'historique du projet en recalculant

a) l'effort global de test (Tests unitaires + tests d'intégration).
On considérera que dans la partie programmation-tests unitaires, le pourcentage qui incombe aux tests unitaires est de 50%.

b) la durée de la période de test, avec les mêmes hypothèses que ci-dessus.

Vous comparerez ces résultats avec ceux résultant du modèle simplifié que vous trouverez dans le Vade-mecum du chef de projet.

Effort $= 3.0(120)^{1.12} = 639.45 = 640$ *hm*
Durée TDEV $= 2.5(640)^{0.35} = 23.99 = 24$ *mois, soit 640/24 = 26 personnes en moyenne.*

Calcul de l'effort de tests :

Codage + tests unitaires	—>	*640×0.31 = 198.4 hm dont 50% pour les tests, soit : 99.2 hm*
Intégration et tests	—>	*640×0.28 = 179,2 hm*
Total tests	—>	*278.4 hm*

Calcul des durées :

Codage + tests unitaires	—>	*24×0.44 = 10.56 mois, soit = 5 mois de tests*
Intégration et tests	—>	*24×0.29 = 6.96 mois*
Total durée des tests	—>	*5 + 7 = 12 mois (soit 50% du temps total du projet)*

Selon Vade me cum :
120.000 / 350 = 343 hm (c'est voisin du mode S —> 2.4(120)$^{1.05}$ *= 365 hm)*
Durée avec 8-10 personnes : = 3 ans, dont 1.5 ans de tests.

Question N° 2 : Calculer :
a) l'effort de test moyen rapporté au nombre de lignes sources (c'est à dire x hj/LS). On prendra comme unité d'effort, l'homme-jour, avec 1 homme-mois = 21 hommes-jour.
b) l'effort par test unitaire, en considérant que 1 test unitaire porte sur un composant logiciel de 100 LS (c'est à dire 1200 composants pour les 120 KLS). Vous en déduirez le nombre d'essai-tests unitaires (NB: un même test peut donner lieu à plusieurs essais; un essai-test est estimé à = 0.5j).
c) l'effort par test d'intégration, en considérant que 1 module d'intégration comporte 1000 LS (c'est à dire 120 modules à intégrer).Vous en déduirez le nombre d'essai-tests d'intégration sachant que la préparation d'un essai-test d'intégration prend 3 jours.

L'effort moyen par ligne de code est : (278.4×21)/120.000 = 0.0487j par ligne.
Soit pout un module de code de 100 lignes = 5 jours.

Pour les tests unitaires, on a : (99.2×21)/120.000 = 0.0173j par ligne.
Soit pour un module de 100 lignes = 1.7 jours; c'est à dire en arrondissant 3 essai-tests
Nombre total d'essai-tests unitaires = 3×1200 = 3600

Pour les essai-tests d'intégration, on a (179.2×21)/120.000 = 0.0314j par ligne.
Soit pour un module de 1000 lignes 31.4 jours; c'est à dire en arrondissant 10 essai-tests.
Nombre total d'essai-tests d'intégration = 10×120 = 1200 essai-tests.

Question N° 3 : Calcul du nombre de tests constituant la suite de tests du produit Fenêtres-95.
Une suite de tests est un ensemble de tests {T1, T2,..., Tn). Le passage de l'étape i à l'étape i+1 nécessite la ré-exécution des tests 1 à i pour satisfaire au critère de non-régression. Si l'on a n tests, le nombre d'essais, en appliquant une stricte règle de non régression, est:

$$\sum_{1}^{n} n = \frac{1}{2} \times n \times (n+1) = 1+2+\ldots+n \qquad (F1)$$

(NB : On ne demande pas de démontrer cette formule d'arithmétique, par ailleurs bien connue).
À partir des nombres d'essai-tests unitaires et intégration calculés précédemment, vous en déduirez le nombre de tests effectifs avec lequel le produit Fenêtres-95 a été réellement vérifié et validé, soit:

Nombre de tests $T_{unitaire}$ = ?
Nombre de tests $T_{intégration}$ = ?

Le nombre de TU est donné par la formule

$$\frac{1}{2} \times n(n+1) = 3600 \text{ soit : } n^2 + n - 7200 = 0$$

En calculant la racine positive de cette équation, on a :

$$n = \frac{-1+\sqrt{1+4\times 7200}}{2} = 84.35 \approx 85$$

De même le nombre de TI est :

$$n = \frac{-1+\sqrt{1+4\times 2400}}{2} = 48.99 \approx 49$$

Question N° 4 : Calcul du seuil de rentabilité.
Une analyse critique des essais fait ressortir les faits suivants: On a intérêt à distinguer la réalisation d'un test lors du 1er passage, pour un coût C1, et le coût du même test lorsqu'il est ré-exécuté lors des non-régressions, pour un coût C2. Avec un moniteur de tests, le coût C2 sera très inférieur à C1.
La situation lors du passage du dernier test peut être décrite par le tableau suivant:

T1						
T1	T2					
T1	T2	T3				
...	...	...	...			
T1	T2	T3	...	Ti		
...	...	...	...	Ti	...	
T1	T2	T3	...	Ti	...	Tn
n fois	n-1 fois			n-i+1 fois		1 fois

La diagonale du tableau correspond au coût du 1er passage C1, alors que la partie inférieure correspond au coût C2.
a) En vous servant de la formule (F1), calculer le bilan de l'opération (c'est à dire une formule du type N1×C1 + N2×C2, dans laquelle il faut évaluer N1 et N2 en fonction de n).
b) En prenant C2 = 0.1×C1, calculer le nombre n de tests à partir duquel le moniteur de tests devient rentable.
c) Calculer les gains escomptés en prenant comme valeurs de n le nombre de tests unitaires, puis le nombre de tests d'intégration.

Le coût total peut s'écrire :

$$CT = n \times C1 + \frac{1}{2} \times (n-1) \times n \times C2$$

Si l'on n'a pas d'outil de non régression, le coût CT devient :

$$CT_1 = \frac{1}{2} \times n(n+1) \times C1$$

L'outil de non régression est rentable dès que n est tel que : $CT_1 > CT$
Avec C2 = 0.1×C1 la formule ci-dessus devient :

$$\frac{1}{2} \times n(n+1) \times C1 = n \times C1 + \frac{1}{2} \times (n-1) \times n \times 0.1 \times C1$$

soit, en simplifiant : n > 1
Dès le 2ème test, l'opération est rentable.

La formule du gain peut s'écrire :

$$G = \frac{1}{2} n(n+1) \times C1 - [n \times C1 + \frac{1}{2} n(n-1) \times C2]$$

Soit avec C2 = 0.1×C1 :

$$G = \frac{0.99 \times C1}{2} \times n(n-1)$$

En appliquant cette formule à TU, puis à TI, on obtient :

$$G_1 = \frac{0.99 \times 0.5}{2} \times 85(85-1) = 1767 \text{ hj} = 84 \text{ hm}$$

$$G_2 = \frac{0.99 \times 3}{2} \times 49(49-1) = 3493 \text{ hj} = 166 \text{ hm}$$

Soit au total un gain de 250 hm.
Ce gain sert à amortir l'outillage nécessaire et à développer de nouveau tests.

EXAMEN DE GÉNIE LOGICIEL
Cours BO
session du 12 avril 1996

Tous documents et calculettes sont autorisés.

QUESTION DE COURS: Analyse d'un projet de développement logiciel.

CORRIGÉ

Les corrections sont en *italiques*.

Le responsable du projet MOT6 de la société LOGICIEL-MICRO a établi le devis du développement du logiciel MOT6 sur les bases suivantes:

— Conception générale du produit (durée environ 4 mois);
— Codage (durée environ 12 mois):
— Qualification (durée environ 2 mois);

suite aux recommandations du directeur technique de LOGICIEL-MICRO. Ce dernier, pour assurer une bonne rentabilité du projet, estime que 10 personnes, en moyenne, sont amplement suffisantes pour une durée de développement de 18 mois (NB : ce qui inclut obligatoirement des congés).

La montée en puissance de l'équipe est programmée comme suit:

— à T_0 : 4 analystes architectes, dont le chef de projet;
— à $T_0 + 4$ mois : passage à 10 personnes;
— à $T_0 + 6$ mois : complément d'effectif pour produire effectivement le budget alloué (qualification du produit).

QUESTION N° 1 : Quel est le budget (en Hommes-Mois) alloué pour ce développement. Dessiner sur un diagramme de charge la montée en puissance des effectifs en fonction du temps calendaire; en particulier, calculer l'effectif à $T_0 + 6$ mois. Selon ce diagramme, à quelle date se termine la conception du produit ? À quelle date la programmation devrait-elle être achevée (vous pouvez utiliser le Vade mecum du chef de projet qui est dans le cours) ?

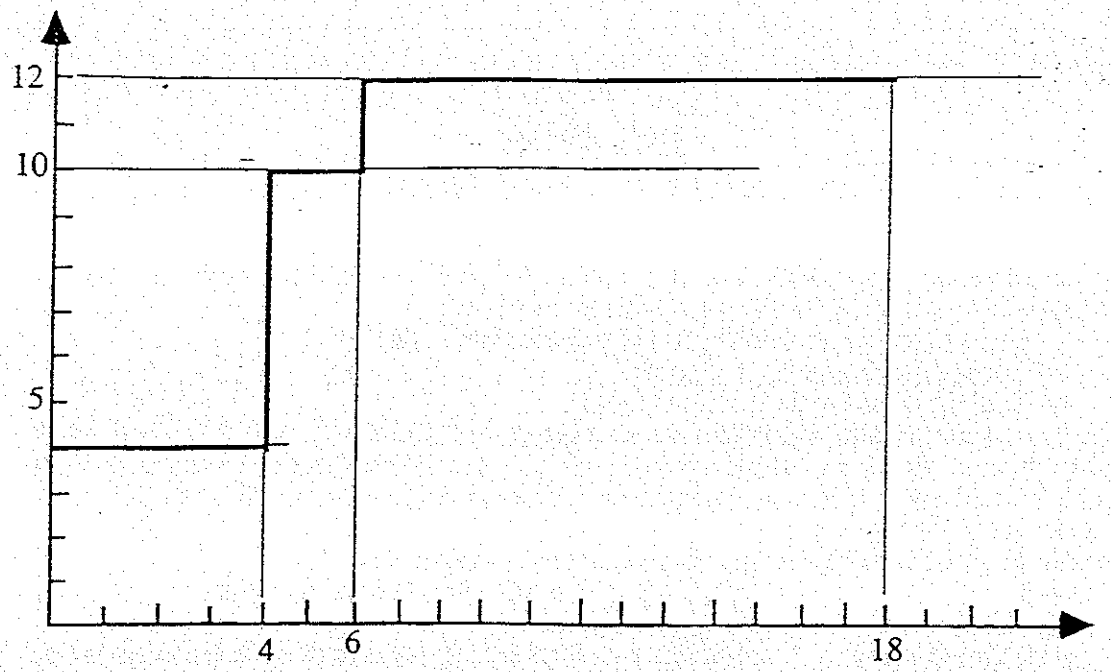
Le budget brut est de 18 Mois x 10 Personnes = 180 HM. Mais il faut retrancher les vacances de 10 personnes; le budget net est donc de 170 HM.
Soit x l'effectif à T0 + 6; on a donc:

$$4\,Mois \times 4\,Personnes + 2M \times 10P + 12M \times x = 170HM,\ soit:\ x = \frac{170 - 16 - 20}{12} = 11.2$$

L'effectif doit donc passer à 12 personnes.

La conception du produit doit être terminée à $T_0 + 4$. au moment de la première montée en puissance. En utilisant le Vade mecum. la programmation doit être finie à mi parcours. soit à $T_0 + 9$ mois.

Diagramme des effectifs :



La conception du produit doit être terminée à $T_0 + 4$, au moment de la première montée en puissance. En utilisant le Vade mecum, la programmation doit être finie à mi parcours, soit à $T_0 + 9$ mois.

QUESTION N° 2 : En vous servant du modèle d'estimation COCOMO joint en annexe, vous allez vérifier la cohérence des données de ce projet par rapport au modèle COCOMO.
Pour simplifier les calculs, on considérera que le logiciel est entièrement de type S (c'est à dire ***Organic*** dans le franglais COCOMO); on utilisera les tables 5-2 ou 6-8 pour obtenir le découpage de l'effort et de la durée conformément aux phases du modèle; on considerera que la taille du produit est 32 KLS pour éviter des calculs fastidieux d'extrapolation linéaire entre 2 colonnes.
Calculer les efforts et la durée de chacune des phases. Reporter ces phases et les durées correspondantes sur le diagramme des effectifs de la question 1. Calculer le nombre de lignes source prévisibles.

La table 5-2 est suffisante. Il faut toutefois remarquer que l'ensemble Plan + Développement est égal à 106%, pour un effort total de 170HM. La ventilation par phases est donc:

Plan et expression de besoin :	*(170 / 1.06)×6%*	*= 9.6HM*
Conception :	*(170 / 1.06)×16%*	*= 25.7HM*
Programmation :	*(170 / 1.06)×62%*	*= 99.4HM*
Intégration et test :	*(170 / 1.06)×22%*	*= 35.3HM*

La formule de durée de COCOMO nous donne: $TDEV = 2.5\times(170)^{0.38} = 17.6$ mois.
Les durées par phases sont donc :

Conception :	*17.6×19% = 3.3 mois*
Programmation :	*17.6×55% = 9.7 mois*
Intégration et test :	*17.6×26% = 4.6 mois*

Le nombre de lignes source s'obtient en résolvant l'équation $170 = 2.4\times(x)^{1.05}$.

soit en passant aux logarithmes: $Log(x) = \dfrac{Log\dfrac{170}{2.4}}{1.05} = \dfrac{4.3}{1.05} = 4.1$

On a donc : $x = e^{4.1} = 57.8 \equiv 58$ *KLS, soit 58.000 lignes de code source.*