Il est par contre indispensable de retoucher fortement les charges en tests, à partir de l'effort de codage : pour faire simple j'ai pris 50%, alors que les modèles d'estimation conseillent plutôt 100%. J'ai fait l'hypothèse que, malgré tout, le codage comportait tout de même des tests !
L'effort d'intégration est du même ordre que l'effort de conception. En prenant 100% de l'effort indiqué, on a un minorant car en toute rigueur il faudrait intégrer une partie de l'effort venant du prototype. Quand la valeur est aberrante, j'ai donné une valeur plausible, sans charger la barque (i.e. 50% de l'effort de codage) !

	Charges *en h.j*	Estimation des manques en test	Estimation des manques en intégration	
			Minimum	Moyenne probable
Evaluation Lot 1		285		
Conception :	42	NB. : Chiffre très conservatoire car l'IHM a été entièrement refaite !		
Codage :	360			
Test :	75		42	180
Recette et corrections :	50			
Coordination et AQ :	52			
Evaluation Lot 2			45	140
Conception :	45			
Codage :	279			
Test :	...			
Recette et corrections :	45		...	
Coordination et AQ :	16			
Evaluation Lot 3		NB : Le chiffrage de ce lot est surprenant car c'est l'intégration de l'IHM et de la BTV. Il devrait y avoir 0 en codage et beaucoup en test, recette et corrections ! ! !	50	50
Conception :	3			
Codage :	50			
Test :	3			
Recette et corrections :	3			
Coordination et AQ :	4			
Evaluation Lot 4		160	15	102
Conception :	15			
Codage :	205			
Test :	45			
Recette et corrections :	30			
Coordination et AQ :	30			
Evaluation Lot 5		97	12	65
Conception :	12			
Codage :	130			
Test :	33			
Recette et corrections :	20			
Coordination et AQ :	18			
Evaluation Lot 6		NB : Le chiffrage de ce lot est surprenant car c'est essentiellement de l'intégration	80	80
Conception :	3			
Codage :	160			
Test :	16	144		
Recette et corrections :	...			
Coordination et AQ :	18			

Evaluation Lot 7 Conception : Codage : Test : Recette et corrections : Coordination et AQ :	 50 135 ... 30 7	125	50	70
Evaluation Lot 8 Conception : Codage : Test : Recette et corrections : Coordination et AQ :	 ... ... 65 10	NB : Ce lot correspond à **l'intégration complète** de CEREX V1, soit 3,1% du total de l'estimation. ➘ Très forte sous-estimation.	161(*1)	323(*1)
Evaluation Lot 9+10 Conception : Codage : Test : Recette et corrections : Coordination et AQ :	 1 15 3 0+10 2	NB : Ces 2 lots devraient n'en former qu'un seul, par homogénéité avec les autres lots. La somme est vraisemblable.		
Bilan analytique des sous estimations	2421j	811j	455	1010
Bilan total des sous estimations	= 115 h.m	Fourchette haute : 1821j, soit : 87 h.m Fourchette basse : 1266j, soit : 60 h.m		

NB. : on prend 1 h.m = 21 jours.

Compléments et oublis divers

Intégration globale (*1), Cf. Lot N°8

Le coût d'intégration global a été estimé forfaitairement à 5 ou 10% (ce qui est inférieure à ce qui est couramment observé en moyenne dans de nombreux projets, i.e. ≈20%) de l'estimation par lot qui représente un effort de 2421j auquel on a rajouté les oublis en tests, soit un effort de 2421+811 = 3232, ce qui conduit à la fourchette [161j ; 323j] indiquée.

Impact de la livraison d'une version intermédiaire

On peut assimiler le coût minimum d'une livraison intermédiaire à une intégration supplémentaire. En prenant comme référence le total réestimé, soit [175 h.m ; 202 h.m] correspondant à la fourchette ci-dessus, il est vraisemblable d'estimer ce coût entre [35 h.m ; 40 h.m] ; la 1ère intégration est plus coûteuse que la seconde.

Analyse des estimation en recette – corrections

Lorsque l'on donne une estimation de *xj* en recette-corrections, on fait implicitement une hypothèse de calcul sur le nombre de corrections effectuées, et sur la non régression de ces corrections.

Si l'on prend, à titre d'exemple, les 50j de recette – corrections du lot 1, cela correspond à 50 corrections si l'on compte 1j de travail par correction, ou à 100 corrections si l'on compte ½ j par correction. Une correction significative nécessite un passage de tous les tests déjà acquis si l'on souhaite garantir que l'on a pas régressé en qualité. Si l'on dispose de *n* tests pour effectuer ces

non régressions, cela oblige à exécuter les tests $\frac{n\times(n+1)}{2}$ fois, ce qui prend beaucoup de temps si les tests sont manuels (c'est le cas des IHM). Une autre façon de décrire les effets des non régression est de dire que le nombre de scénarios de tests produit croit comme la racine carrée de l'effort de test, soit :

$$Nombre\ de\ tests = K \times \sqrt{Effort\ de\ test}$$

Estimation du nombre de corrections à partir de la volumétrie de CEREX

A défaut de la volumétrie réelle de CEREX que l'on obtiendra peut être avant la fin de cette étude, on peut faire le raisonnement suivant :

A. En programmation pure (i.e. un source compilé sans erreur de compilation, hors conception et test) la productivité moyenne d'un programmeur est de l'ordre de 100 à 125 LS par jour ; ce qui correspond à une productivité annuelle de LS documentées et testées aux alentours de 5.000 LS conforme aux modèles d'estimation généralement admis par la profession.

B. Pour le lot N°1 qui compte 360 jours de programmation on peut s'attendre à un volume de programmation aux alentours de 40.000 LS, soit un nombre d'erreurs probables à corriger par les tests unitaires et les tests d'intégration aux alentours de 4.000 erreurs (en faisant l'hypothèse qu'il y a une erreur toutes les 10 à 15 lignes).

C. En admettant que l'on corrige les erreurs par paquet de 10 (ce qui est certainement une hypothèse optimiste avec une équipe de programmeur débutant sans expérience de l'application), on voit que l'on peut s'attendre à environ 400 corrections lourdes qui nécessiteront des non régressions afin de s'assurer que les corrections effectuées n'introduisent pas de nouvelles erreurs. Par rapport aux 75 jours de tests annoncés dans le lot N°1, on voit bien qu'il y a incohérence, ce qui est une autre façon de mettre en évidence la sous-estimation chronique du projet.

Question N°2 : En utilisant les données projet du lot N°1 calculer le volume de programmation probable correspondant aux 360 jours qui ont été alloués à cette tâche. On conviendra que la tâche de programmation correspond à l'écriture de programmes qui sont compilés sans erreur. L'activité de test est comptabilisée dans les rubriques Test et Recette/Corrections. Justifier la productivité de la programmation en utilisant les données COCOMO ou le modèle rudimentaire du Vade-mecum du chef de projet.
A défaut, en fonction de votre propre expérience, indiquez le volume de programmation que vous êtes capable de produire en 360 jours de travail normal, i.e. 8h par jour.

Réponse

Sur la base du Vade-mecum, en prenant 4KLS par HA, selon une répartition 40-20-40, on obtient :
$\frac{360}{220}\times 4\times\frac{100}{20} = 32.7 KLS$, soit 91LS par jour de programmation (compilée sans erreur), ce qui n'est pas si mal. Voir ci-dessus B.

Question N°3 : Le volume de programmation estimé ci-dessus comporte un certain nombre d'instructions (IF, CASE OF, DO WHILE ou UNTIL, etc.) qui altèrent le flot de contrôle quand le programme s'exécute. Ces instructions permettent de construire le graphe de contrôle du programme. Pour simplifier les calculs, on considérera qu'il y a en moyenne une instruction de contrôle toutes les 10 instructions. Quel est le nombre moyen d'instructions de contrôle ?

Sachant qu'une instruction de contrôle contient une expression conditionnelle qui manipule en moyenne 4 variables (par exemple dans des expressions comme IF A(I)=B(J) ...), combien faudrait-il écrire d'instructions d'assignation pour faire varier toute les conditions au moins une fois dans le cas vrai et dans le cas faux ? Expliquez vos hypothèses de calcul.
Un tel programme constitue un programme de test associé au programme initial qui lui a donné naissance (c'est d'ailleurs ce que vous écrivez quand vous utilisez un debugger symbolique pour forcer les variables de vos programmes !). Le texte obtenu est structuré en bloc de 100 lignes (i.e. une double page) en moyenne constituant 1 test au sens habituel du terme.
Combien obtient-on de tests avec une telle méthode de construction des tests ?

Réponse

Nbre d'instructions de contrôle : $\frac{32700}{10} = 3270$

Si il y a 4 variables, il faut 4 assignations pour calculer une condition ; pour faire varier la condition, et obtenir le cas VRAI et le cas FAUX, il faut modifier une des assignation, soit au total 5 instruction différentes. Soit $3270 \times 5 = 16350 LS$ pour faire varier correctement les conditions.
Notons que pour être vraiment complet, il faudrait rajouter un cas ou la condition est incohérente, soit entre 1 et 4 instructions (ceci correspond à des tests de robustesse). Si l'on part sur une moyenne de 2, il faudrait rajouter 6540 instructions d'assignation.

Le nbre de test est donc de 163 dans le cas V/F et de 228 dans le cas V/F/Incohérent

Question N°4 : Le nombre de tests calculé ci-dessus est jugé suffisant pour amener le programme à un niveau de qualité satisfaisant.
Si l'on considère qu'en fin de programmation (au sens précisé ci-dessus) il y a en moyenne 15 erreurs par millier de lignes source, le nombre de tests vous paraît-il suffisant ? Justifier votre réponse.

Réponse

Le nbre d'erreurs probable est de 32.7×15=487, pour un nbre de tests de 163, soit $\frac{487}{163} \approx 3$ erreurs par test, ce qui semble plausible.
Notons que les variables manipulées dans les conditions le sont également dans le code environnant la condition. Elles contribuent donc également à vérifier ce code, mais probablement pas exhaustivement.

Question N°5 : Combien de fois faut-il exécuter les tests pour assurer la non régression du programme à tester suite aux modifications effectuées sur ce programme.
Si les résultats des tests doivent être dépouillés et analysés manuellement, quel est l'effort correspondant (prendre une durée de 1 heure pour un dépouillement de 1 tests, ou une autre valeur qui vous paraîtrait plus raisonnable mais que vous justifierez) à un passage complet de tous les tests avec une non régression.
NB. Le nombre de passage est une somme S=1+2+3+...+n, n étant le nombre de tests disponibles.

Réponse

En appliquant brutalement la formule, on obtient $\frac{163\times164}{2}=13366$ passages, soit, en HM un effort de $\frac{13366}{8\times21}=80\,\text{HM}$ qui constitue un maximum (NB : c'est plus que l'effort de programmation !).

Si l'on regroupe les tests par paquet de 10, pour ce qui concerne la non régression, on peut diviser le résultat précédent par 100, soit 0.8HM, auquel il faudra rajouter le dépouillement de tous les tests au moins une fois, soit : $\frac{163}{8\times21}\approx1\,\text{HM}$. Si l'on prend comme hypothèse que le paquet de 10 tests nécessite 10 heures de dépouillement, il faut multiplier le résultat précédent par 10, soit : 8HM de dépouillement.

Au total, on aurait, selon cette hypothèse, un effort de 11HM pour le dépouillement des tests.

Ces calculs sont à rapprocher de ceux de la question N°1 qui faisait ressortir une sous estimation de $285+42=327\approx16HM$ ou $285+180=465\approx23HM$; le modèle analytique n'est pas en contradiction avec le modèle synthétique.

Question N°6 : A partir du programme de tests élaboré à la question 3, que faudrait t-il faire pour rendre le dépouillement des résultats automatique (NB. : ce que vous vérifiez à l'œil nu lors du debugging du programme peut être également programmé, ce qui augmentera d'autant le volume du programme de test). Estimer l'accroissement de taille du programme de test.

Selon vous quelle est le choix le plus raisonnable, dépouiller manuellement, ou rendre le programme de test automatique ? Expliquer et justifier votre réponse.

Réponse

Les assignations comptabilisées précédemment sont intégrées aux programmes à tester sous la forme d'auto-tests (une pratique recommandée dans le cours !) ; Il faut rajouter une instruction de comparaison supplémentaire pour valider que le code du programme initial, et celui de l'auto-test fournissent des résultats identiques : en cas de désaccord, il faut émettre un message et/ou une trace pour signaler l'incohérence, soit une instruction de comparaison et une trace dans chaque cas, soit 4 instruction par instruction de contrôle. Ceci a pour effet de faire passer le volume ce code à programmer de 32700LS à 32700+16350+ 13080=62130LS. Notons que le programme à tester constitue la documentation de ce code qui n'a donc pas besoin d'être documenté en tant que tel (ce code additionnel est par définition de type S). Pour la fabrication de ce code, on est en droit de prendre des hypothèses de productivité très élevée (8 a 12 KLS par HA), surtout si ce code est écrit en même temps que le code nominal et par la même personne (ce qui peut poser quelques problèmes pour des logiciels critiques, mais rien n'empêche de le faire programmer par un autre programmeur, évidemment au prix d'une productivité moindre).

Un tel programme est dit auto vérifiant puisqu'il contient le code nominal qui assure la fonction à programmer, et les tests qui la vérifie. Son dépouillement devient alors linéaire. Il est donc judicieux de rendre le programme auto-vérifiant, chaque fois que c'est possible.

EXAMEN DE GÉNIE LOGICIEL
Cours B5

Session II 5 juin 1999
Arcueil

LA QUESTION DE COURS ET L'EXERCICE DOIVENT IMPERATIVEMENT ETRE TRAITES SUR DEUX COPIES DISTINCTES.

Prenez le temps de bien lire l'énoncé.

Tous documents et calculettes sont autorisés.

Barème :

Question de cours 12 points

- Q1 3
- Q2 4
- Q3 2
- Q4 3

Bonus

- Q5 2
- Q6 1
- Q7 1

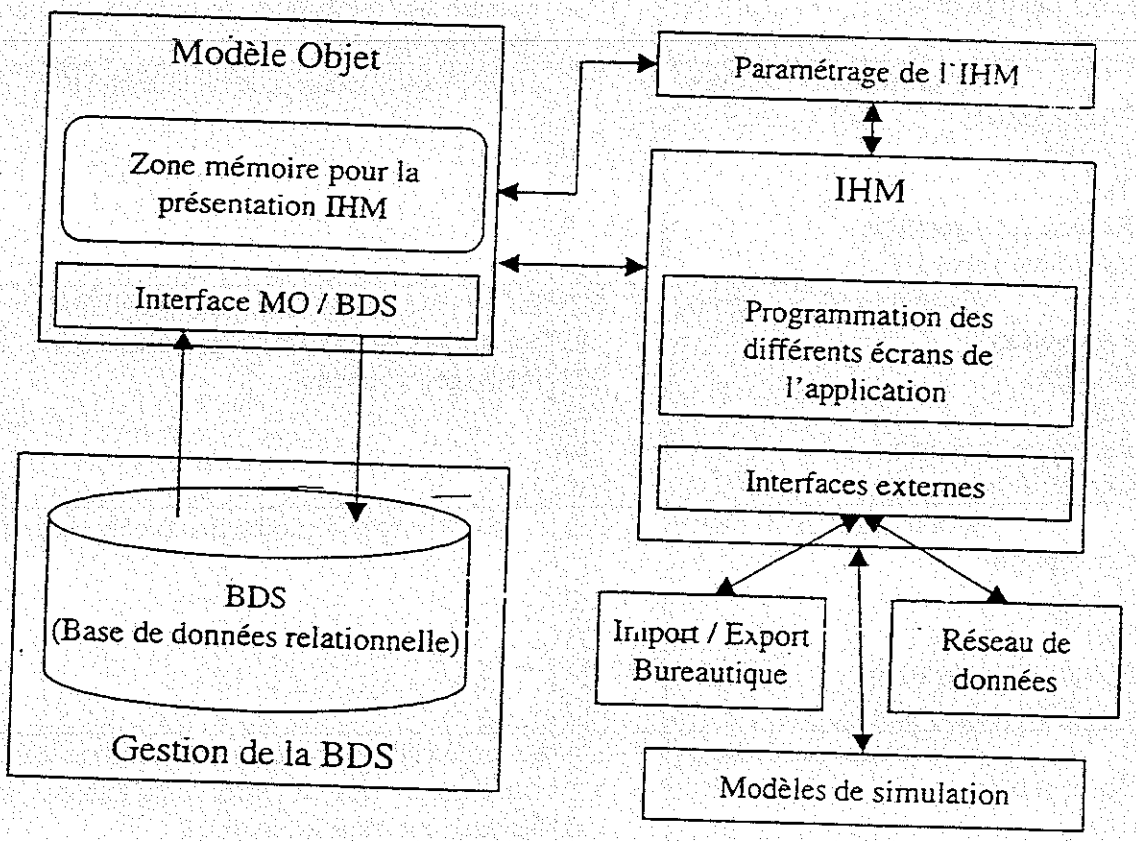
Cours B5 - Contrôle de Génie Logiciel

2ème session

Etude de cas

La société Power Inc. qui agit comme maître d'ouvrage souhaite industrialiser un prototype d'une application destinée à effectuer des prévisions de consommation électrique. L'application est considérée comme vitale du point de vue économique, car son bon fonctionnement évitera des investissements en équipements lourds très coûteux (plusieurs milliards de F). La duré de vie prévisible de l'application est de 15-20 ans.

L'architecture de l'application est donnée par le schéma suivant :



L'IHM et le Modèle Objet forment un bloc fonctionnel IHM/MO constituant le lot de réalisation N° 1. Le modèle objet est un programme de gestion mémoire destiné à préparer les données nécessaires aux différents écrans et aux modèles de calcul en FORTRAN utilisés pour la simulation. L'application est programmée en C++. Le SGBD de la base de données est ORACLE.

84

Les données de la gestion de projet fourni par le maître d'œuvre industriel sont les suivantes :

	Charges *en h.j*	Descriptif du lot
Evaluation Lot 1 (*1), dont :	692	Le lot 1 correspond à l'industrialisation du prototype de l'IHM réalisé sur station UNIX. La version industrielle est réalisée sous Windows NT. Elle est entièrement reprogrammée par une nouvelle équipe d'ingénieurs débutants ; seul le responsable de l'équipe qui est également le chef de projet a participé à la réalisation du prototype.
Conception :	42	
Codage :	360	
Test :	75	
Recette et corrections :	50	
Coordination et AQ :	52	
Evaluation Lot 2, dont :	441	Le lot 2 correspond à l'industrialisation de la base de données de simulation (BDS). La conception de cette base est identique à celle du prototype (le modèle conceptuel est le même). Seule l'interface avec l'IHM est nouveau. La version industrielle doit prendre en compte des contraintes de fiabilité, de performance et de maintenabilité.
Conception :	45	
Codage :	279	
Test :	...	
Recette et corrections :	45	
Coordination et AQ :	16	
Evaluation Lot 3, dont :	69	Le lot 3 est l'intégration de l'IHM/MO et de la BDS
Conception :	3	
Codage :	50	
Test :	3	
Recette et corrections :	3	
Coordination et AQ :	4	
Evaluation Lot 4, dont :	380	Le lot 4 réalise un complément d'IHM, en particulier en terme de paramétrage, d'aides en ligne et de messages d'erreurs.
Conception :	15	
Codage :	205	
Test :	45	
Recette et corrections :	30	
Coordination et AQ :	30	
Evaluation Lot 5, dont :	263	Le lot 5 intègre de nouvelles fonctions de simulation.
Conception :	12	
Codage :	130	
Test :	33	
Recette et corrections :	20	
Coordination et AQ :	18	
Evaluation Lot 6, dont :	220	Le lot 6 réalise des fonctions d'export des résultats de simulation vers d'autres applications et vers la bureautique pour les éditions.
Conception :	3	
Codage :	160	
Test :	16	
Recette et corrections :	...	
Coordination et AQ :	18	
Evaluation Lot 7, dont :	247	Le lot 7 réalise des fonctions de mise en librairies de façon à gérer les contextes et les données de travail nécessaires à la simulation, ainsi que des sauvegardes.
Conception :	50	
Codage :	135	
Test :	...	
Recette et corrections :	30	
Coordination et AQ :	7	

Evaluation Lot 8, dont : Conception : Codage : Test : Recette et corrections : Coordination et AQ :	75 ... ... 65 10	Le lot 8 est un lot de validation globale l'application.
Evaluation Lot 9+10, dont : Conception : Codage : Test : Recette et corrections : Coordination et AQ :	34 1 15 3 10 2	Les lots 9 et 10 est un petit complément qui intègr une fonction nouvelle dans les librairies (c'est u complément du lot 7).
Total de l'estimation en jours	2421j	
Bilan des sous estimations		Le bilan des sous estimations par lot, a fait l'objet de l'examen de la 1ère session. La sous-estimation initiale est de l'ordre de 70 hm

NB. : pour les conversions en homme.mois (hm) on prendra 1 mois = 21 jours, pour les conversions en homme.an (ha) on prendra 1 an = 220 jours.

Dans cette étude de cas nous nous proposons d'analyser la volumétrie de ce projet.

Question N°1 : Analyse de la volumétrie
La réalisation du projet a été confiée à deux sociétés S1 et S2.

S1 assure la maîtrise d'œuvre industrielle du projet et réalise également les lots :
N°1, 3 (mais seulement à 60%), 4, 5, 6, 7, 8 (mais seulement à à 46%), 9 et 10.

La société S2 réalise le reste (tout ce qui concerne la ~~BDS~~).

Aucune intégration intermédiaire ne figure dans le plan projet (et n'a d'ailleurs pas été demandée par le maître d'ouvrage Power Inc.).
La réalisation a été précédée d'une phase de maquettage/prototypage de chacun des éléments IHM/MO et BDS, mais les maquettes correspondantes n'ont jamais été intégrées. La société S1 a complètement renouvelé l'équipe qui avait réalisé la maquette IHM/MO sur UNIX avec une équipe d'ingénieurs débutants dont c'est la première réalisation industrielle ; de plus, la responsabilité globale du projet a été confiée au responsable du projet IHM/MO (qui était auparavant le responsable de la maquette IHM/MO). Le responsable du projet BDS a conservé l'essentiel de l'équipe maquette dont il pense réutiliser une grande partie du code source.
Après six mois de retard, le maître d'ouvrage Power Inc a fait réaliser un audit dont les données de volumétrie sont les suivantes :

- Volume de code réalisé par S1 : 198.532 LS, que l'on arrondit à 200 KLS.
- Volume de code réalisé par S2 : 59.000 LS.

Pensez-vous que l'effort consenti pour la fabrication du maquettage/prototypage a été correctement rentabilisé ? Justifier votre réponse par des arguments de bon sens.

Question N°2 : En utilisant les informations sur les maquettages/prototypages de la pré-étude et la maturité des deux équipes, recalculer les données du projet (effort et délai) à partir de la volumétrie. Vous pourrez utiliser indifféremment le vade-mecum et/ou COCOMO. Justifier le type S/P/E que vous choisirez (ou la combinaison). Comparer ces résultats avec ceux du tableau correspondant au devis initial (NB : c'est à dire combien de lignes de code documentées et testées peut-on espérer développer avec 2421j + 70hm).

Question N°3 : Comment estimer le coût d'une intégration intermédiaire représentant les 2/3 de la programmation totale ? Une telle intégration vous paraît-elle souhaitable ? inutilement coûteuse ? Dans tous les cas justifiez votre analyse.
Quel est le surcoût occasionné par l'intégration de 259 KLS par rapport à deux projets de 200 et 59 KLS complètements disjoints ?

Question N°4 : Politique d'assurance qualité.
Compte tenu de l'importance de l'application, comment doit (ou pourrait !) être organisée l'assurance qualité d'un tel projet. Vous prendrez soin de bien faire apparaître les actions concernant les caractéristiques fonctionnelles, et celles concernant les caractéristiques non fonctionnelles compte tenu du niveau de criticité de l'application. Essayer de chiffrer approximativement son coût en raisonnant sur les données connues que vous pourrez compléter par des arguments de bon sens.

Question N°5 : que doit faire le maître d'ouvrage Power Inc pour être en mesure de contrôler la situation ? Donner des mesures concrètes et de bon sens facile à mettre en œuvre par le maître d'ouvrage.

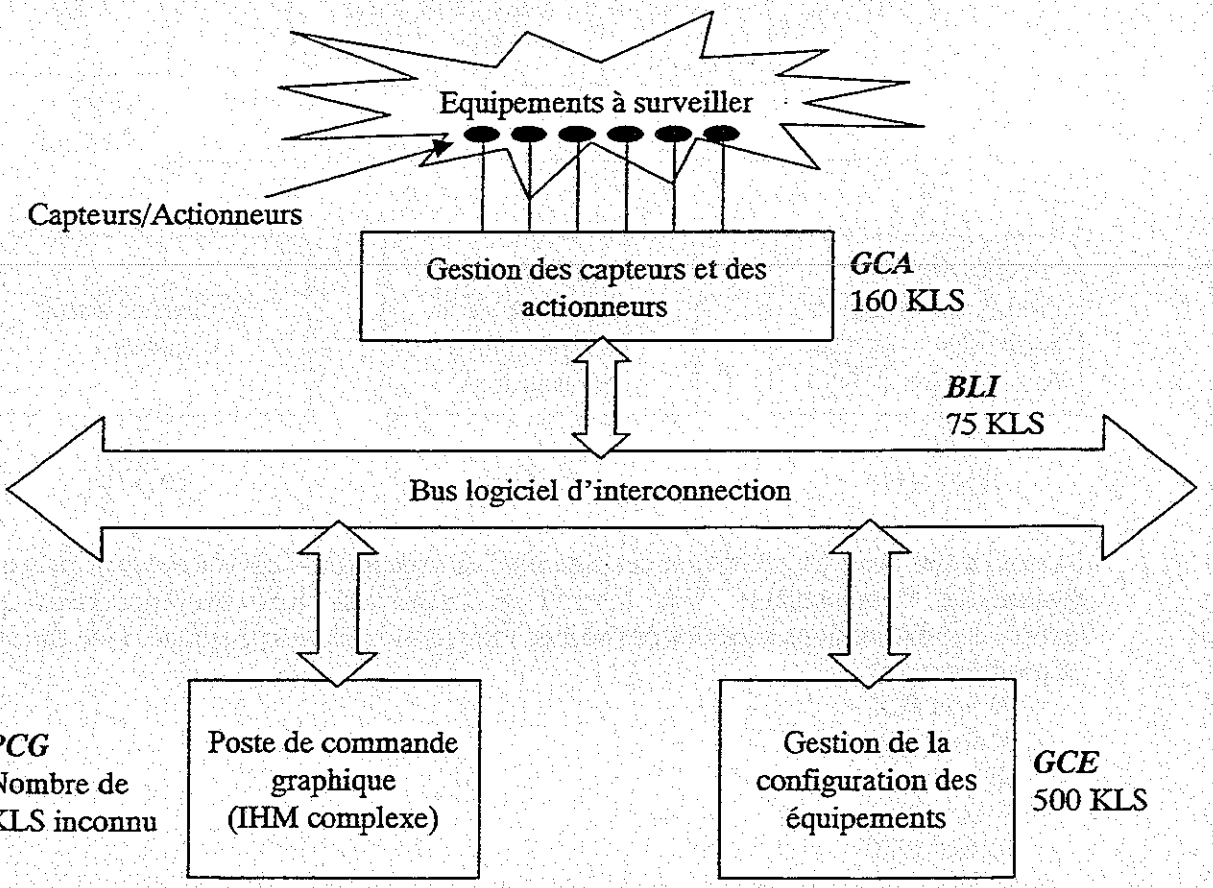
Question N°6 : que doit faire le maître d'œuvre industriel (la société S1) vis à vis de ses propres personnels, et de l'équipe de la société S2. Que doit voir le maître d'ouvrage à partir des résultats des actions qualité entreprises par le maître d'œuvre.

Question N°7 : comment le maître d'ouvrage doit-il organiser la recette de l'ensemble (volume et type de tests à prévoir, coûts de développement et d'exécution des tests, durée de la recette, etc.) ?
NB : pour cette question, ne pas faire de longs calculs ! Raisonner avec votre bon sens, en justifiant vos recommandations.

Cours B5 - Contrôle de Génie Logiciel

Etude de cas

La société Power S.A réalise un système informatique destiné à la surveillance et au contrôle-commande d'un ensemble d'équipements permettant la distribution de l'énergie (SC2DE).
Le système est architecturé comme suit :

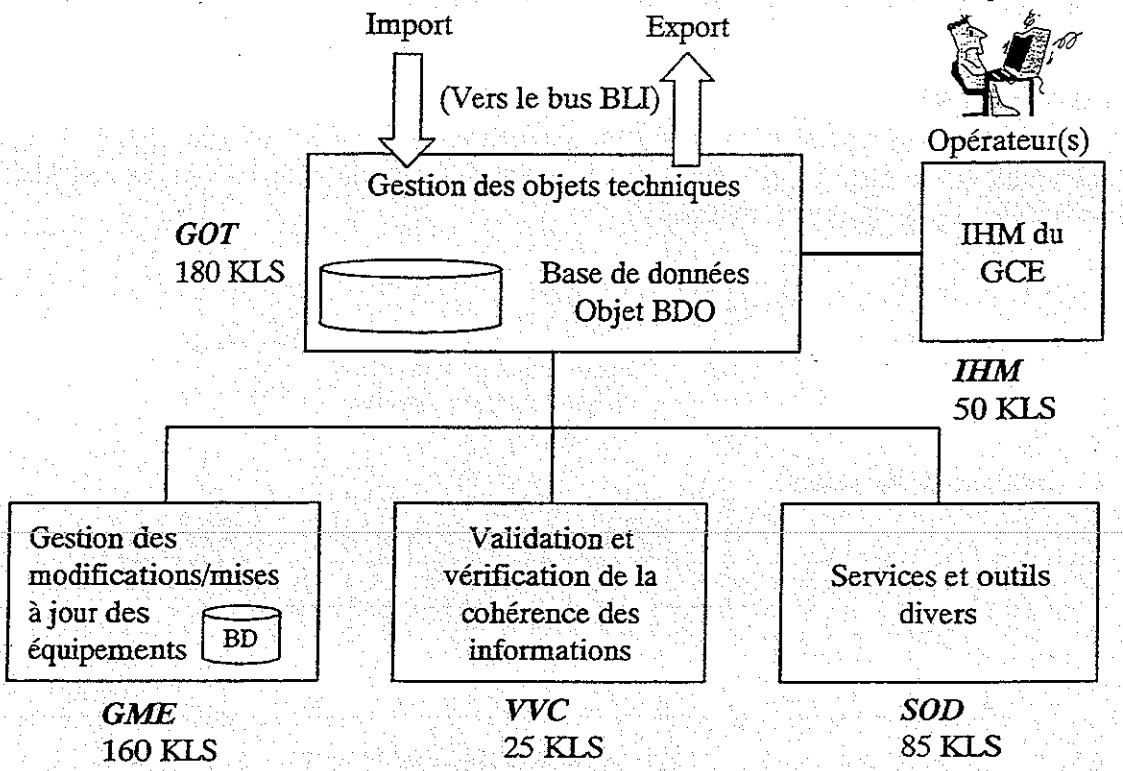


Module GCA
Le GCA est un ensemble temps réel qui doit garantir des temps de réponses de l'ordre de la seconde. Il a été réalisé à partir d'un portage à l'identique d'un système plus ancien afin de pouvoir utiliser une plate-forme plus moderne (unix, C++, etc.). Le portage garantit la reproduction à l'identique du système ancien (y compris des défauts qu'il peut encore contenir).

Module BLI
Le BLI est un "bus" logiciel permettant d'interconnecter, au moyen de librairies, différentes applications et/ou sous-systèmes permettant la gestion des équipements à surveiller. Le bus peut fonctionner sur une seule machine ou dans un environnement distribué (architecture en client-serveur). Il assure des fonctions de stockage temporaire. Le bus utilise le produit ORBIX dérivé des normes CORBA de l'OMG; le produit est encore assez instable mais ses fonctionnalirtés sont attrayantes.

Module GCE

La GCE est un sous système relativement complexe qui permet de configurer globalement le système SC2DE en fonction des équipements gérés.Sa structure est la suivante :



La qualité des données gérées par cet ensemble est primordiale car toute erreur peut entrainer des destructions d'équipements. Une session de travail effectuée par l'opérateur necéssite de nombreuses interactions entre les différentes fonctions de ce sous-système. Une session typique a un profil comme suit :

- 100 fois IHM
- 300 fois GOT
- 50 fois GME
- 100 fois VVC
- 50 fois SOD

Une panne du GCE n'est pas considérée comme critique car elle n'empêche pas les équipements d'être correctement surveillés, mais peut-être pas de façon optimale.

Module PCG

Le PCG est un sous système graphique qui permet de visualiser en temps réel l'état des équipements. Pour des raisons de sûreté de fonctionnement l'informatique qui assure le fonctionnement du PCG est doublée. Vis à vis du système SC2DE le PCG se présente comme une boîte noire qui a fait l'objet d'une sous-traitance mais dont Power SA assure l'intégration.

Le système SC2DE est destiné à être installé et supporté sur environ 150 sites d'exploitation, dont chacun aura une configuration qui lui est propre compte tenu des équipements gérés.

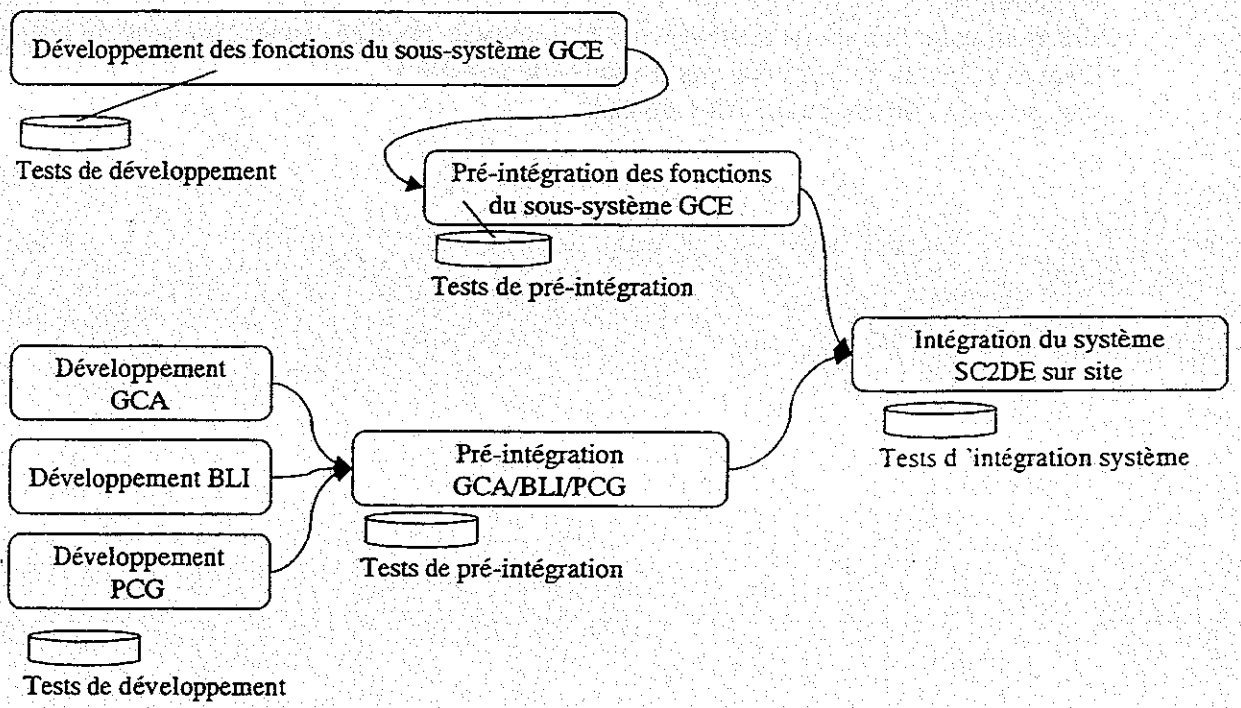
Question N° 1 (2 points)

Sans entrer dans le détail, quelles sont les caractéristiques S, P, E de chacun des éléments du système SC2DE. Pour le GCE vous pouvez utiliser le profil d'exécution. Justifiez brièvement vos réponses.

Question N° 2 (4 points)
En vous servant, soit du vade-mecum du chef de projet, soit du modèle COCOMO, reconstituez le profil de développement de chacun des éléments du système (GCA, BCI et GCE). Donnez une estimation de l'effort d'intégration de l'ensemble.

Question N° 3 (4 points)
Pour intégrer un ensemble aussi complexe, plusieurs scénarios sont possibles :
Scénario N°1 : GCA+BLI+PCG+GCE sous contrôle de l'équipe d'intégration
Scénario N°2 : GCA+BLI+PCG sous contrôle de l'équipe d'intégration qui livre l'ensemble à l'équipe support du site. Le support site fait l' intégration du GCE et en assure le chargement initial. Quels sont les avantages et les inconvénients de chacun des scénarios. En vous servant des résultats de la question N°2, quel est selon vous le scénario le plus économique?(NB : ne pas oublier la formation des équipes supports dans vos analyses).

Question N° 4 (2 points)
Dans le cas du scénario N°2, l'équipe d'intégration a la possibilité de fournir ses tests à l'équipe GCE qui est en retard.
Le processus d'intégration peut se schématiser comme suit :



Quels sont les facteurs de risques d'une telle situation?
(Cf. le phénomène d'amplification des erreurs dans le cours Assurance Qualité). Faut-il gérer les tests à l'aide d'un outil de gestion de configuration? Quels sont les avantages et les inconvénients d'une intégration sur site?

BONUS

Question N° 5 (2 points)

Le bus BLI est l'élément le plus critique d'un tel système. Toute panne du BLI entraîne automatiquement un coupure entre la gestion des équipements GCA et le PCG, ce qui fait que les opérateurs ne sont plus informés de la situation des équipements en temps réel.

Quelle est votre suggestion en matière de conception et de test du bus BLI.Quel est selon vous un taux d'erreurs résiduelles acceptables pour un tel module? Est-ce souhaitable d'utiliser un produit instable comme ORBIX pour réaliser le BLI?

Question N° 6 (2 points)

Fiabilité de l'ensemble GCE en fonction de la fiabilité des fonctions constitutives.

La fiabilité d'une fonction s'évalue comme un probabilité que la fonction effectue correctement la mission qu'elle assure au sein du système. Cette probabilité dépend du taux d'erreurs résiduelles. Une fonction qui est appelée *n* fois durant la même session a une fiabilité résultante f^n selon une formule classique du calcul des probabilités.

A l'aide du profil fourni, quelle est la fiabilité résultante de chacune des fonctions?

Que peut on déduire quant à la pondération de l'effort de test pour chacune de ses fonctions?

NB : vous pouvez remarquer que les fréquences sont des multiples de 50. L'effort de test est une courbe en S.

Faites un calcul pour une session en prenant comme fiabilité élémentaire f_1=0.99 et f_2=0.999;

Quelle est la valeur de f pour que la résultante soit<0.999?

Que peut-on déduire pour l'effort de test?

Cours Génie Logiciel B5 – Année universitaire 2000-2001

Corrigé de la Question de cours (12 points) – 1ère session

La société Power S.A réalise un système informatique destiné à la surveillance et au contrôle-commande d'un ensemble d'équipements (transformateurs + disjoncteurs télécommandés) permettant la distribution de l'énergie (SC2DE).

Le système est architecturé comme suit :

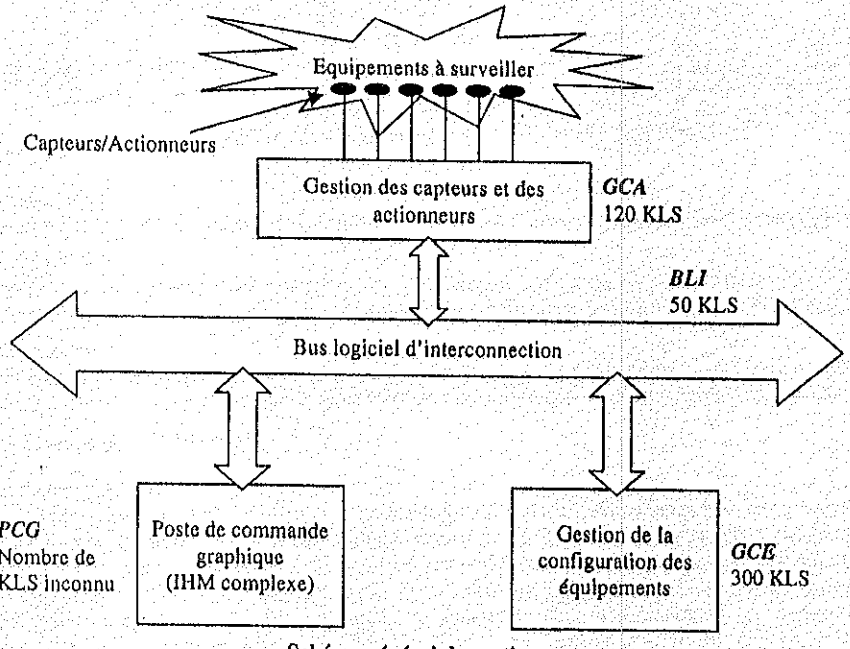


Schéma général du système

Le système SC2DE est destiné à être installé et supporté sur environ 2500 sites d'exploitation, dont chacun aura une configuration qui lui est propre compte tenu des équipements gérés qui sont différents d'un site à l'autre. Certain des équipements à la frontière des régions doivent être connus de deux sites, ce qui nécessite de dupliquer les informations qui les concernent.

Module GCA

Le GCA est un ensemble temps réel qui doit garantir des temps de réponses de l'ordre de la seconde. Il a été réalisé à partir d'un portage à l'identique d'un système plus ancien afin de

pouvoir utiliser une plate-forme plus moderne (Unix, C++, etc.). Le portage garantit la reproduction à l'identique du système ancien (y compris des défauts qu'il peut encore contenir).

La description des équipements est disponible dans la base de données du GCB ; au fur et à mesure de l'initialisation et/ou des modifications les descriptifs sont chargés dans le GCA.

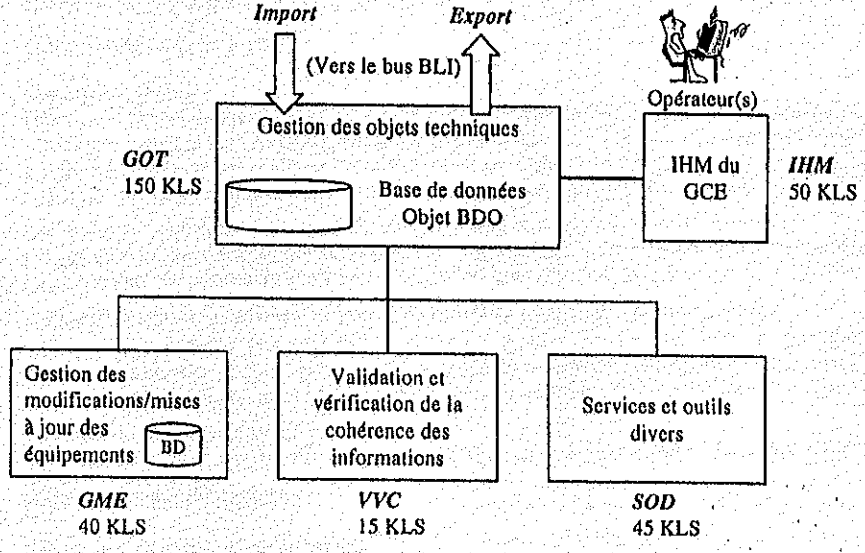
Module BLI

Le BLI est un "bus" logiciel permettant d'interconnecter, au moyen de librairies, différentes applications et/ou sous-systèmes permettant la gestion des équipements à surveiller. Le bus peut fonctionner sur une seule machine ou dans un environnement distribué (architecture en client-serveur). Il assure des fonctions de stockage temporaire. Le bus utilise le produit ORBIX dérivé des normes CORBA de l'OMG (Object Management Group) en cours de standardisation ; le produit est encore instable mais ses fonctionnalités sont attrayantes.

Le bus dispose de fonctions de stockage qui lui sont propres (traces, historique des messages, etc. permettant d'améliorer la maintenabilité de l'ensemble du SC2DE).

Module GCE

La GCB est un sous système relativement complexe qui permet de configurer globalement le système SC2DE en fonction des équipements gérés. Sa structure est la suivante :



La qualité des données gérées par cet ensemble est primordiale car toute erreur peut entraîner des destructions d'équipements. Une session de travail effectuée par l'opérateur nécessite de nombreuses interactions entre les différentes fonctions de ce sous-système. Une session typique a un profil comme suit :

- 100 fois IHM

- 300 fois GOT
- 50 fois GMB
- 100 fois VVC
- 50 fois SOD

Une panne du GCE n'est pas considérée comme critique car elle n'empêche pas les équipements d'être correctement surveillés, mais peut-être pas de façon optimale.

Module PCG

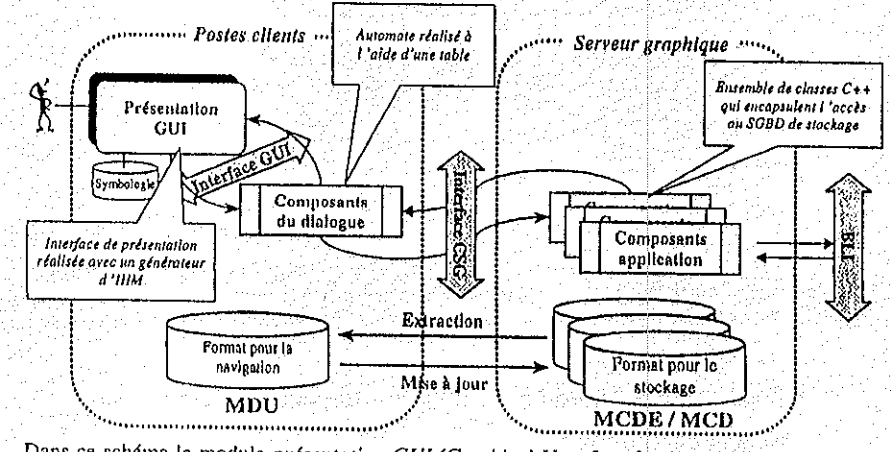
Le PCG est un sous système graphique qui permet de visualiser en temps réel l'état des équipements. Pour des raisons de sûreté de fonctionnement, l'informatique qui assure le fonctionnement du PCG est doublée. Vis à vis du système SC2DE le PCG se présente comme une boîte noire qui a fait l'objet d'une sous-traitance mais dont Power SA assure l'intégration. Le sous-traitant de Power SA n'a pas communiqué le volume source du PCG (cela n'était pas demandé dans le contrat) ; le maître d'œuvre sait toutefois qu'il s'agit de code écrit en C++ qui a été facturé 30 HA.

Question N° 1 (2 points)

Q1.1 : Sans entrer dans le détail, quelles sont les caractéristiques S, P, E de chacun des éléments du système SC2DE. Pour le GCE vous pouvez utiliser le profil d'exécution. Justifiez brièvement vos réponses.

Q1.2 : Sans entrer dans le détail, quelles sont les caractéristiques S, P, E de chacun des modules du PCG. Justifiez brièvement vos réponses. Vous pouvez utiliser les statistiques IBM ci-dessous dans l'annexe.

D'après la documentation remise par le sous-traitant qui réalise le PCG, la structure du PCG est la suivante :



Dans ce schéma le module *présentation GUI* (Graphical User Interface) est le module qui gère toutes les interactions avec l'utilisateur. Le module *composant du dialogue* gère l'enchaînement des différentes fenêtres de dialogue et toute la logique associée à ce dialogue (contrôles de

validité des valeurs entre autres, détection des erreurs, reprise sur erreurs, édition de message d'erreurs, etc. ; cf. le cours *modèle dynamique*) ; une partie de ces programmes (25%) ont été écrits à l'aide de générateur de dialogues comme il en existe de nombreux dans les boites à outil IHM. Le *composant application* effectue un certain nombre de conversions, consulte la partie de la configuration stockée dans le serveur graphique, communique avec le reste du système via des interfaces avec le BLI décrit ci-dessus.

Correction – commentaires

GCA est de type E ; C'est un module temps réel relativement critique car il surveille un ensemble d'équipements sensibles.

BLI est de type P ; c'est un bus logiciel qui intègre des logiciels réseaux, mais ce n'est pas lui même du protocole couche basse (E n'est donc pas justifié). Les fonctions de traces et d'historisation peuvent être délicates à réaliser ; de plus le bus doit pouvoir fonctionner indifféremment en environnement centralisé ou distribué. P est donc pleinement justifié.

GCE est une application très orientée gestion de données : on peut prendre globalement S. L'IHM du GCE est assez fortement sollicitée, ce qui pourrait justifier une rectication en utilisant une moyenne pondérée. Sur les 50 KLS de l'IHM, tout n'est évidemment pas de type E.

Concernant le PCG, seul le module Présentation GUI peut être de type E, même si un générateur d'IHM est utilisé (cela simplifie la programmation, mais ni la conception, ni les tests) ; le composant dialogue peut être pris de type P (ou un mixte P et S si l'on considère que seule la partie automate, et non les actions, est un algorithme complexe) ; la partie composant application sera prise de type S.

Question N° 2 (4 points)

Compte tenu de la taille, le projet global a été découpé en 5 sous-projets : GCA, BLI, PCG, GCE plus un projet de maîtrise d'œuvre générale, dédié à la conception générale (étude de définition) et à l'intégration.

1ère partie : En vous servant, soit du vade-mecum du chef de projet, soit du modèle COCOMO, reconstituez le profil de développement de chacun des éléments du système (GCA, BCI et GCE). Donnez une estimation de l'effort d'intégration de l'ensemble.

2ème partie : En vous servant soit du vade-mecum du chef de projet, soit du modèle COCOMO, faites une estimation vraisemblable du nombre de KLS du module PCG en prenant comme hypothèse que 20% des KLS sont de type E, le reste étant un mélange de types P et S, mais à forte dominante S. Vous donnerez une estimation des efforts pour chacune des phases du projet PCG. L'engagement qualité du sous-traitant est que le module PCG puisse entrer directement en exploitation après quelques vérifications effectuées par le maître d'œuvre.

Sur cette base, a combien peut-on chiffrer l'effort d'intégration du PCG qui incombe à l'équipe d'intégration de Power SA qui assure la maîtrise d'œuvre industrielle (MOI) de l'ensemble.

Correction – commentaires

Effort GCA = $3{,}6(120)^{1.2} = 1125hm = 94ha$ en coût brut ; il s'agit d'un portage à l'identique donc on peut rectifier le coût comme suit : 40% de conception =0 car la conception est inchangée ; 20% de programmation et TU, soit 20%×0.5=10% car il faut refaire les tests unitaires et qquelques retouches à la main après le portage ; 40%×0.5=20% car le code porté est

a priori bon, mais il faut quand même le valider après le portage ; soit au total 30% de coût réel, donc :

Effort GCA réel =337hm=28ha.

Effort BLI = $3(50)^{1.12} = 240hm = 20ha$

Effort GCE = $2{,}4(300)^{1.05} = 958hm = 80ha$

Le coût du PCG est de 30ha. En prenant les barèmes du vade-mecum, avec une productivité à 4KLS/ha on aurait 120 KLS, avec 2KLS/ha on aurait 60KLS. Compte tenu de l'existence probable d'un générateur d'IHM, on peut estimer que la génération automatique compense la partie E du module présentation du PCG. Une estimation à 120 KLS de type S est donc tout à fait réaliste.

Le volume total du SC2DB est donc 120(E)+50(P)+120(S)+300(S)= 590KLS

Hors PCG, le coût du SC2DB est de :

$$\frac{120}{470}\times 3{,}6(470)^{1.2} + \frac{50}{470}\times 3(470)^{1.12} + \frac{300}{470}\times 2{,}4(470)^{1.05} \cong 2772hm$$

Avec PCG, le coût du SC2DB est de :

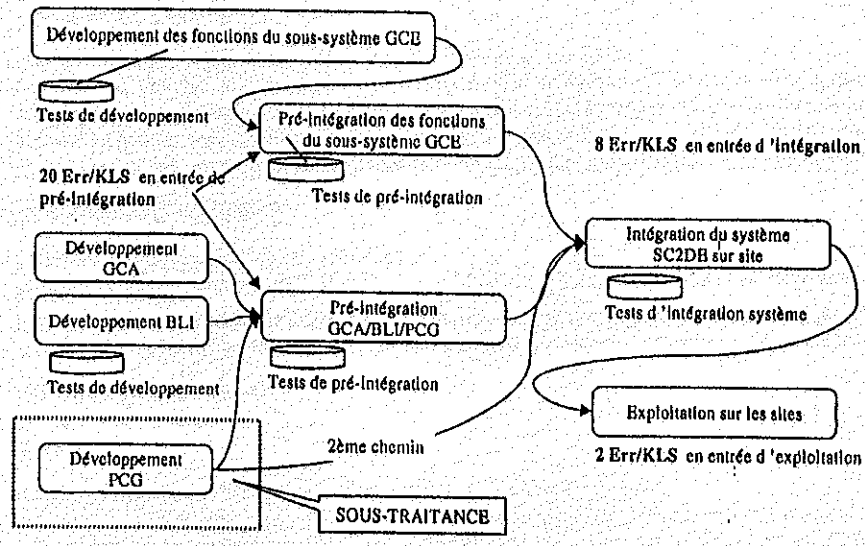
$$\frac{120}{590}\times 3{,}6(590)^{1.2} + \frac{50}{590}\times 3(590)^{1.12} + \frac{420}{590}\times 2{,}4(590)^{1.05} \cong 3257hm$$

Le coût brut du PCG dans le contexte global est donc de 3257-2772=485hm, soit 40ha.

La différence entre le coût négocié (30ha) et le coût calculé selon le modèle, correspond à la fois au coût de conception (nécessaire à la spécification des interfaces) et au coût d'intégration proprement dit. Si l'on applique une peréquation du type vade-mecum, on peut compter que le coût d'intégration sera de l'ordre de 5ha. Si l'engagement qualité du fournisseur se traduit effectivement dans les faits, on peut ramener ce coût à 3ha.

Question N° 3 (2 points)

Etude de l'intégration. Le processus d'intégration global est conforme au schéma ci-dessous :



NB : le nombre d'erreurs en entrée de l'exploitation est un objectif qualité ; pour l'atteindre, il faut que l'intégration et les pré-intégrations purgent le maximum de défauts et s'assurent que les tests unitaires ont été correctement effectués. (Cf. le phénomène d'amplification des erreurs dans le cours Assurance Qualité + statistiques B.Beizer sur la distribution des erreurs).

Quels sont les facteurs de risques d'une telle situation, et plus particulièrement les choix d'intégration pour le module PCG qui a été sous-traité ?

Faut-il gérer les tests à l'aide d'un outil de gestion de configuration?

En vous servant des résultats de la question 2, quels sont les efforts requis pour les pré-intégration, et pour l'intégration ? Proposer une ventilation qui vous paraît raisonnable.

Correction – commentaires

Intégrer directement le PCG sans préintégration revient à faire une confiance absolue au fournisseur du PCG. C'est évidemment une situation à haut risque, sauf si Power Inc. Connaît très bien son fornisseur.

Il est donc prudent d'effectuer une préintégration d'abord.

Il faut gérer les tests en configuration, avec le même sérieux que si c'était du code.

Une répartition 50-50 paraît raisonnable.

Question N° 4 (2 points)

Durant le processus de pré-intégration GCA/BLI/PCG, le maître d'œuvre constate un taux anormal de défauts concernant le PCG. Plus de 50% des défauts constatés auraient du être détectés dans la phase de test unitaires, à la charge du sous-traitants. Compte tenu du rythme de détection des anomalies, le MOI s'attend à émettre environ 1500 rapports d'anomalies à destination du sous-traitant PCG.

En vous servant de la table HP fournie en annexe, quel est le coût des corrections à effectuer par le sous-traitant ? Que peut-on en déduire sur la qualité de son devis initial ?

Décompte pour 1500 défauts

On utilise la table HP, soit :

Temps /durée /effort moyen de correction des 1500 défauts
25% des défauts sont diagnostiqués et corrigés en 2h : $1500\times0.25\times\frac{2}{8}=94$ hj
50% des défauts sont diagnostiqués et corrigés en 5h : $1500\times0.50\times\frac{5}{8}=469$ hj
20% des défauts sont diagnostiqués et corrigés en 10h : $1500\times0.20\times\frac{10}{8}=375$ hj
4% des défauts sont diagnostiqués et corrigés en 20h : $1500\times0.04\times\frac{20}{8}=150$ hj
1% des défauts sont diagnostiqués et corrigés en 50h : $1500\times0.01\times\frac{50}{8}=94$ hj
Total des corrections : 1182 hj = 54 hm = 4,5 ha ; soit un coût moyen de 0,8 hj par défaut

Les 4,5ha sont nettement supérieurs aux prévisions ; la qualité du devis initial n'était pas très bonne. Parmi les explications possibles, il suffirait que le fournisseur ait remplacé du personnel expérimenté par du personnel débutant pour obtenir ce type de dérapage.

NB : on remarquera que pour purger 12 erreurs par KLS, il faut détecter environ 1500 erreurs dans le PCG, compte tenu de l'estimation à 120 KLS ($12\times120=1440$).

1500 erreurs ont été trouvées mais ce ne sont pas celles qui auraient du être trouvées en intégration (erreurs sur les interfaces entre les modules). On notera que cela ne préjuge pas du nombre d'erreurs, en valeur absolue, en entrée en pré-intégration /intégration.

Question N° 5 (2 points)

Sur la base du constat précédent effectué avec des tests dont la finalité est l'intégration, et non pas les tests unitaires, quel est le risque que prend le MOI à mettre en exploitation le système SC2DE en supposant que les 1500 défauts aient tous été corrigés ?

Quel est le nombre de défauts probables qui restent encore à découvrir ? pour cela, faites une hypothèse raisonnable, par exemple 2 fois plus (soit 3000 défauts), 5 fois plus ou 10 fois plus ?

Sur la base de cette analyse quel est le nombre probable des lignes de code source du PCG ?

NB :

a) le coût réel que le sous-traitant aurait du facturer est : 30HA + le coût des défauts découverts en intégration par le MOI.

b) Si le sous-traitant a fait très peu de tests unitaires, cela veut dire qu'il a livré du code compilé sans erreur, et donc que seule les erreurs syntaxiques et quelques erreurs de types (c'est du C++) ont été découvertes ; à ce stade de la programmation, le nombre d'erreurs résiduelles est plutôt de 2 à 3 erreurs par page de programmation (50LS).

Vous pouvez également utiliser les tables 7-1, 7-2 et 7-3 de l'annexe COCOMO de votre cours.

Correction – commentaires

Le maître d'œuvre (MOE) prend un très grand risque à mettre le système en exploitation car l'existence d'un grand nombre d'erreurs découvertes en intégration prouve que le fournisseur du PCG a livré prématurément avant d'avoir atteint un niveau de qualité suffisant pour entrer en intégration.

On peut faire diverse hypothèse pour expliquer cela. La plus fréquente est qu'il a probablement sous-estimé le volume de programmation qu'il avait à gérer. Pour tenir la date de livraison annoncée, le fournisseur a probablement fait l'impasse sur une partie des tests, ce qui peut laisser penser que le volume de code est probablement de l'ordre de 200 KLS.

Si 50% des défauts trouvés en intégration auraient du l'être en développement, cela veut dire que 750 défauts d'intégration restent encore à découvrir.

Si comme tout le laisse à penser, il y a eu inflation de lignes de code (80 KLS, avec une hypothèse à 200 KLS), cela veut dire que les 80 KLS supplémentaire porte avec elles un potentiel de défaut que l'on peut estimer à $80\times18=1440$ défauts.

NB : en raisonnant par différence on a : $(20-8)+(8-2)=18$

Au total, selon notre hypothèse à 200 KLS, on a donc 750+1440=2190 d'erreurs qui restent encore à découvrir ; au lieu de démarrer les installations avec un taux résiduel de 2 Err/KLS, le taux est, a minima, de $\frac{2190}{200}\cong11$ Err/KLS.

Remarquons qu'un programme de 120 KLS compilé sans erreur détectées à la compilation et à l'édition de liens, le nombre de défauts à découvrir est de l'ordre de 50 Err/KLS, soit 6.000 erreurs ; ce nombre passe à 10.000 pour un programme de 200 KLS. Une impasse sur les tests se traduit immédiatement par une augmentation du taux d'erreurs résiduelles.

Bonus

Question N° 6 (1 points)

A propos du tableau IBM donné en annexe, vous remarquerez que la productivité augmente avec la taille, surtout pour les petits programmes, et qu'elle semble se stabiliser au delà de 50 KLS !

Quelle explication pouvez-vous donner à cet apparent paradoxe (ce n'est pas conforme à l'équation d'effort du modèle COCOMO I) ?

Correction – commentaires

C'est la traduction, en chiffre des phénomène d'apprentissage et de réutilisation de modèles de programmation au sein d'un même programme. Dans tout travail de programmation il y a un effort initial quasi incompressible de préparation qui va pénaliser les petits programmes. Pour les très gros programmes, d'autres phénomènes apparaissent, ce qui fait que statistiquement l'allure des équations COCOMO est bonne, mais il faut toujours être prudent.

Question N° 7 (3 points)

A combien peut-on évaluer le préjudice supporté par le MOI du fait de la mauvaise qualité de la fourniture du sous-traitant ? Vous pourrez prendre comme coût moyen de l' H.A 750 KF.

Etude du volume de documentation correspondant à l'expression de besoin du système SC2DE.

En faisant l'hypothèse que 50 LS de code source correspond, en moyenne, à 1 ligne de texte dans l'expression de besoin qu'elle est le volume de l'expression de besoin (vous prendrez 1 page = 50 LS) ?

Quel est le coût de la ligne de l'expression de besoin (cf. règle d'imputation de cet effort dans les tableaux COCOMO) ?

Proposer un nombre d'erreurs probables, ou vraisemblables, de l'expression de besoin.

A votre avis, qu'aurait du faire le MOI pour détecter de façon précoce la mauvaise qualité de la fourniture du sous-traitant réalisant le PCG ? justifiez votre analyse en vous servant des résultats des questions ci-dessus.

Correction – commentaires

Si le volume est de 200 KLS, cela veut dire que l'impasse sur les test est de 50ha-30ha=20ha, en prenant comme productivité le modèle vade-mecum (4 KLS/ha). On peut faire l'hypothèse que la productivité des 80 KLS supplémentaire a été excellent, par exemple en réutilisant bien ce qui a été fait. Si l'on prend comme productivité moyenne 6 KLS/ha (ce qui est certainement très optimiste !) l'écart se réduit à 33,5-30=3,5ha, soit 2.625 KF. On voit que le transfert de coût est de toute façon très important, d'où l'importance, pour le MOI de bien comprendre le fonctionnement et l'efficacité du système qualité de son fournisseur.

Annexe documentaire

Productivité des programmeurs par type de programmes

Source IBM (productivité relative, mesurée sur le système d'exploitation MVS) :

	Volume de code brut /volume de code modifié		
Type	Moins de 15 KLS	15 à 50 KLS	> 50 KLS
P (Langages de commande / Compilateur)	1	2	2.5
E (Contrôle)	0.9	0.45	0.6 – 0.35
E (Protocoles de communications)	0.5	Fourchette : 0.4 – 0.3	Fourchette : 0.5 – 0.3

Coût de correction des défauts

Vous pourrez vous aider des éléments statistiques suivant :

Source HP (moyenne observée sur un grand nombre de projet) :

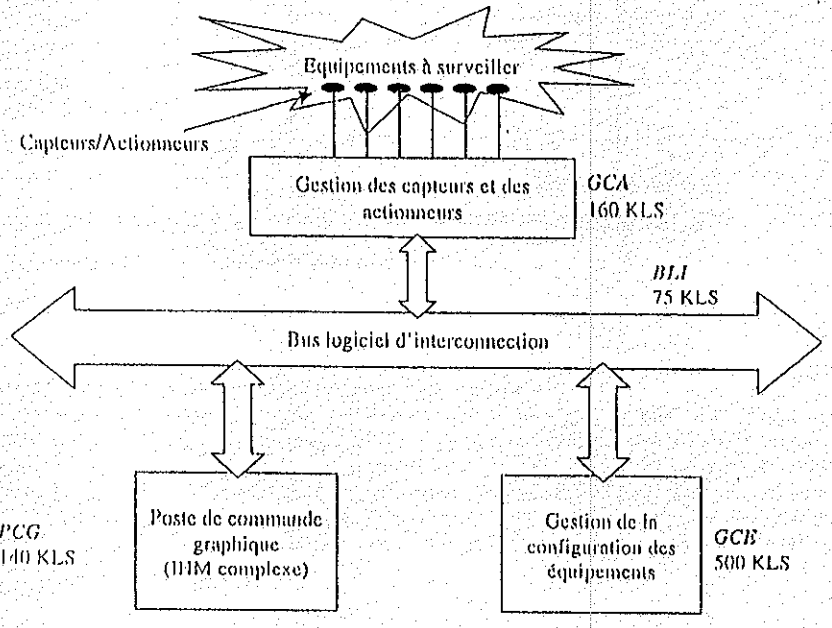
Temps /durée /effort moyen de correction des défauts
25% des défauts sont diagnostiqués et corrigés en 2h
50% des défauts sont diagnostiqués et corrigés en 5h
20% des défauts sont diagnostiqués et corrigés en 10h
4% des défauts sont diagnostiqués et corrigés en 20h
1% des défauts sont diagnostiqués et corrigés en 50h

NB : on prendra 1 HM = 22j ; 1j = 8H.

Cours B5 - Contrôle de Génie Logiciel 2ème session - CORRIGE

Etude de cas avec les corrections

La société Power S.A réalise un système informatique destiné à la surveillance et au contrôle-commande d'un ensemble d'équipements permettant la distribution de l'énergie (SC2DE).
Le système est architecturé comme suit :

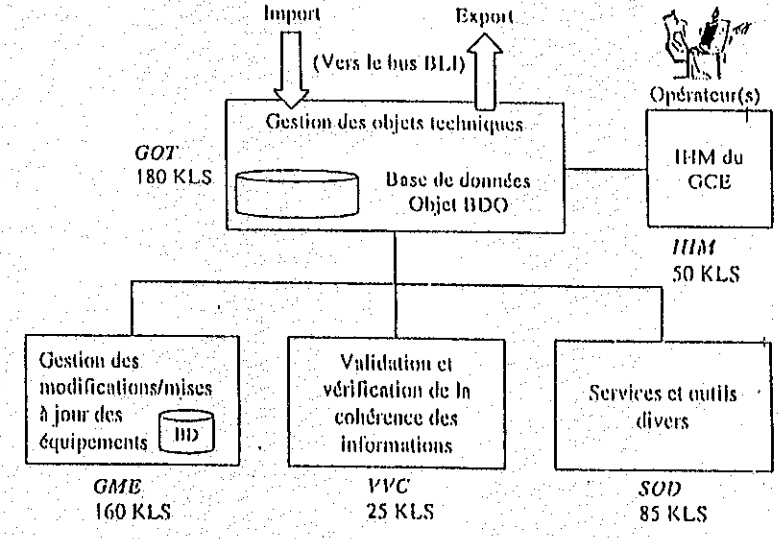


Module GCA
Le GCA est un ensemble temps réel qui doit garantir des temps de réponses de l'ordre de la seconde. Il a été réalisé à partir d'un portage à l'identique d'un système plus ancien afin de pouvoir utiliser une plate-forme plus moderne (unix, C++, etc.). Le portage garantit la reproduction à l'identique du système ancien (y compris des défauts qu'il peut encore contenir).

Module BLI
Le BLI est un "bus" logiciel permettant d'interconnecter, au moyen de librairies, différentes applications et/ou sous-systèmes permettant la gestion des équipements à surveiller. Le bus peut fonctionner sur une seule machine ou dans un environnement distribué (architecture en client-serveur). Il assure des fonctions de stockage temporaire. Le bus utilise le produit ORBIX dérivé des normes CORBA de l'OMG; le produit est encore assez instable mais ses fonctionnalités sont attrayantes.

Module GCE

La GCE est un sous système relativement complexe qui permet de configurer globalement le système SC2DE en fonction des équipements gérés. Sa structure est la suivante :



La qualité des données gérées par cet ensemble est primordiale car toute erreur peut entrainer des destructions d'équipements. Une session de travail effectuée par l'opérateur nécessite de nombreuses interactions entre les différentes fonctions de ce sous-système. Une session typique a un profil comme suit :

- 100 fois IHM
- 300 fois GOT
- 50 fois GME
- 100 fois VVC
- 50 fois SOD

Une panne du GCE n'est pas considérée comme critique car elle n'empêche pas les équipements d'être correctement surveillés, mais peut-être pas de façon optimale.
Module PCG
Le PCG est un sous système graphique qui permet de visualiser en temps réel l'état des équipements. Pour des raisons de sûreté de fonctionnement l'informatique qui assure le fonctionnement du PCG est doublée. Vis à vis du système SC2DE le PCG se présente comme une boîte noire qui a fait l'objet d'une sous-traitance mais dont Power SA assure l'intégration. Le sous-traitant de Power SA a indiqué que le volume source du PCG était de 140 KLS majoritairement écrit en C++.
Le système SC2DE est destiné à être installé et supporté sur environ 150 sites d'exploitation, dont chacun aura une configuration qui lui est propre compte tenu des équipements gérés.