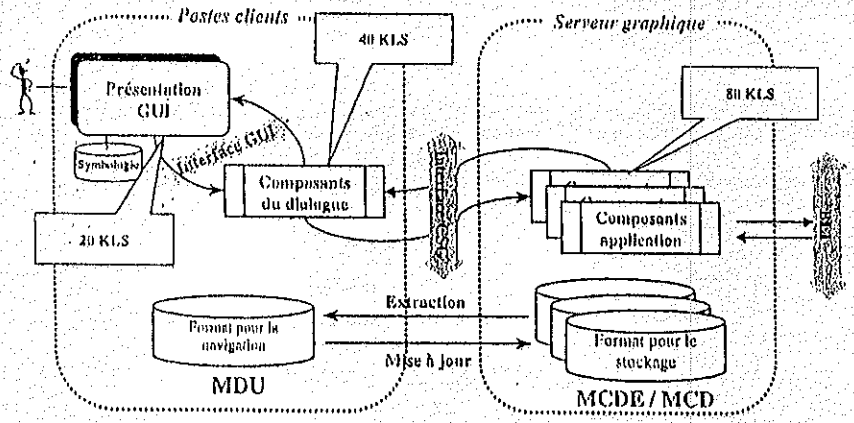
Question N° 1 (2 points)
La structure du PCG est la suivante :



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.

Sans entrer dans le détail, quelles sont les caractéristiques S, P, E de chacun des modules du PCG. Justifiez brièvement vos réponses.

- *GUI est du type E ; c'est le composant qui est en contact direct avec l'utilisateur du système. Les tests ne sont pas automatisables.*
- *CD est globalement de type P car c'est un automate qui gère l'interface entre les interactions de l'utilisateur (GUI) et l'interrogation vers la base de données via CA. 25% des programmes de CD, soit 10 KLS, ont été générés via un générateur de dialogues : le programmeur a donc écrit 30 KLS + ce qui est nécessaire à la génération (1 KLS si on prend un facteur d'expansion de 10). Si l'on prend 30 KLS écrite, la marge d'erreur relative est donc :*

$$\varepsilon = \frac{3.0(30)^{1.12}}{3.0(31)^{1.12}} = (0.97)^{1.12} = 0.967 \quad NB : (1-0.03)^{1.12} \approx 1-0.03\times 1.12 = 0.9664$$

Il est donc légitime de prendre 30 KLS car la différence est dans le bruit de fond du modèle.

- *CA est de type S.*

Question N° 2 (4 points)
En vous servant du modèle COCOMO, reconstituez le profil de développement du PCG. Sur cette base, a combien peut-on chiffrer l'effort d'intégration du PCG par l'équipe d'intégration de Power SA.

Pour les calculs vous pourrez faire l'hypothèse que les 735 KLS du système, hors PCG, se répartissent en 90% de code S et 10% de code P.

Pour le PCG, il faut calculer une moyenne pondérée, sur 130 KLS (et non 140, puisque 10 KLS sont produites sans effort) soit :
GUI : 15%, CD : 23%, CA : 62%.
L'effort moyen pondéré est donc :

$$E(PCG) = \left(3.6(130)^{1.12}\times 0.15 + 3.0(130)^{1.12}\times 0.23 + 2.4(130)^{1.05}\times 0.62\right) = 186+161+245 = 593\ hm$$

Soit : 49 ha de développement total.
En utilisant la table 6-8, l'effort d'intégration et test peut être estimé à 30%, soit : 178 hm ou 15 ha. Une partie de cet effort est nécessaire à l'intégration d'ensemble.
Pour isoler le coût d'intégration du PCG dans l'ensemble complet, y compris le PCG, on procède par double pesée, comme suit :

$$\Delta Intégration = Effort(735+130) - (Effort(735) + Effort(130))$$

Pour le système complet, il faut refaire les moyennes pondérées sur la base de 865 KLS, soit :
Type E → 20 KLS (2%) ; Type P → 73.5+30=103.5 (12%) ; Type S → 661.5+80=741.5 (86%)
Si on assimile la partie E à du code de type P, on ne commet pas une erreur importante, on peut donc prendre comme pondération finale du système complet : 14% de type P et 86% de type S.
Calcul de Effort(735+130) :

$$E1 = \left(3(865)^{1.12}\times 0.14 + 2.4(865)^{1.05}\times 0.86\right) = 818+2504 = 3322 hm$$

Calcul de Effort(735)

$$E1 = \left(3(735)^{1.12}\times 0.10 + 2.4(735)^{1.05}\times 0.9\right) = 487+2208 = 2695 hm$$

La double pesée nous donne : 3322-2695-593=34 hm, soit 2,8 ha de coût brut.
Si l'équipe en charge de l'intégration connaît parfaitement la structure des interfaces, on est en droit d'appliquer une réduction de coût qui est fonction de l'expertise de l'équipe (Cf. le tableau 8-2, attribut ACAP et AEXP).

Question N° 3 (2 points)
Pour valider , vérifier et tester un ensemble de modules comme le PCG, plusieurs stratégies sont possibles :
Stratégie N°1 : Intégrer les trois modules du PCG, indépendamment du SC2DE ; puis intégrer le PCG au système en se servant de l'interface BLI.
Stratégie N°2 : Intégrer le poste client; puis intégrer le serveur graphique au BLI ; puis intégrer l'ensemble. Selon vous, quels sont les avantages et les inconvénients de chacune de ces stratégies ? Donner des arguments en terme de coût, qualité, délai de réalisation, ainsi que de gestion de projet de chacune de ces stratégies (ne pas oublier que le PCG est sous-traité!).

b) Interface CSG ?

Expliquez vos choix? Donner les avantages et les inconvénients des différentes possibilités?

Dans le cas de la stratégie N°2 , ce peut être l'interface CSG car comme il a été dit cette interface doit obligatoirement être simulée. Le langage de la simulation permet de constituer des scénarios préenregistrés qui permettront, le moment venu d'effectuer les non régressions.

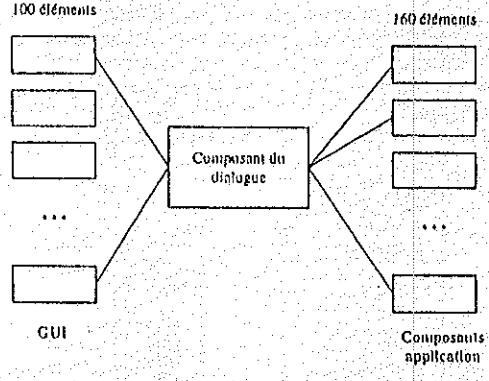
Toutefois, dans la mesure où seul le composant GUI est en contact avec l'utilisateur, et que par ailleurs le CD peut évoluer, il y a intérêt, dans tous les cas de figure à privilégier l'interface GUI. D'un point de vue architecture, l'interface GUI constitue le langage de commande du système ; l'architecte du système (c'est une responsabilité sous l'autorité de la MOE) a tout intérêt à matérialiser ce langage car cela facilitera considérablement le travail d'intégration et de recette car les scénarios ainsi constitués pourront être automatisés très facilement.

Question N° 6 (2 points)

En vue d'améliorer la stratégie de test, Power SA a communiqué à son sous traitant les informations suivantes :

a) le module *Composants du dialogue* est sollicité à chaque interaction, soit par l'un des éléments du module *Présentation GUI* (100 éléments logiciel) ou du module *Composants application* (160 éléments logiciel)
   NB : Ces éléments sont en parallèles du point de vue de la fiabilité
b) le module *Composants du dialogue* est un automate à états finis qui doit fonctionner avec le minimum d'erreurs (si possible 0 erreur !) car il est capable de rattraper des erreurs commises par l'opérateur et de relancer les *Composants application* en cas de défaillance du BLI.

NB : Le schéma de l'interaction est le suivant :



Pour chacun des ensembles *GUI* et *Composants application* on peut considérer que 20% des éléments totalisent 80% des interactions.

Au vu de ces fréquences d'emplois et des volumes de code source les plus fréquemment utilisés quel est, selon vous, l'effort de test à faire sur le module composant du dialogue par rapport aux deux autres modules du PCG ?

Pistes pour l'analyse.

On dispose de 3 ensembles de scénarios. Ces scénarios peuvent être construits de la façon suivante :

- *Les 40 scénarios du GUI servent à préparer les 30 scénarios du CD (c'est une stratégie de test où l'on joue à fond la réutilisation des jeux de données qui servent à construire les tests). Résultat : on ne dispose que de 40 scénarios réellement différents pour le CD.*
- *Même raisonnement avec le CA qui peut être alimenté à partir des 40 scénarios GUI.*

On voit donc que selon les stratégies de tests pour les composants (indépendance des scénarios ou réutilisation) on va disposer d'un nombre de scénarios différents = 120 si on réutilise au maximum, 190 si on se débrouille pour que toutes les données soit différentes (NB : en faisant attention aux données, on dispose, à coût marginal, de 60% de tests en plus !).

Si l'on veut que les scénarios de tests respectent les fréquences d'utilisation, il faut s'arranger pour que :

- *Les 4 KLS du GUI les plus fréquemment utilisées totalisent 40×0.8=32 scénarios.*
- *Les 16 KLS du CA totalisent 96 scénarios.*

Dans le meilleur des cas de scénarios précédents, on peut considérer que le composant CD est sollicité au maximum 190 fois. Est-ce suffisant ?

- *Soit f_{GUIf} et f_{GUIr} les fiabilités des éléments fréquents/rares du GUI ; f_{CD} la fiabilité du composant dialogue, et f_{CAf} et f_{CAr} celles des éléments fréquents/rares du composant application.*

Si l'on néglige les éléments rares, on peut estimer une première fiabilité, sur n itérations, comme suit : $F[n]=(f_{GUIf})^{0.2*n}\times(f_{CD})^{n}\times(f_{CAf})^{0.2*n}$ pour des volumes de code respectivement 4, 40 (et non pas 30, car les lignes générées peuvent contenir des erreurs) et 16.

Ce qui peut se traduire par 4 KLS de GUI ou 16 KLS de CA activent globalement 40 KLS de CD. Si l'effort de test est proportionnel au nombre de scénarios, on voit immédiatement que la pondération des scénarios n'est pas bonne (en supposant évidemment que les scénarios sont homogènes du point de vue des instructions exécutées).

Pour chiffrer plus exactement l'effort de test à effectuer sur le CD, il faudrait faire des hypothèses sur la façon dont l'automate qui pilote le CD a été codé (par exemple soit avec une table, soit en « dur »). Le codage en table (ou en table compilée) est toujours le meilleur du point de vue de la testabilité. Dans la mesure ou CD est un aiguillage indépendant du contexte, il faudrait être très exigeant en matière de tests unitaires (100% de couverture noeuds+arcs du graphe de contrôle).

NB : un vrai calcul de fiabilité exigerait quelques précautions supplémentaires ! Rappelons que ici, dans un premier temps, c'est d'abord l'ordre de grandeur qui nous intéresse.

EXAMEN DE GÉNIE LOGICIEL

Cours B5 N°16767 et N° 21955

Session I du 10 février 2001
ARCUEIL

LA PARTIE I (COURS) ET LA PARTIE II (EXERCICE) DOIVENT IMPERATIVEMENT ETRE TRAITEES SUR DEUX COPIES DISTINCTES.

Prenez le temps de bien lire l'énoncé.

Tous documents et calculettes sont autorisés.

Barème :

Partie I : 12 points

Partie II : 8 points

Aucun résultat ne sera communiqué par téléphone. Les notes seront affichées par les services de la scolarité .

100

Cours Génie Logiciel B5 – Année universitaire 2000-2001

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 :

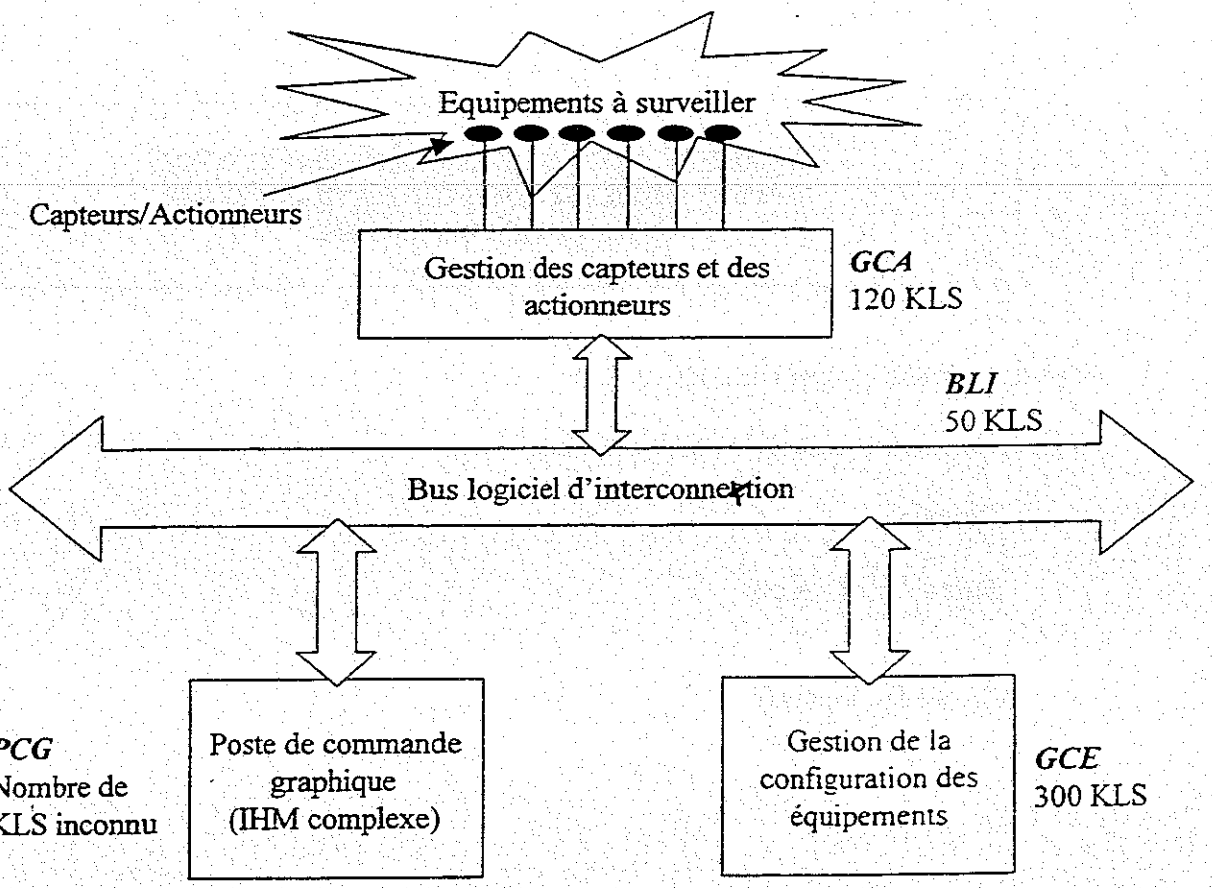


Schéma général du système

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. Certains des équipements à la frontière des régions territoriales 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

101

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 GCE ; au fur et à mesure de l'initialisation et/ou des modifications les descriptifs sont chargés dans le GCA.

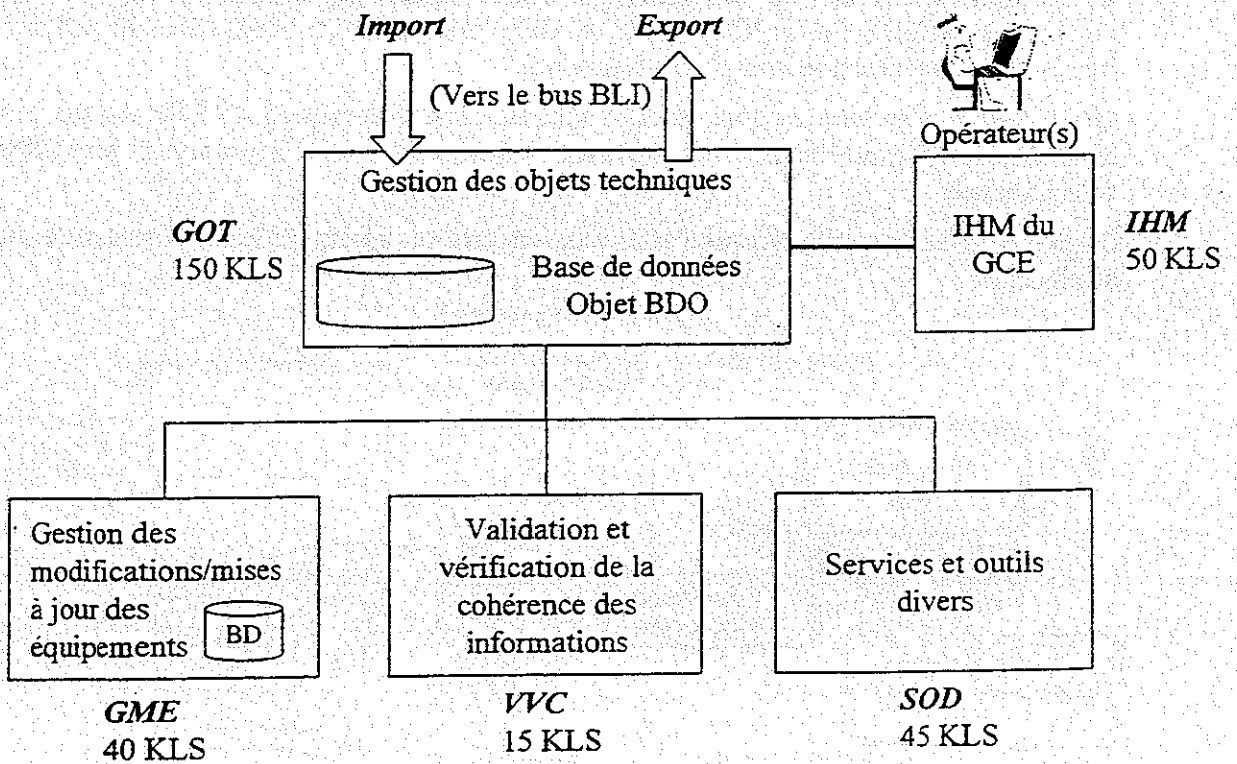
Module BLI

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 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 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 d'exécution 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

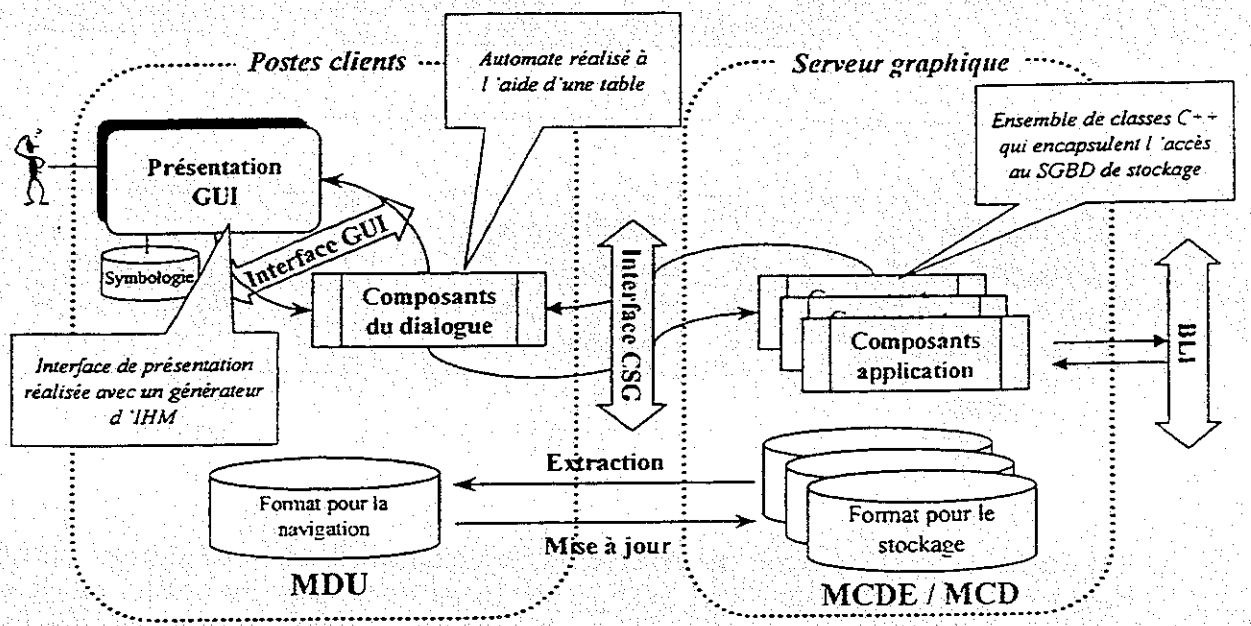
Le PCG est un sous système graphique qui permet de visualiser en temps réel l'état des équipements. Pour des raisons de sûreté de fonctionnement, l'informatique qui assure le fonctionnement du PCG est doublée. Vis à vis du système SC2DE le PCG se présente comme une boîte noire qui a fait l'objet d'une sous-traitance mais dont Power SA assure l'intégration. Le sous-traitant de Power SA 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 (hors PCG). Pour le GCE vous pouvez éventuellement 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-après 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 générateurs de dialogues comme il en existe de nombreux dans les boîtes à outils 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.

Question N° 2 (4 points)

Compte tenu de sa 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.

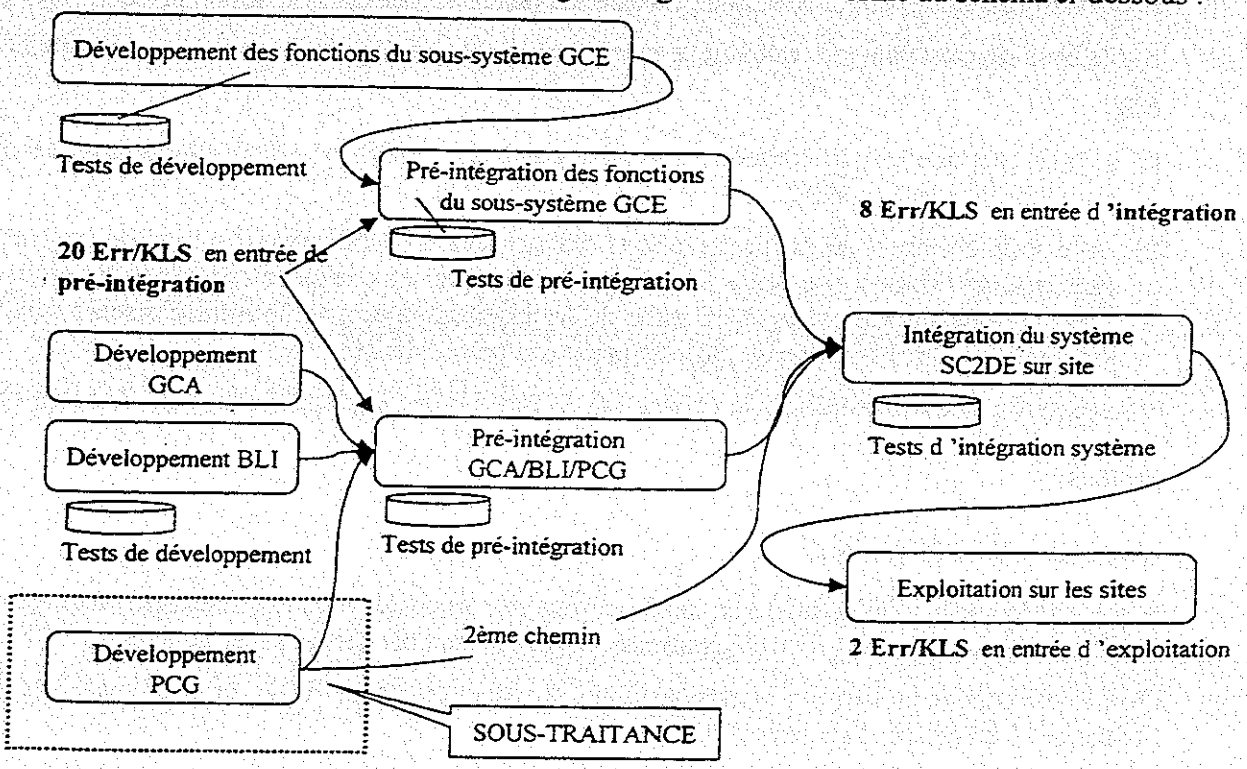
1ère partie : En vous servant, soit du vade-mecum du chef de projet, soit du modèle COCOMO, reconstituez le profil global de développement de chacun des éléments du système (GCA, BLI et GCE). Donnez une estimation de l'effort d'intégration de l'ensemble.

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, à 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.

Question N° 3 (2 points)

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



104

NB : le nombre d'erreurs en entrée de l'exploitation est un objectif de 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 risque 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-intégrations, et pour l'intégration ? Proposer une ventilation qui vous 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 tests 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 ?

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 dû 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.

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 (NB : ce n'est pas conforme à l'équation d'effort du modèle COCOMO !) ?

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 correspondent, en moyenne, à 1 ligne de texte dans l'expression de besoin, quel 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 dû 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.

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 suivants :

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

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 Génie Logiciel B5 – Année universitaire 2000-2001

Corrigé de la Question de cours (12 points) – 2ème 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 :

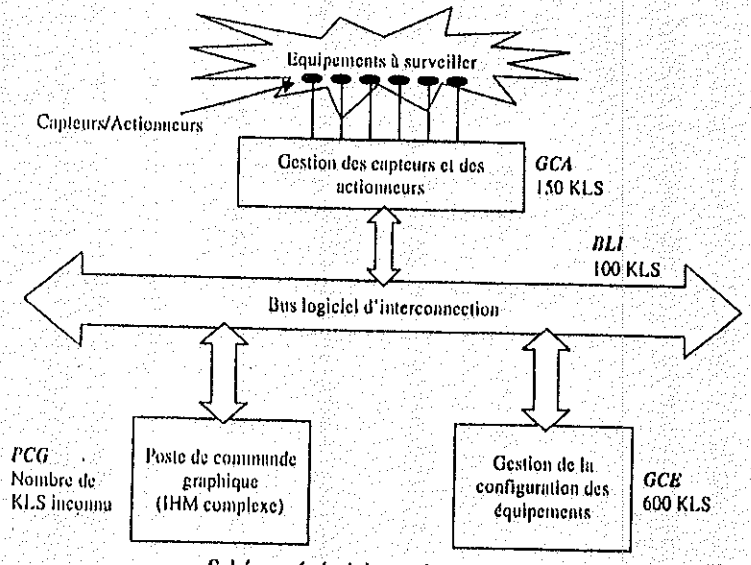


Schéma général du système SC2DE

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. Certains des équipements à la frontière des régions territoriales 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 GCE ; au fur et à mesure de l'initialisation et/ou des modifications les descriptifs sont chargés dans le GCA.

Module BLI

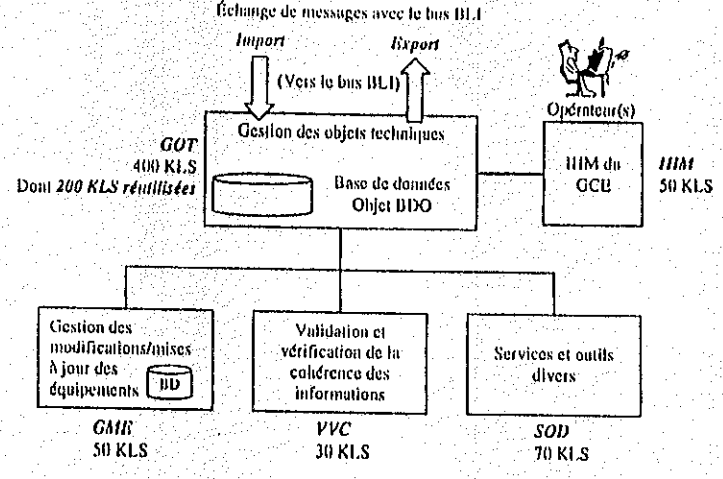
Le BLI est un "bus" logiciel permettant d'interconnecter, au moyen de librairies, différentes applications et/ou sous-systèmes permettant la gestion des équipements à surveiller. Le bus peut fonctionner sur une seule machine ou dans un environnement distribué (architecture en client-serveur). Il assure des fonctions de stockage temporaire des messages qui ***ne doivent être perdus en aucun cas***. 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) ainsi que des fonctions d'interrogations locales et à distance pour la télémaintenance.

Module GCE

La GCE est un sous système relativement complexe qui permet de configurer globalement le système SC2DE en fonction des équipements gérés. La base GOT contient en particulier des objets cartographiques permettant la description des éléments sur le terrain ; les fonctions d'accès à ces éléments ont été récupérées dans une application plus ancienne (soit 200 KLS de code source) où il donnait entière satisfaction. Ces fonctions n'ont pas été modifiées mais il a fallu faire dans GOT un adaptateur d'interface permettant de les utiliser. Il a été nécessaire de tout recompiler et de s'assurer que l'intégration était correcte

La structure du module GCE 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 d'exécution comme suit :

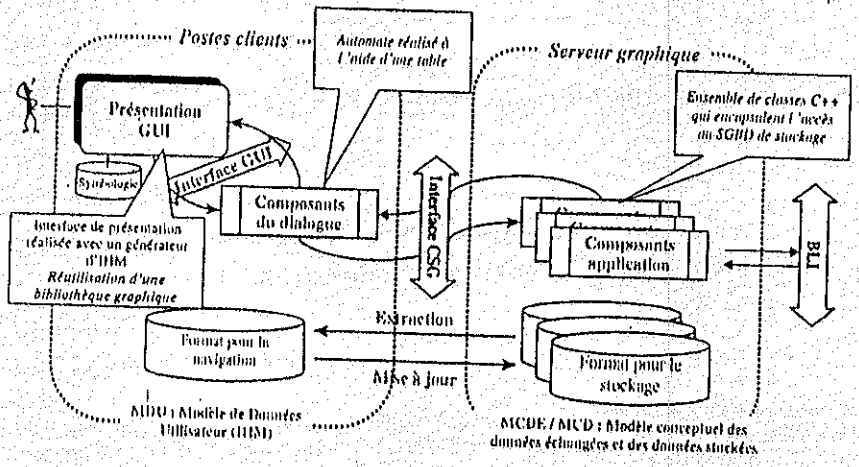
- 100 fois IHM
- 300 fois GOT
- 50 fois GME
- 100 fois VVC
- 50 fois SOD

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 n'a pas communiqué le volume source du PCG (cela n'était pas demandé dans le contrat) ; le maître d'œuvre Power SA sait toutefois qu'il s'agit de code écrit en C++ qui a été facturé 20 HA. Le produit PCG utilise une bibliothèque graphique développée dans un contrat précédent qui avait été facturé 15 HA. On estime que 10% du code correspondant a été modifié et que l'essentiel des tests a pu être récupéré.

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 à l'aide de la bibliothèque graphique réutilisée. 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 générateurs de dialogues comme il en existe de nombreux dans les boîtes à outils IHM ; le modèle mémoire MDU est une structure de données complexe qui permet d'optimiser la navigation pour la présentation graphique. Le *composant application* effectue un certain nombre de conversions simple, consulte la partie de la configuration stockée en mémoire sous la forme de tables relationnelles ; il communique avec le reste du système via des interfaces avec le BLI décrit ci-dessus. Le MDU et le MCDE/MCD sont des modèles de données en mémoire centrale utilisés par les composants et la présentation.

Avant de commencer à répondre lisez et relisez très soigneusement l'énoncé de l'étude de cas. Ne répondez pas à des questions qui n'ont pas été posées.

Question N° 1 (2 points)

Compte tenu du fait que le module GOT contient du code réutilisé, dans quelle type de code pouvez-vous classer globalement le GOT. Quel serait selon vous le type du code réutilisé si il était à refaire. Justifiez brièvement vos réponses.

Le MDU, en cas de panne, peut être reconstruit à partir des données du MCDE/MCD. Quel est le type du module *Composants du dialogue* du PCG. MCDE est l'image mémoire du MCD stockée sur disque ; quel est le type du *Composant application*. Justifiez brièvement mais précisément vos réponses. Vous pouvez utiliser les statistiques IBM ci-après dans l'annexe si elles vous paraissent pertinentes.

Correction - commentaires

Les fonctions cartographiques de GOT sont des algorithmes complexes développés dans une application antérieure où elles donnent satisfaction. Ces fonctions ont donc un bon niveau de maturité. Elles ne sont vues dans la partie écrite de GOT que par l'adaptateur d'interface. Les fonctions d'import et d'export sur le bus sont analogues à des entrées-sorties munies de convertisseurs de format qui sont des programmes simples. Le volume de programmation de GOT est donc 200KLS de type S.

Compte tenu des algorithmes à mettre en œuvre pour la cartographie et de l'utilisation de la base de données objet qui requiert une navigation complexe, les fonctions cartographiques sont de type P.

Le module Composants du dialogue ne fait que piloter l'affichage ; ce n'est pas un module critique pour l'ensemble de l'application. C'est un automate d'affichage du même type que celui vu dans le cours conception. Il est réalisé à l'aide d'une table. Pour cette raison on pourrait prendre une équation COCOMO intermédiaire entre P et S, par exemple $Eff = 2{,}7(KLS)^{1.05}$; (si l'automate était programmé et complètement mélangé avec le code de vérification et d'édition des erreurs, il vaudrait mieux prendre le type P).

Question N° 2 (3 points)

Compte tenu de sa 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.

En vous servant du modèle COCOMO, reconstituez le profil global de développement de chacun des éléments du système (GCA, BLI et GCE).

Pour le GCE vous tiendrez compte du code réutilisé : a combien chiffrez-vous le coût de cette réutilisation ? Expliquez et justifiez vos hypothèses ?

Donnez une estimation de l'effort d'intégration de l'ensemble.

Correction – commentaires

GCA : 150 KLS portée à l'identique

Effort GCA = $3{,}6(150)^{1.2} = 1471hm = 122ha$ 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 quelques 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 = 441hm = 37ha.

Durée : $0{,}5\times\sqrt[3]{37} = 1{,}6$ année

Si on avait assimilé GCA a du code de type S on aurait obtenu : $2{,}4(150)^{1.05} = 462hm = 38ha$!

BLI module algorithmique critique de 100 KLS de type P, soit :

Effort BLI = $3(100)^{1.12} = 521hm = 43ha$

Durée : $0{,}5\times\sqrt[3]{43} = 1{,}7$ année

GCE module composite de 600 KLS dont 200 de réutilisées dans GOT ; globalement on peut prendre un type S, soit :

Effort GCE = $2{,}4(400)^{1.05} = 1295hm = 108ha$

Compte tenu de la maturité du code réutilisé dans GOT, on est en droit de le comptabiliser pour 0 dans l'effort et dans la durée. Si on a le moindre doute, il faudra le comptabiliser comme suit :

Effort GCE (+code réutilisé) = $2{,}4(600)^{1.05} = 1982hm = 165ha$, soit 57 *ha* de plus

Le coût brut de développement de 200 KLS est de : $2{,}4(200)^{1.05} = 625hm = 52ha$ qui doivent être retranché. Le coût de *conception – intégration* est donc de 57-52 = 5 *ha*, soit pour l'intégration seule : 2,5 *ha* (soit 2,3%).

Durée : $0{,}5\times\sqrt[3]{108} = 2{,}3$ années

Question N° 3 (3 points)

En vous servant du modèle COCOMO, faites une estimation vraisemblable du nombre de KLS du module PCG, en prenant comme hypothèse que la bibliothèque graphique réutilisée est de type E (attention, ce n'est pas le même type de réutilisation que pour le GCE), le reste étant un mélange de types P et S, que vous avez caractérisé dans la question N°1. 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 Power SA.

Sur cette base, à 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

Structure de coût du PCG : code C++, 20 *ha* + 15 *ha* de bibliothèque graphique de type E.

On a donc :

$15\times12 = 3.6(x)^{1.2} \rightarrow \ln 180 = (\ln 3{,}6) + 1{,}2\ln x \rightarrow \ln x = 3{,}26 \rightarrow x = e^{1.16} \approx 26$ KLS

10% de ces KLS sont modifiées, soit 2,6 KLS à écrire.

La programmation du PCG correspond donc à : y KLS + 2,6 KLS (à modifier parmi les 26 KLS).

Le coût du PCG (20×12=240 *hm*) correspond à : Eff(y)+Eff(2,6)+ CI (Coût Intégration).

Si on compte pour simplifier CI=0, on a :

$$240 = 2{,}7(y)^{1.09} + 3.6(2{,}6)^{1.2} \rightarrow \ln(y) = \ln\left(\frac{229/2{,}7}{1{,}09}\right) = \ln(78) \approx 4{,}36 \rightarrow y = 78 \text{ KLS au plus !}$$

Si l'on compte, comme ci-dessus, 2,3% de coût d'intégration, soit 240×2,3=5,52 hm, on obtient une meilleure valeur approchée en reprenant l'équation avec 234, soit :

$$234 = 2{,}7(y)^{1.09} + 3.6(2{,}6)^{1.2} \rightarrow \ln(y) = \ln\left(\frac{223/2{,}7}{1{,}09}\right) = \ln(76) \rightarrow y = 76 \text{ KLS vraisemblablement.}$$

Pour un calcul mathématiquement exact, l'équation serait beaucoup plus complexe car il faudrait faire intervenir les moyennes pondérées sur la somme : y KLS + 2,6 KLS (type E) à modifier + 23,4 KLS réutilisée.

$$CI = 0.5\left(\left(\frac{y}{y+26}\times 2{,}7(y+26)^{1.09} + \frac{26}{y+26}\times 3{,}6(y+26)^{1.2}\right) - \left(2{,}7(y)^{1.09} + 3{,}6(26)^{1.2}\right)\right)$$

que l'on pourrait simplifier en utilisant le développement en série de :

$(1+x)^a = 1 + ax + a(a-1)x^2 + \ldots$

L'approximation précédente est largement suffisante.

Pour la répartition des efforts par phase, on prend la table 6-8 de l'annexe COCOMO avec 128 KLS.

Le volume de code du PCG à prendre en compte pour l'intégration est 78+26=104 KLS, pour un coût de 35 *ha*.

Le coût des modules non intégrés (754 KLS) est de :

37 ha(GCA) + 43 ha(BLI) + 108 ha(GCE) + 35 ha(PCG) = 223 ha

Pour l'intégration, il faut faire la somme pondérée par type de code, soit :

$$\frac{150}{754}2{,}4(754)^{1.05} + \frac{100}{754}3(754)^{1.12} + \frac{400}{754}2{,}4(754)^{1.05} + \frac{104}{754}2{,}7(754)^{1.09}$$

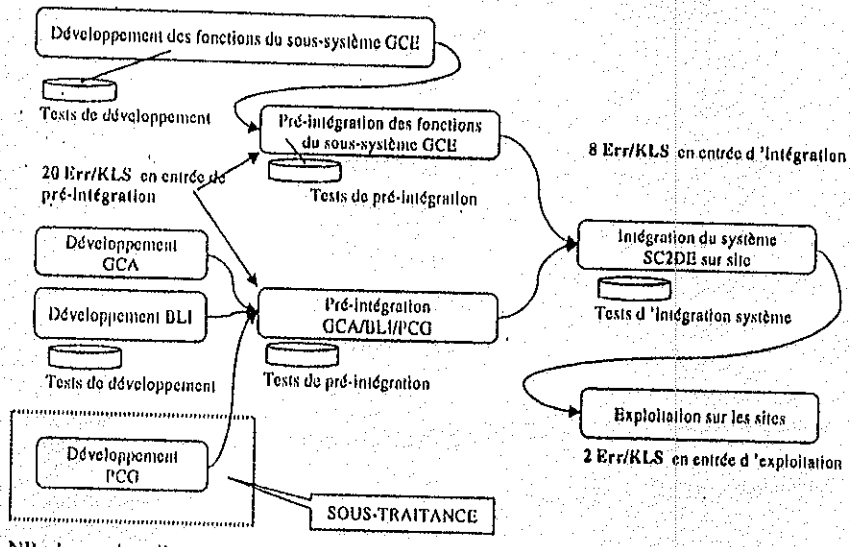
$= 501 + 664 + 1337 + 510 = 3012hm = 251ha$

NB : on aurait pu regroüper directement GCA et GCE.

Soit un coût d'intégration : $0.5(251-223) = 14ha$ (environ 6,5% du coût des modules non intégrés).

Question N° 4 (2 points)

Etude de l'intégration. Le processus d'intégration choisi par Power SA est conforme au schéma ci-dessous :



NB : le nombre d'erreurs en sortie de l'intégration est un objectif de 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 risque d'une telle situation, en particulier suite aux hypothèses de bonne qualité du code réutilisé dans le GCE et le PCG ? Que doit faire Power SA pour s'assurer que ce code répond bien au objectifs qualité du nouveau projet ?

En vous servant des résultats des questions N° 2 et 3, quels sont les efforts requis pour les pré-intégrations, et pour l'intégration ? Proposer une ventilation qui vous paraît raisonnable. Quel est le coût moyen d'une erreur détectée en pré-intégration/intégration ?

Correction – commentaires

La stratégie d'intégration adoptée par Power SA minimise les risques grâce aux 2 pré-intégrations, surtout si celles-ci ont été précédées de revues qualité avec les fournisseurs des différents modules (revues de conception et inspection du code).

Les 108 ha du GCE comptabilisent la pré-intégration, que l'on peut cependant estimer à ≈4% de l'effort total GCE (par rapport aux 2 calculs déjà effectués), soit 4,5 ha.

Les 14 ha d'intégration calculés à la question 3, concernent la somme GCE+GCA+BLI+PCG. Ils incluent donc la pré-intégration GCA+BLI+PCG (soit 354 KLS ; ≈50% du total). Par homogénéité avec le GCE, on peut estimer cette pré-intégration à 5 ha, soit 9 ha pour l'intégration finale du système.

Le coût total pré-intégration + intégration est : 4,5 + 14 = 18,5 ha.

Ces 18 ha ont permis de purger 18 Err/KLS intégrées soit un coût moyen par erreur de :

$$\frac{18\times12\times22\times8}{18\times754} = 2{,}8 heures$$

Question N° 4 (2 points)

Durant le processus de pré-intégration du GCE, le maître d'œuvre constate un taux anormal de défauts concernant le code réutilisé pour les fonctions d'accès aux objets cartographiques. Le nouveau contexte très interactif (alors que le précédent était plutôt batch) fait apparaître des erreurs qui étaient restées latentes (conflits d'accès sur les données, perte d'intégrité, violation de partage, etc.).

Selon vous quel est l'effort moyen de diagnostic et de correction de ces nouvelles erreurs ?

Compte tenu du rythme de détection des anomalies, Power SA s'attend à trouver entre 100 et 200 erreurs de cette catégorie qui vont nécessiter soit de modifier le code réutilisé, soit de modifier les interfaces d'accès développées pour utiliser ce code.

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

Correction – commentaires

Les erreurs nouvelles détectées sont plutôt graves et difficiles à diagnostiquer. Par rapport à la table HP, on est plutôt dans la catégorie 10h d'effort de correction par défaut.

Le surcoût correspondant à 100 défauts de ce type est donc : 125 hj (≈6 hm) ; et ≈1 ha pour 200 défauts.

Le plus grave dans une telle situation est l'impact sur les délais car il va y avoir de nombreux retour arrière vers la programmation que l'on croyait terminée. La bibliothèque étant supposée stabilisée, les programmeurs qui l'ont développé sont certainement employés ailleurs. Le surcoût réel est donc beaucoup plus important que le 1 ha visible des corrections.

Pour l'équipe d'intégration, il peut y avoir des temps mort qui consommeront de l'effort.

La qualité du devis initial n'était pas très bonne ; en particulier les caractéristiques non fonctionnelles ont été mal analysées car il était normalement assez simple de s'apercevoir du changement de contexte d'emploi du batch à l'interactif.

Bonus

Question N° 5 (2 points)

En vous basant toujours sur le devis initial de la bibliothèque graphique, recalculer le volume probable de cette bibliothèque C++, sachant que la bibliothèque a été programmée en utilisant les classes de base fournies avec l'environnement de programmation disponible sur le poste client et à l'aide d'un générateur automatique de code. Ces classes de base gèrent tous les événements fins de l'IHM de sorte que le programmeur n'a à programmer que les interfaces de haut niveau. En fonction de cette hypothèse, peut-on encore affirmer que la partie réellement programmée de l'IHM est toujours de type E ?

Correction – commentaires

Il s'agit des 15 ha rajouté au PCG, évaluée initialement à 26 KLS.

Tous les événements fins sont gérés par la bibliothèque constructeur (par exemple une bibliothèque de type X-windows). Le code d'accès à cette bibliothèque n'est donc pas de type E, mais plutôt de type P, voire même S.

Le volume de code à valider est probablement beaucoup plus gros que 26 KLS, surtout si un générateur d'IHM a été utilisé. Soit au minimum 50 KLS, voire 70 à 80 KLS, bien que le programmeur ait eu l'impression d'en écrire beaucoup moins !

Question N° 6 (1 point)

L'architecte en charge du module GCE à le choix entre deux solutions pour appeler les fonctions d'accès aux objets cartographique :

S1 : Laisser les programmeurs appeler librement les fonctions, en utilisant par exemple des macros définitions disponibles en C ou en C++.

S2 : Faire une interface qui encapsule complètement les fonctions d'accès de façon à ce que les programmeurs soient obligés d'utiliser cette interface pour appeler les fonctions.

Quel est selon vous le meilleur choix. Justifiez votre réponse.

Correction – commentaires

Il vaut mieux faire une interface unique.

Question N° 7 (1 points)

Suite aux nouvelles erreurs découvertes dans les fonctions d'accès réutilisées du module GOT, le chef de projet décide, sur le conseil de son architecte, d'instrumenter le code avec des traces pour faciliter le diagnostic. On compte programmer un point d'appel de la trace en moyenne toutes les 100 LS, et un petit module d'édition de 500 LS. Quel est le volume de programmation pour faire cette instrumentation ? Quel est le type du programme correspondant ? Quel est le coût de ce programme (justifiez la productivité que le programmeur de la trace peut pratiquer) ?

Correction – commentaires

Il faut programmer 2000 appels à la trace, soit une macro de 1 ligne + les 500 ls du module d'édition.

L'instrumentation nécessite 2,5 KLS, de type S (soit 2 à 3 hm de travail pour une productivité de 8 à 10 KLS/hm sur ce type de code).

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 suivants :

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

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.

EXAMEN DE GÉNIE LOGICIEL

Cours B5 N°16767 et N° 21955

Session I du samedi 16 février 2002
ARCUEIL 9h30 –12h30

LA PARTIE I (COURS) ET LA PARTIE II (EXERCICE) DOIVENT IMPERATIVEMENT ETRE TRAITEES SUR DEUX COPIES DISTINCTES.

Prenez le temps de bien lire l'énoncé.

Tous documents et calculettes sont autorisés.

Barème :

Partie I : 12 points (+ bonus)

Partie II : 8 points

Aucun résultat ne sera communiqué par téléphone. Les notes seront affichées par les services de la scolarité .

Cours Génie logiciel B5 – Année universitaire 2001-2002

1ère session - Etude de cas (CORRIGE)

La société Power S.A réalise, maintient, adapte un ensemble de systèmes informatiques destinés à la surveillance et au contrôle-commande de différentes catégories d'équipements (Haute tension / HT, Moyenne tension / MT, Basse tension / BT) permettant la distribution de l'énergie (SC2DE).
Les systèmes qui ont tous une structure semblable répondant à un même besoin, sont, ou devraient être, architecturés comme suit :

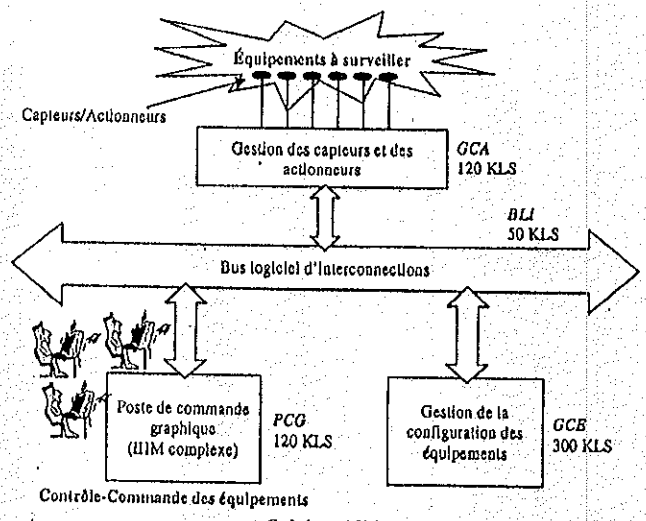


Schéma N°1

Dans cette architecture (Schéma N°1), les concepteurs font jouer au BLI un double rôle :

1. Echanges d'information entre les modules des 3 sous-systèmes exclusivement par messages,
2. Aucune donnée n'est partagée entre les sous-systèmes (chaque sous-système est donc autonome vis à vis de ses données, ce qui améliore le confinement en cas d'erreur et aussi la sécurité).

Le risque de cette architecture est probablement la performance du Bus.

Cependant, certains systèmes, conçus par des équipes différentes, n'ont pas jugé utile d'interfacer les différents composants du système à l'aide d'un bus (Schéma N°2). Les échanges et les enchaînements entre modules sont réalisés par du partage de données et/ou de fichiers, par des enchaînements séquentiels classiques, ou par des tâches asynchrones construites à l'aide des fonctions du système d'exploitation.
Dans cette configuration, l'effort d'architecture est moindre, mais en contrepartie le volume de programmation augmente car ce qui était potentiellement factorisable n'a pas été identifié. On se propose d'analyser et de quantifier ces différences.

M3

L'architecture est alors la suivante :

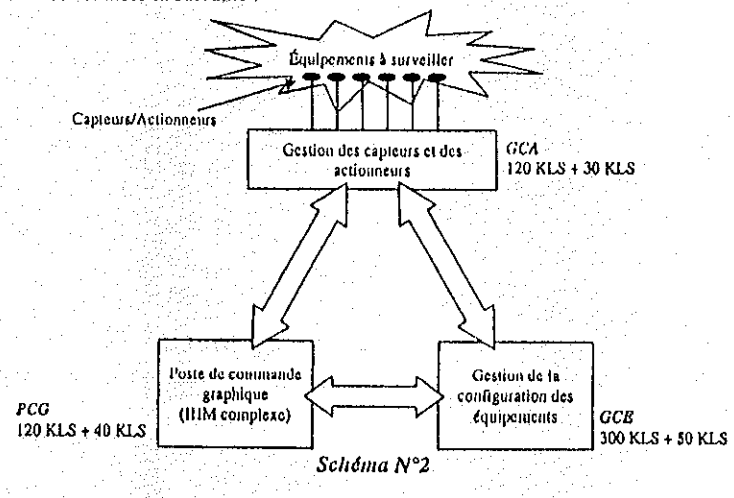


Schéma N°2

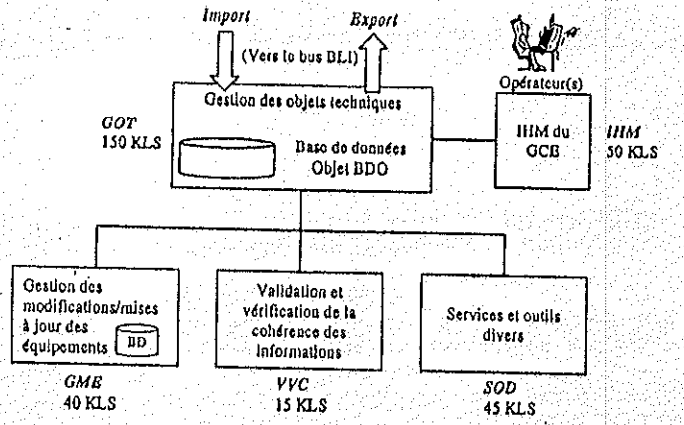
NB : on remarquera sur ce schéma N°2 que le code qui remplace le bus BLI entraîne un surcroît de programmation égal à : 30 KLS(GCA) + 40 KLS(PCG) + 50 KLS(GCE) = 120 KLS (en lieu et place des 50 KLS du bus BLI)

Description des modules

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 le GCE ; 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 ; le produit est encore assez instable mais ses fonctionnalités sont attrayantes.
Le bus dispose de fonctions de stockage qui lui sont propres (traces, historique des messages, etc.).

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 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 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 plus de façon optimale.

Module PCG

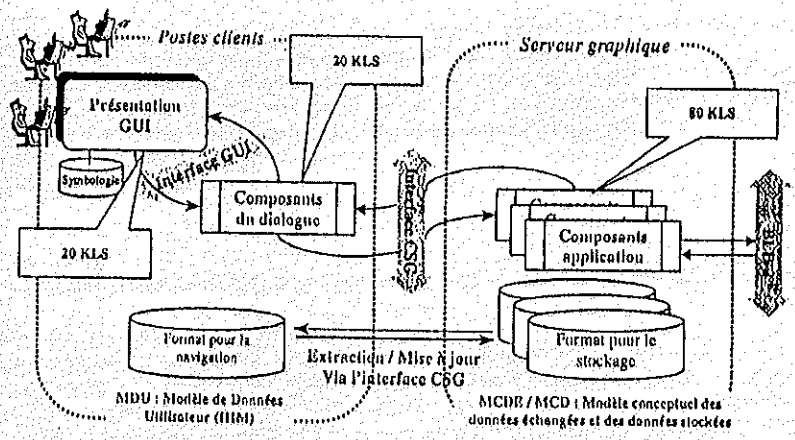
Le PCG est un sous système graphique qui permet de visualiser en temps réel l'état des équipements. Pour des raisons de sûreté de fonctionnement l'informatique qui assure le fonctionnement du PCG est doublée. Vis à vis du système SC2DB le PCG se présente comme une boîte noire qui a fait l'objet d'une 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 120 KLS majoritairement écrit en C++.

Le schéma ci-dessous représente une architecture typique de composants IHM où le concepteur a fait un très grand effort de rationalisation, en séparant soigneusement les fonctions de présentation à l'écran, les fonctions de dialogues qui permettent d'enchaîner les différentes fenêtres, et enfin les fonctions de types applicatives qui manipulent les données affichées à l'écran (accès, stockages intermédiaires, conversions, etc.)

Cette approche conduit à bien distinguer le modèle de données utilisateur (MDU) et les modèles conceptuels de données pour le stockage et les échanges (MCD et MCDB).

Un IHM comporte toujours de nombreux composants ; c'est une bonne pratique architecturale de les regrouper dans ces trois catégories. Si ce n'est pas le cas, cela conduit généralement à de très importantes duplications de code qui augment la taille de l'IHM.

La structure idéale 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%) a été écrite 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. CSG désigne l'interface entre le poste client C (client léger), et le serveur graphique SG.

Le système SC2DB 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 (3 points)

Sans entrer dans le détail, quelles sont les caractéristiques de complexité S, P, E de chacun des éléments du système SC2DB. Pour le GCE vous pouvez utiliser le profil d'exécution. Justifiez brièvement vos réponses. Vous pouvez utiliser les statistiques IBM et HP ci-dessous.

Quels sont, d'un point de vue qualitatif, en terme de productivité du développement, les avantages et les inconvénients de chacune des architectures (Vous distinguerez la productivité en terme d'effort, et la productivité en terme de délai).

Vous pouvez utiliser la grille des caractéristiques FURPSE (cf. cours assurance qualité) et/ou celle de la norme ISO/CEI 9126.

NB : rappel des caractéristiques FURPSE

F : fonctions offertes
U : facilités d'emploi/ergonomie
R : fiabilité (Reliability, en anglais)
P : performance
S : garantie de service / maintenabilité (serviceability, en anglais)
E : évolution (maintenance adaptative et/ou évolutive, par opposition à maintenance corrective qui fait partie de S).

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 rectification 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.

Dans le schéma N°1, on concentre la complexité dans le BLI ; GCA, PCG, GCE sont alors clairement S. De plus l'existence du BLI est un élément simplificateur pour l'intégration finale des 3 sous-systèmes. Durant toute la phase d'intégration le BLI sera très utilisé ; on peut donc considérer que le code correspondant aura une maturité bien meilleure que son équivalent ventilé dans les 3 sous-systèmes. Du point de vue de la maintenabilité, le schéma N°1 sera également meilleur car on peut modifier l'un quelconque des 3 sous-systèmes sans pour autant être obligé de modifier les 2 autres.
Concernant les performances, on peut considérer que la fabrication de l'interface de dialogue est plus coûteuse en temps CPU, mais pas en entrées/sorties, ce qui n'est pas pénalisant si la machine est surdimensionnée en CPU (ce qui est quasiment toujours le cas avec les CPU actuels). Il est même vraisemblable que le schéma 2 sera de toute façon également mauvais sous cet angle car le code correspondant est plus volumineux, ce qui risque de solliciter plus fréquemment la mémoire virtuelle et/ou le « ramasse miettes » ; il suffit de qq. entrées/sorties parasites pour faire basculer la performance du mauvais coté.

Question N°2 (3 points)
Dans l'architecture sans BLI, le code qui correspondrait au BLI est ventilé dans chacun des trois modules GCA, PCG et GCE (voir le NB du schéma N°2). Selon vous, quel est l'impact en terme de coûts de cette ventilation (c'est un facteur de complexité). Comment faut-il considérer ce code source du point de vue S, P, E ? Justifier vos réponses.
En particulier, peut-on considérer que l'ajout de ce code dans les différents sous-systèmes GCA, PCG, GCE, du fait de l'absence du BLI, correspond à une simple augmentation du coût de programmation (environ 50% de l'effort total si l'on prend les tableaux COCOMO, car on pourrait considérer que cela ne change presque rien à la conception générale et à l'intégration) ; ou bien faut-il tout compter ?
Faire un calcul de l'effort de développement du code correspondant dans chacun des deux schémas d'architecture, en justifiant vos hypothèses de programmation avec ou sans augmentation du coût de l'architecture du sous-système.

Dans le cas du schéma N°2 (sans BLI), peut-on envisager de faire une seule équipe de développement pour développer le code source correspondant ? justifier votre réponse.

Correction – commentaires

Le code du BLI est de type P, car c'est du code d'interface jugé complexe. Ce code, une fois ventilé dans GCA, PCG, GCE restera complexe, et de plus il aura tendance à complexifier le sous-système dans lequel il s'ajoute. Pour le GCA qui est un module temps réel, on serait en droit de le considérer entre P et E, par exemple en réajustant l'équation d'effort dans le sens P+.

Coût du BLI dans le schéma N°1 :
$E=3(50)^{1.12} = 240hm = 20ha$
Coût du BLI dans le schéma N°2 :
GCA : 30KLS, $E1=3(30)^{1.12} = 135hm = 11,3ha$
PCG : 40KLS, $E2=3(40)^{1.12} = 187hm = 15,6ha$
GCB : 50KLS, $E3=3(50)^{1.12} = 240hm = 20ha$
L'effort brut total est donc : 46,9ha (ce coût ne tient pas compte de la conception générale et de l'intégration globale).
Si l'on ne compte que la partie programmation, *ce qui est évidement très optimiste*, on est en droit de ne considérer que 58% de ce coût brut (cf. Tableau 6-8 de l'annexe COCOMO), soit 27,2ha (dans le meilleur des cas).

NB : Si l'on assimile le code ventilé dans le PCG et le GCE à du type S (ce qui n'est pas justifié présentement), on aurait :
PCG : 40KLS, $E2=2,4(40)^{1.05} = 115hm = 9,6ha$
GCE : 50KLS, $E3=2,4(50)^{1.05} = 146hm = 12,2ha$
Dans cette hypothèse, le coût brut serait de : 33,1ha, soit en programmation 19,2ha.

Dans tous les cas de figure, même en étant très optimiste, on voit que le schéma N°2 est plus mauvais, bien qu'il puisse paraître plus simple à un chef de projet inexpérimenté.

De plus, dans le cas de l'architecture N°2 il est très difficile, sinon impossible de faire une équipe de développement distincte des équipes GCA, PCG, GCE. Ce schéma d'architecture implique d'augmenter la taille des équipes, et/ou d'augmenter le délai de réalisation de chacun des sous-systèmes. Cela augmente les problèmes de communication entre les équipes. De plus l'interface entre les sous-systèmes n'est pas vraiment perçue comme telle (on interface 2 à 2 les sous-systèmes). On peut imaginer un groupe de coordination transverse qui pourrait prendre en charge la définition des interfaces, pour autant que ceux-ci aient été bien identifiées ; le fonctionnement d'un tel groupe n'est jamais simple (perte de temps, absence de décision claire, etc.). Enfin, les responsabilités et les compétence sont diffuses, alors qu'il est certainement « payant » de mettre en place une petite équipe expérimentée pour réaliser le BLI.

L'écart probable en terme de coût entre les deux schémas est donc de 27,2-20, soit 7,2 ha, dans le meilleur des cas.

Question N° 3-1 (2 points)
1ère partie : architecture avec le BLI
En vous servant, soit du vade-mecum du chef de projet, soit du modèle COCOMO, selon les cas (mais il faudra justifier l'usage du vade-mecum, car il est trop simplificateur dans certain cas) reconstituez le profil de développement de chacun des éléments du système (GCA, GCE, PCG et BLI). Donnez une estimation de l'effort d'intégration de l'ensemble.
Pour simplifier, vous pouvez considérer que GCA, GCE, PCG sont de type S, et BLI de type P.

Correction – commentaires

Pour chacun des sous-systèmes du schéma N°1 (590 KLS), on a :

$E(GCA)=2{,}4(120)^{1.05}=366hm=30{,}5ha$

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

$E(PCG)=2{,}4(120)^{1.05}=366hm=30{,}5ha$

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

Soit au total : 161ha.

Le coût avec conception générale et intégration, s'obtient avec la formule de pondération comme suit :

$$E(\text{Intégré})=\frac{540}{590}\times 2{,}4(590)^{1.05}+\frac{50}{590}\times 3(590)^{1.12}=1783\text{hm}+322\text{hm}=175\text{ha}$$

La différence des deux donne le coût recherché, soit : 14ha

Le modèle vade-mecum n'est applicable qu'au total, soit $\frac{590}{4}=147{,}5ha$ (mais la productivité du BLI est 2KLS/ha, donc un surcoût de 12,5 ha) ; soit au total 160ha (à comparer au 161ha précédent), ce qui montre l'intérêt du vade-mecum pour débroussailler.

L'effort d'intégration proprement dit est 14/2, soit 7ha.

Question N° 3-2 (2 points)

2ème partie : architecture sans BLI

En vous servant soit du vade-mecum du chef de projet, soit du modèle COCOMO, reconstituez le profil de développement du système sans BLI (GCA, GCE, PCG). Donnez une estimation de l'effort d'intégration de l'ensemble.

NB : pour tenir compte de l'impact du code ventilé correspondant au BLI, vous pourrez prendre une équation d'effort rectifiée comme suit : $E=2{,}6(KLS)^{1.08}$ que l'on peut obtenir par simple extrapolation (si vous n'êtes pas d'accord avec cette façon de procéder, justifier votre préférence).

Correction – commentaires

Pour tenir compte de l'impact du code ventilé qui est de complexité P, on rectifie l'équation d'effort comme suit :

Coefficient constant 2,4 à 3, soit un écart de 0,6 : on augmente de 1/3, soit 2,6.

Exposant 1,05 à 1,12, soit un écart de 0,07 : on augmente également de 1/3, soit 1,08.

Dans le schéma N°2, on a :

$E(GCA)=2{,}6(150)^{1.08}=582hm=48{,}5ha$

$E(GCB)=2{,}6(350)^{1.08}=1454hm=121ha$

$E(PCG)=2{,}6(160)^{1.08}=624hm=52ha$

Soit au total : 221,5ha.

A comparer avec le coût intégré : $2{,}6(660)^{1.08}=2884hm=240ha$

Soit une différence de 19ha.

L'effort d'intégration proprement dit est 19/2=9,5ha

Question N° 4 (2 points)

Sur la base de vos résultats (en 2, 3-1 et 3-2), a combien peut-on chiffrer le différentiel entre les deux architectures en termes de coûts et de délai.

Comment faut-il interpréter ces résultats par rapport à ce que vous avez considéré à la question N°2.

Correction – commentaires

En terme d'effort, la différence entre les deux architectures est de 2,5 à 3 ha, ce qui peut paraître faible.

Cependant on a vu que le BLI permet une intégration indépendante des 3 sous-systèmes, ce qui va très certainement autoriser un bien meilleur parallélisme lors de la phase d'intégration. On est donc en droit d'affirmer que dans le schéma N°2, les temps d'attentes seront beaucoup plus difficiles à récupérer, ce qui va occasionner une perte d'effort quasiment impossible à récupérer.

Des calculs plus détaillés, avec des hypothèses sur les flux d'erreurs, et des temps d'attentes sur les équipes des 3 sous-systèmes, montreraient que probablement ce coût incompressible doit être multiplié par 4 ou 5, au minimum, pour une qualité comparable à la livraison.

A comparer à l'écart de 7,2 calculé à la question N°2 déjà considéré comme très optimiste.

Bonus

Question N° 5 : gestion de la configuration (2 points)

En vue de préparer le plan d'intégration, on se propose d'analyser la complexité combinatoire des deux architectures en présence (nombre de composants à intégrer), afin d'estimer l'effort de tests d'intégration.

On se restreint aux composants constitutifs du PCG et du GCE.

On appellera composant le résultat du travail de programmation (hors conception et intégration) de 1 ou 2 programmeurs.

Q 5-1 : dénombrement des composants

En vous servant du Vade-mecum du chef de projet, quelle est la productivité de :

1 programmeur en 3 mois ? 6 mois ?

2 programmeurs en 3 mois ? 6 mois ? (prendre en compte le coût des interactions)

Sur cette base donner une estimation du nombre de composants constitutifs de chacun des modules PCG et GCE. On prendra des distributions en nombre de composants : 25%(3hm), 50%(6hm), 25%(12hm) ou 10%, 80%, 10% ; laquelle vous paraît la plus plausible ?

(NB : 1 programmeur, hors conception - intégration, produit en moyenne par an, selon le vade-mecum, un volume de code 4 KLS×5 = 20 KLS [cf. règle 40-20-40] ; vous pourrez utiliser les informations du tableau IBM dans l'annexe).

Vous pourrez présenter vos résultats à l'aide d'un tableau, comme suit :

	Durée : 3 mois		Durée : 6 mois	
	1 programmeur	2 programmeurs	1 programmeur	2 programmeurs
Taille des composants				
Nombre de composants				
Taille par catégorie				
Taille du module	PCG : 120 ou 160 KLS selon architecture GCE : 300 ou 350 KLS selon architecture			

NB : la somme $\sum_{i}^{4} Taille(composant)\times Nombre = Taille(module)$

Correction – commentaires

On prend une distribution 10% (≤3hm), 80%(6 à 12hm), 10%(≥12hm)

Q5-2 : dénombrement du nombre de scénarios d'intégration

Pour faire ce dénombrement, on peut imaginer 3 modes de calcul

Mode 1 : les modules sont linéairement indépendants, soit n

Mode 2 : on examine les combinaisons 2 à 2, soit $\frac{n(n-1)}{2}$

Mode 3 : on examine toutes les partitions possibles, soit 2^n

Correction – commentaires

Il faut faire qq. hypothèses sur la nature des graphes de contrôle et sur le partage ou non de données entre les modules et/ou les sous-systèmes.

Question N° 6 (2 points)

Pour pouvoir desservir d'autres types d'équipements, le système SC2DE doit subir un certain nombre d'évolutions fonctionnelles qui se traduisent par un accroissement de la taille des différents sous-systèmes et modules.

Dans le cas de l'architecture avec BLI, cela donne un accroissement net de :

- GCA : + 40 KLS
- BLI : + 25 KLS (essentiellement de nouveaux services de traces, de surveillance, de contrôle de la validité des messages, etc.)
- PCG : + 20 KLS
- GCB : + 200 KLS

Rajouter des KLS ne peut pas se faire sans modifier les KLS existantes. Selon vous, comment faut-il comptabiliser ces modifications pour véritablement chiffrer l'effort nécessaire à cette évolution fonctionnelle (faites des hypothèses raisonnables).

Pour l'architecture sans BLI, comment faudrait-il traduire l'augmentation de taille du BLI dans la ventilation sur les différents sous-systèmes.

Selon vous, l'ajout de 200 KLS dans le GCB peut-il se faire à structure du GCB constante, ou faut-il revoir la structuration ? Analyser qualitativement l'impact dans chacune des architectures, en particulier en terme de qualité et contrat de service.

Quelle est le coût de la ligne nouvelle (ou du KLS) dans chacune des architectures ?

Correction – commentaires

Dans l'architecture sans BLI, l'impact sera beaucoup plus important car les interfaces sont diffus, ce qui n'est pas le cas avec le BLI.

Annexes

Vous pourrez vous aider des éléments statistiques suivant :

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

Temps /durée /effort 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

Remarques :

1. ces statistiques ne tiennent pas compte des durées de non régression.
2. cette statistique évolue au cours du processus de développement et d'exploitation. En début de phase de développement on trouve les erreurs « simples », et en fin d'intégration, et a fortiori en exploitation, il reste les erreurs les plus difficiles (parfois non reproductibles). En intégrant au cours du temps, on obtient cette distribution.

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
B (Contrôle)	0.9	0.45	0.6 – 0.35
E (Communications)	0.5	Fourchette : 0.4 – 0.3	Fourchette : 0.5 – 0.3

NB : 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 !

Question complémentaire : quelle explication pouvez-vous donner à cet apparent paradoxe ?