Cours Génie logiciel B5 – Année universitaire 2001-2002

2ème session - Etude de cas (12 points + bonus) ***CORRECTION***

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 électrique. Ces systèmes qui ont tous une structure semblable répondant à un même besoin, sont, ou devraient être, architecturés comme suit :

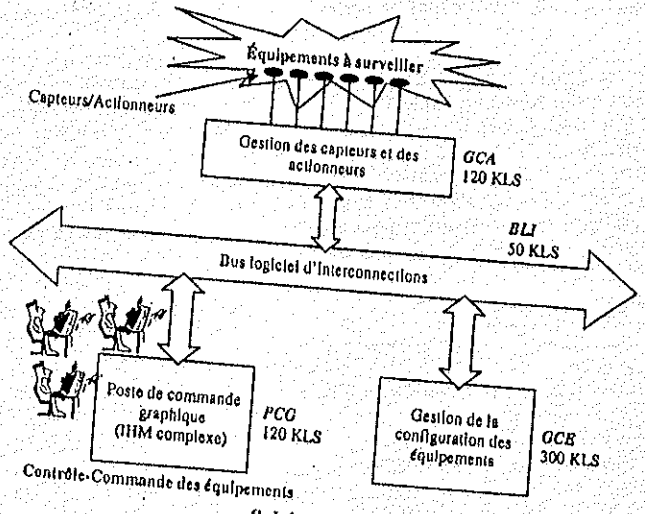


Schéma N°1

Dans cette architecture (Schéma N°1), les concepteurs font jouer au BLI un double rôle :
- Echanges d'information entre les modules des 3 sous-systèmes exclusivement par messages,
- 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 point faible de cette architecture est probablement la performance du Bus.

Description des modules

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 fonctionnalité à l'identique du système ancien (y compris des défauts qu'il peut encore contenir).

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

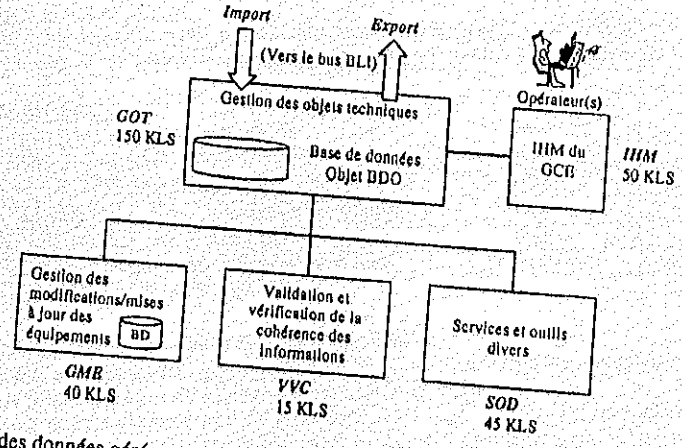
Module BLI

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 ; 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

Le 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 :

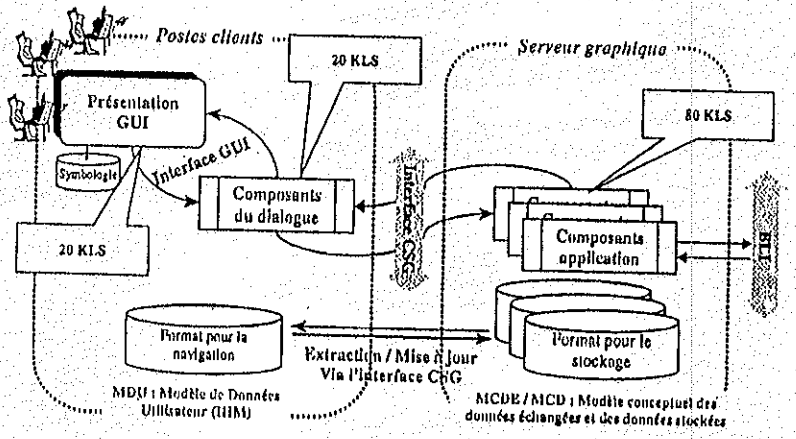
- 100 fois IHM
- 300 fois GOT
- 50 fois GMB
- 100 fois VVC
- 50 fois SOD

Une panne du GCB 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 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 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 MCDE).
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. CSG désigne l'interface entre le poste client C (client léger), et le serveur graphique SG.
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 boîtes 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.

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.

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

Dans le cas de l'architecture avec BLI (cf. schéma N°1), cela donne un accroissement net de volume de code que le chef de projet à estimer comme suit :

- GCA : + 40 KLS
- BLI : + 25 KLS (essentiellement de nouveaux services de traces, de surveillance, de contrôle de la validité des messages, etc. permettant d'instrumenter soigneusement le comportement du système SC2DE en observant ce qui transite sur le bus logiciel BLI)
- PCG : + 20 KLS
- GCE : + 200 KLS

Le directeur du projet a décider de faire du BLI un sous-système assurant un double rôle :

- Organe de communication entre les autres sous-systèmes qui est la fonction primaire du BLI,
- Organe d'intégration entre les différents sous-systèmes (c'est une fonction technique secondaire) de façon à optimiser les délais et la qualité globale du système SC2DE.

Dans cette étude on se propose d'analyser en détail le sous projet BLI.

Question N° 1 (3 points)
Le sous-système BLI, de par la nature des fonctions qu'il assure, est clairement de type P. Le directeur du projet SC2DE a décidé de constituer l'équipe BLI en lui affectant un chef de projet et des collaborateurs très expérimentés ayant déjà réalisés des développements analogues et connaissant bien les progiciels utilisés pour le BLI (ORBIX, etc.).
Cette hypothèse permet au directeur de projet de rectifier l'équation d'effort à partir des données du tableau 8-2 (cf. annexe COCOMO du cours).

NB : Rappel du tableau 8-2

Expérience et/ou maturité des programmeurs	Niveau : élevé	Niveau : très élevé
ACAP (compétence de l'architecte)	0,86	0,71
AEXP (Expérience d'applications analogues)	0,91	0,82
PCAP (compétence des programmeurs)	0,86	0,70
VEXP (connaissance des progiciels)	0,90	idem
LEXP (connaissance des outils de programmation)	0,95	idem

Calculer la nouvelle équation d'effort prenant en compte le coefficient de rectification, selon la méthode indiquée dans le cours (cf. annexe COCOMO), dans chacune des hypothèses : élevé (H1) et très élevé (H2).

Corrections – commentaires

L'équation rectifiée est $E_{rectifié} = 3{,}0 \times k(KISL)^{1{,}12}$, avec k facteur *multiplicatif*, soit :
Hypothèse N°1 (élevé) : $k1 = 0{,}86 \times 0{,}91 \times 0{,}86 \times 0{,}90 \times 0{,}95 = 0{,}58$
Hypothèse N°2 (très élevé) : $k2 = 0{,}71 \times 0{,}82 \times 0{,}70 \times 0{,}90 \times 0{,}95 = 0{,}35$
Soit :
H1 : $E_{rectifié} = 1{,}74(KISL)^{1{,}12}$, soit une amélioration de productivité :
$\frac{E_{nominal}}{E_{rectifié}} = \frac{1}{0.58} = 1{,}72$

H2 ; $E_{reelpt} = 1{,}05(KISL)^{1,12}$, soit une amélioration de productivité : $\frac{E_{modèle\ nominal}}{E_{reelpt}} = \frac{1}{0.35} = 2{,}86$

Le facteur k traduit le fait que ***l'équipe est très expérimentée***, ce qui veut dire qu'elle produit plus vite que l'équipe correspondant au modèle nominal des équations d'effort.

Calculer l'amélioration de productivité dans chacune des hypothèses, par rapport à l'équation nominale. Vous ferez le calcul avec les deux volumes de code source avant évolution (50 KLS) et après évolution (75 KLS).

NB : la productivité d'un projet est égale à $p = \frac{\text{Taille du logiciel}}{\text{Effort de développement}}$.

Corrections – commentaires

Productivité nominale

Avant évolution : 50 KLS	Après évolution : 75 KLS
$Effort = 3(50)^{1,12} = 240hm = 20ha$ $p = \frac{50}{20} = 2{,}50 KLS/ha$	$Effort = 3(75)^{1,12} = 378hm = 32ha$ $p = \frac{75}{32} = 2{,}34 KLS/ha$

Amélioration de la productivité

Avant évolution : 50 KLS	Après évolution : 75 KLS
H1 (Effort = 139 hm =11,60 ha) ; Durée : 1,12 ans ; équipe = 10 personnes. $p = \frac{2{,}50}{0{,}58} = 4{,}31 KLS/ha$	H1 (Effort = 219 hm =18,25 ha) ; Durée : 1,31 ans ; équipe = 14 personnes. $p = \frac{2{,}34}{0{,}58} = 4{,}03 KLS/ha$
H2 (Effort = 84 hm =7,00 ha) ; Durée : 0,95 an ; équipe = 7 personnes. $p = \frac{2{,}50}{0{,}35} = 7{,}14 KLS/ha$	H2 (Effort = 132 hm = 11,21 ha) ; Durée : 1,11 ans ; équipe = 10 personnes. $p = \frac{2{,}34}{0{,}35} = 6{,}69 KLS/ha$

NB : il s'agit ***d'effectif moyen***.

Reconstituer le profil de distribution des efforts du projet par phase, dans les deux cas et dans les deux hypothèses, en vous servant du tableau 6-8 de l'annexe COCOMO du cours. Proposer un profil d'équipe en faisant l'hypothèse que l'équipe démarre avec 3 personnes et que l'effectif varie ensuite en fonction des efforts à fournir.

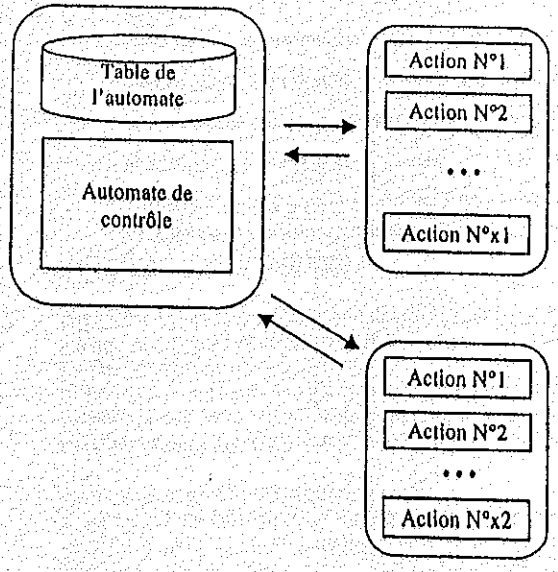
Corrections – commentaires

On utilise la table 6-8, mode P, taille 128, soit :

	Avant évolution : 50 KLS		Après évolution : 75 KLS	
	H1 (11,60 ha)	H2 (7 ha)	H1 (18,61 ha)	H2 (11,21 ha)
Conception générale (17%)	2			
Conception détaillée (24%)	2,8			
Programmation/Tests unitaires (31%)	3,6			
Intégration et tests (28%)	3,2			

Question N° 2 (3 points)

Le concepteur du projet de développement du sous-système BLI a décidé de structurer le BLI à l'aide d'un automate de contrôle (cf. cours conception) piloté par une table. L'architecture du programme BLI peut être schématisé comme suit :



La partie automate regroupe l'essentiel du contrôle. C'est un code qui contient beaucoup d'instructions du type `IF cond THEN ...` ; ce code à donc un nombre cyclomatique très élevé car il contient l'essentiel de la combinatoire du programme.
La partie action est faite de séquence d'instructions impératives, comportant très peu d'instruction `IF cond THEN ...` (car le concepteur a délibérément décidé de les concentrer dans l'automate) ; le nombre cyclomatique de ce code est en général égal à 1, avec quelques séquences à 2 ou 3.

Peut-on considérer que le code correspondant aux blocs d'actions est de même complexité que le code du module automate de contrôle ? Est-il justifié de considérer que le code des actions est d'une complexité qui justifie le choix du type P ? Donner des arguments justifiant l'une ou l'autre des appréciations.

Corrections – commentaires

Compte tenu du bon choix de l'architecte, seul le code de l'automate doit être considéré de type P. Le code des actions qui a un très faible nombre cyclomatique, c'est à dire très peu de IF, est de ce fait très facile à tester. C'est un code de type S. Si les actions on été conçues indépendantes, il n'y a aucune interaction entre elle ; c'est un cas très intéressant où l'on peut prendre l'exposant de l'équation = 1, d'où l'intérêt de piloter la programmation par un automate de contrôle.
Globalement, on peut considérer que le BLI est à un niveau intermédiaire entre S et P, mal plus près de S que de P compte tenu du style de programmation par automate.

En fonction de l'analyse de la complexité du BLI, le concepteur du BLI propose au directeur de projet de rectifier l'exposant de l'équation d'effort et de choisir une valeur intermédiaire entre 1,05 et 1,12. Il propose de faire une moyenne pondérée à partir du volume respectif de l'automate et des actions, en faisant l'hypothèse que pour les 50 KLS initial, la répartition est 40% pour l'automate et 60% pour les actions. Calculer les volumes de code respectif ?
Selon vous, l'ajout des 25 KLS pour les évolutions du BLI augmente-t-il l'automate, ou uniquement les actions.

Corrections – commentaires

Volume automate : 20 KLS
Volume des actions : 30 KLS
L'écart des exposants est : $1,12-1,05=0,07$; les actions représentent 60% de cet écart, on peut donc prendre un exposant égal à $1,05+0,07\times0,4=1,078\approx1,08$
L'ajout des 25 KLS augmentent essentiellement le bloc des actions (aux appels près des nouvelles actions qui nécessitent de modifier l'automate). Dans ce cas, la pondération devient :
automate $\frac{20}{75}=0,27$, actions $\frac{55}{75}=0,73$; soit très proche de 25% — 75%.
L'exposant correspondant, en appliquant la même méthode de calcul, est égal à $1,05+0,07\times0,25=1,0675\approx1,07$.
NB : Si l'on prend en compte le fait que les actions sont indépendante les unes des autres, l'exposant de l'équation pour les actions est égal à 1, ce qui donnerait une équation avec un exposant ≈ 1,02 ou 1,03.

Recalculer les efforts en utilisant les équations d'effort ainsi rectifiées, dans chacune des hypothèses de la question N°1 (élevé et très élevé). Quelle est l'influence de l'exposant

Corrections – commentaires

Nouvelles équations pour BLI à 50 KLS :
H1 : $E_{recalé}=1,74(KISL)^{1,08}$, soit $E(H1)=119hm\approx10ha$
H2 : $E_{recalé}=1,05(KISL)^{1,08}$, soit $E(H2)=72hm\approx6ha$

Nouvelles équations pour BLI à 75 KLS :
H1 : $E_{recalé}=1,74(KISL)^{1,07}$, soit $E(H1)=177hm\approx15ha$; avec 1,08 : 184hm, soit une erreur de 7hm, soit 3,8% d'erreur.

H2 : $E_{recalé}=1,05(KISL)^{1,07}$, soit $E(H1)=107hm=9ha$; avec 1,08 : 112hm, soit une erreur de 5hm, soit 4,5% d'erreur.
L'erreur induite par l'exposant est dans la marge d'erreur du modèle !

Question N° 3 (3 points)
Dans cette question on se propose d'analyser la vraisemblance de l'ajout des 25 KLS qui permettent d'instrumenter de BLI (initialement calibré à 50 KLS) et d'offrir des services d'aides aux diagnostics. Pour faire son estimation, le chef de projet raisonne comme suit :
20 % du code source est constitué d'instructions déclaratives ; les 80% restants sont des instructions impératives.
Les appels au module d'instrumentation sont, en moyenne placés :
a) toutes les 5 LS, ou
b) toutes les 10 LS.
Il faut en moyenne 2 LS par appel.
Le module d'instrumentation gère une trace dans un fichier circulaire ; il dispose d'une fonction d'interrogation à distance pour la télémaintenance et la surveillance par les administrateurs de l'exploitation. Tout ce qui transite sur le BLI est noté dans une base de données simple afin de reconstituer les historiques, si nécessaires en cas de défaillance/panne. La taille de la fonction est estimée à 8 ou 10 KLS.
Enfin, le module d'instrumentation dispose d'une fonction d'édition sur l'écran, sur l'imprimante et sur un format permettant la récupération des informations dans un tableur EXEL. La taille de cette fonction est estimée à environ 2 KLS.

Calculer la taille du module en fonction des deux hypothèses a) et b). Vous supposerez que la proportion des instructions déclaratives est la même que précédemment. Est-ce que cela vous paraît compatible avec les 25 KLS estimées initialement ?
Le type de ce code est-il S, P ou B ?
Quelle productivité peut-on envisager pour la réalisation d'un module de ce type ? Peut-on faire mieux que ce qui a été calculé à la question 1, et si oui pourquoi ?

Corrections – commentaires

Le volume des instructions impératives est : $50\times0,8=40KLS$.
1 trace toutes les 5 KLS nécessite 8.000 appels, soit 16 KLS d'instructions impératives, auxquelles il faut rajouter $\frac{16}{4}=4KLS$ d'instructions déclaratives, soit au total 20 KLS.
1 trace toutes les 10 KLS nécessite 4.000 appels, soit 8 KLS d'instructions impératives, auxquelles il faut rajouter $\frac{8}{4}=2KLS$ d'instructions déclaratives, soit au total 10 KLS.
Le code d'instrumentation, selon la finesse des traces est donc compris entre 22 KLS et 32 KLS.
C'est donc tout à fait compatible avec les 25 KLS initialement estimées.

Ce code est clairement de type S car le programme à instrumenter est la spécification de l'instrumentation. De plus, il ne s'agit que d'éditions simples sans algorithme. Ce code n'a pas besoin d'une documentation particulière, et les tests pour le valider sont par définition très simples, on doit donc pouvoir améliorer fortement la productivité car tous les facteurs de coût sont à corriger dans le bon sens. L'effort après évolution doit donc être revu à la baisse, par exemple :
25 KLS produit avec une productivité de 12 KLS/ha rajoute 2 ha d'effort, soit 13,60 ha ou 9 ha en corrigeant les résultats de la question 1.

faciliter l'identification et la correction des défauts. L'effort nécessaire à ce développement doit être compensé par une amélioration importante de la durée moyenne de correction des défauts (cf. la table HP en annexes). Au total, l'effort doit donc être moindre !

Question N° 4 (3 points)

On se propose maintenant d'analyser en détail le contenu de l'effort de test du code fonctionnel du BLI. Le code de l'automate a été obtenu à partir d'un générateur de code qui transforme automatiquement les diagrammes réalisés initialement en UML dans la phase de conception. Seules les actions sont à valider exhaustivement.

L'estimation du décompte des Tests Unitaires TU est effectuée en supposant qu'il faut 1 test unitaire pour en moyenne 10 *LS* ; les TU sont regroupés en scénarios de tests, soit en moyenne 50 tests par scénario.

Calculer le nombre de TU nécessaires ainsi que le nombre de scénarios.

Quel est le coût moyen d'un test dans chacune des hypothèses H1 et H2 (pour cela vous utiliserez les résultats obtenus à la question 1) ?

Est-ce que l'effort disponible vous paraît raisonnable par rapport au travail à effectuer (considérer le cycle de développement des tests dans le cours test) ?

Que va-t-il se passer si le dépouillement des tests est manuel (en particulier, calcul du nombre de fois que les scénarios seront exécutés) ?

Corrections – commentaires

Le générateur de code pour l'automate de contrôle est un outil du type LEX-YACC (disponible sans licence sur UNIX-LINUX). Il permet de générer automatiquement, et sans erreur, le code source correspondant à partir d'une grammaire qui a été élaborée dans la phase de conception (par exemple un automate en UML).

De plus, sur les 75 KLS que compte le BLI avec son instrumentation, seules les 30 KLS des actions initiales sont véritablement fonctionnelles.

L'effort de test porte donc principalement sur le code des actions. 1 TU pour 10 LS génère 3.000 TU, soit 60 scénarios de tests comportant chacun 50 TU, en moyenne.

L'effort de test, en prenant 15% pour la partie P/TU, donne 15+28=43%, soit :

1. H1 : $0{,}43\times11{,}6 \approx 5ha = 60hm = 1320hj$
2. H2 : $0{,}43\times7 \approx 3ha = 36hm = 792hj$

Le coût de fabrication de 1 TU, en moyenne, est donc : $\frac{1320}{3000} = 0{,}44hj$ ou $\frac{792}{3000} = 0{,}26hj$

Ce coût inclut le développement et l'exécution du TU.

Cet effort n'est pas aberrant dans la mesure où il est raisonnable de penser qu'il y a de la redondance, donc potentiellement des économies d'échelle et des factorisations possibles.

Si le dépouillement est manuel, l'effort est très insuffisant car le nombre de rejeu est $\frac{60\times59}{2} - 60 = 1710$ passages.

nus

estion N° 5 (2 points)

Entre la fin de programmation et la mise en exploitation, les statistiques de tests nous indiquent qu'il faut découvrir, en moyenne, pour des programmeur de niveau normal, 80 erreurs par KLS de façon à démarrer l'exploitation avec un taux d'erreurs résiduelles acceptable. En utilisant la table HP ci-dessous, quel est le coût moyen de la découverte de ces erreurs dans le code fonctionnel du BLI ?

Selon vous, la répartition des erreurs est elle la même dans la partie automate et dans la partie actions ? Justifier votre réponse.

Comment traduire le fait que l'équipe, comme il est dit dans l'énoncé, est très expérimentée ?

Comment peut-on rectifier le calcul précédent (faites des hypothèse de productivité, en les justifiant) ?

Par rapport au profil du projet, ce nouveau décompte vous paraît-il raisonnable ?

Corrections – commentaires

Selon le tableau HP, une erreur coûte en moyenne 6,3 heures. Car :

$25\times2+50\times5+20\times10+4\times20+1\times50 = 630$ *heures* pour 100 défauts.

Si l'on prend 80 défauts à purger en moyenne par KLS, il y aura dans le BLI $80\times30 = 2400$ défauts à traiter, soit, selon la statistique HP :

$2400\times6{,}3 = 15.120heures = 1.890hj = 86hm = 7ha$.

L'automate a été conçu lors de la conception du BLI ; puis il a été généré automatiquement. Le code d'instrumentation n'intervient pas car il n'est pas fonctionnel, quoique, en toute théorie il faudrait lui affecter qq. défauts.

Cette valeur est très supérieure à ce qui a été calculé en Q1 et Q2, mais cela s'explique par le fait que ***l'équipe est très expérimentée*** ; elle fait donc beaucoup moins d'erreurs, par exemple 15 ou 20 au lieu de 80, et plus particulièrement beaucoup moins d'erreurs coûteuses à réparer. La statistique HP doit donc être rectifiée en conséquence.

Question N° 6 (2 points)

Selon vous, comment faut-il affecter les membres de l'équipe sur les différentes fonctions et actions, par rapport aux compétences des programmeurs (voir l'effectif calculé à la question 1) ?

L'effectif au démarrage est-il suffisant pour faire la conception ?

Que se passe t-il, en terme de productivité, si l'architecte, au lieu de séparer le code de l'automate de celui des actions, laisse les programmeurs sans directive précise, ce qui conduira à une imbrication des deux codes ?

Que se passe-t-il en terme d'effectif à mettre sur le projet si l'on comprime les délais de réalisation de 30% ?

Corrections – commentaires

Il est clair que les membres les plus expérimentés de l'équipe doivent être affectés à la conception de l'automate de contrôle (en fait, 1 programmeur), car tout le projet dépend de la bonne structuration de cet automate. Si l'architecte n'est pas aussi expérimenté qu'il le dit, le mélange des 2 codes fera que tout redeviendra de type P, ce qui conduit directement à une très grave sous-estimation du projet, et donc des retards prévisibles. Un compression des délais augment l'effectif instantané, donc les communications entre les membres de l'équipe, ce qui va dégrader la productivité.

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 ?

DEUXIÈME PARTIE

LANGAGE UNIFIÉ DE MODÉLISATION

(L.U.M. / UML)

La plupart de ces exercices sont adaptés de l'ouvrage ***UML par la pratique****, où l'on pourra retrouver des corrigés-types.*

Conception avec U.M.L.

I - Analyse: Le point de vue fonctionnel

EXERCICE I:

On considère un système simplifié de Guichet Automatique de Banque (G.A.B.). Le GAB offre les services suivants:

1. Distribution d'argent à tout porteur de carte de crédit (carte Visa, ou de la banque), via un lecteur de carte et un distributeur de billets.
2. Consultation de solde de compte, depôt en numéraire et depôt de chèques pour les clients de la banque porteurs d'une carte de crédit de la banque.

Par ailleurs:

- Toutes les transactions sont sécurisées, via:
  - le Système d'Autorisation Visa, pour les transactions de retrait effectuées avec une carte Visa;
  - le Système d'Information de la banque, pour autoriser toutes les transactions effectuées par un client avec sa carte de la banque, mais également pour accéder au solde des comptes.
- Le GAB nécessite des actions de maintenance, telles que le rechargement en billets du distributeur, la récupération des cartes avalées et des chèques déposés, etc

Attention! L'énoncé précédent est sans doute incomplet et imprécis, comme il en est dans les projets réels.

QUESTIONS:

1. **Identification des acteurs:**
   a. Identifiez les principaux ACTEURS du GAB.
   b. Elaborez le DIAGRAMME DE CONTEXTE STATIQUE du GAB.
2.
3. **Réalisation du diagramme des cas d'utilisation:**
   a. Proposez une liste préliminaire des cas d'utilisation du GAB, par acteur.
   b. Donnez un DIAGRAMME préliminaire DES CAS D'UTILISATION. Proposez-en une version plus élaborée, utilisant le concept de généralisation, que l'on discutera.
   c. Complétez le DIAGRAMME DES CAS D'UTILISATION préliminaire en y ajoutant les ACTEURS SECONDAIRES. *(Pour simplifier, on n'y tiendra pas compte de l'Opérateur de maintenance.)*
4.
5. Donner une description textuelle des cas d'utilisation, avec la DESCRIPTION DES ENCHAÎNEMENTS.
6. Description grahique des Cas d'utilisation: réalisez un diagramme de séquence Système qui décrit le scénario nominal du cas d'utilisation: *Retirer de l'argent avec une carte Visa.*
7.
8. **Organisation des cas d'utilisation:**
   a. Identifiez une partie commune aux différents cas d'utilisation, et factorisez-la dans un nouveau cas inclus dans ces derniers.
   b. Identifiez une relation de **généralisation** qui implique 2 cas d'utilisation du client de la banque.

EXERCICE 2

Cet exercice traite d'un système simplifié de caisse enregistreuse de supermarché. Le déroulement normal d'utilisation de la caisse est le suivant:

- Un client arrive à la caisse avec des articles à payer.
- Le caissier enregistre le n° d'identification de chaque article, ainsi que la quantité si elle est supérieure à 1.
- La caisse affiche le prix de chaque article et son libellé.
- Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente.
- La caisse affiche le total des achats.
- Le client choisit son mode de paiement.
  1. Liquide: le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client.
  2. Chèque: le caissier vérifie la solvabilité du client en transmettant une requêteà un centre d'autorisation via la caisse.
  3. Carte de crédit: un terminal bancaire fait partie de la caisse. Il transmet une demande d'autorisation en fonction du type de la carte.
- La caisse enregistre la vente et imprime un ticket.
- Le caissier donne le ticket de caisse au client.

Lorsque le paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion des stocks.

Tous les matins, le responsable du magasin initialise les caisses pour la journée.

QUESTIONS

1. Elaborez un Diagramme de Cas d'Utilisation détaillé de la Caisse enregistreuse.N'hésitez pas à utiliser les relations entre cas d'utilisation pour rendre le diagramme plus précis

2. Ecrivez une description textuelle détaillée du cas d'utilisation principal: *Traiter le passage en caisse*.

3. Réalisez un diagramme de séquence système qui décrive le scénario nominal du cas d'utilisation *Traiter le passage en caisse*, en ne considérant que le paiement en liquide.

4. Montrez, par un Diagramme d'Etats, la succession forcée des opérations système pour le cas d'utilisation *Traiter le passage en caisse*, en ne considérant toujours que le paiement en liquide.

Conception avec U.M.L.

II - Analyse: Point de vue statique

Les Diagrammes de Classe UML

EXERCICE I:

On considère un système simplifié de réservation de vols pour une agence de voyages.

Des entrevues avec un certain nombre d'experts du métier, on peut tirer les éléments de connaissance suivants sur le domaine:

1. Des compagnies aériennes proposent différents vols.
2. Un vol est ouvert à la réservation et refermé sur ordre de la Compagnie.
3. Un client peut réserver des places sur un ou plusieurs vols, pour des passagers différents.
4. Une réservation concerne un seul passager sur un seul vol.
5. Une réservation peut être annulée ou confirmée.
6. Un vol a un aéroport de départ et un aéroport d'arrivée.
7. Un vol a un jour et une heure de départ, un jour et une heure d'arrivée.
8. Un vol peut comporter des escales dans des aéroports.
9. Une escale a une heure d'arrivée et une heure de départ.
10. Chaque aéroport dessert une ou plusieurs villes.

On se propose de réaliser progressivement un **modèle conceptuel statique** à partir des éléments de connaissance décrits par les phrases ci-dessus.

QUESTIONS:

1. Modéliser les phrases 1 et 2 en termes de **classes**, de **comportements** (opérations), et d'**associations** entre elles.
2. **Modélisation des phrases 6, 7 et 10:**
   a. A l'aide de la phrase 7, compléter la description de la classe Vol mise en évidence dans la Q.1, en identifiant ses attributs.
   b. Quelles sont les différentes solutions pour modéliser la phrase 6? Donner leurs avantages et leurs inconvénients.
   c. Quelle est la multiplicité du côté de la classe Aéroport impliquée par la modélisation de la phrase 10?
3. **Modélisation des phrases 8 et 9:**
   a. Effectuez une modélisation préliminaire des phrases 8 et 9 à l'aide d'une nouvelle classe Escale.
   b. Complétez la multiplicité des associations sur le modèle (3,a).
   c. Proposez une solution plus élaborée pour modéliser les escales.
4. **Modélisation des phrases 3, 4 et 5 traitant du concept de réservation:**
5.
   a. Faut-il distinguer les concepts de client et de passager? Complétez alors le modèle à l'aide de nouvelles classes *ad hoc*.
   b. Modélisez la phrase 3 et complétez les multiplicités des associations entre ces classes.
6. Répertorier les attributs indispensables pour chacune des classes du modèle, en suivant les conventions de nomination UML.

EXERCICE 2: Relations structurelles entre classes.

Soient les phrases suivantes:

1. Un répertoire contient des fichiers.
2. Une pièce contient des murs.
3. Les modems et les claviers sont des périhériques d'entrée-sortie.
4. Une transaction boursière est un achat ou une vente.
5. Un compte bancaire peut appartenir à une personne physique ou morale.

QUESTION

Déterminez la relation statique appropriée (généralisation, composition, agrégation ou association) dans chaque phrase de l'énoncé précédent.

Conception avec U.M.L.

III - Analyse: Point de vue dynamique

EXERCICE I:

On considère un système simplifié de Publiphone à pièces

1. Le prix minimal d'une communication interurbaine est de 40 centimes d'euro.
2. Après l'introduction de la monnaie, l'utilisateur a 2 minutes pour composer son numéro (ce délai est décompté par le standard).
3. La ligne appelée peut être libre ou occupée.
4. Le correspondant peut raccrocher le premier.
5. Le Publiphone consomme de l'argent dès que l'appelé décroche, et à chaque unité de temps (U.T.) engendrée par le standard.
6. On peut ajouter des pièces à tout moment.
7. Lors du raccrochage, le solde de monnaie est rendu.

A partir des 7 phrases ci-dessus, il sera demandé de:

- identifier les Acteurs et les Cas d'utilisation;
- construire un diagramme de séquence système;
- construire le **diagramme de contexte dynamique;**
- élaborer le **diagramme d'états** du Publiphone.

QUESTIONS:

1. Identifier les Acteurs, et dessiner le Diagramme des Cas d'utilisation du Publiphone à pièces.
2. Réaliser un Diagramme (préliminaire) de séquence système qui décrive le Scénario nominal du cas d'utilisation: *Téléphoner*.
3. Enrichir le Diagramme de séquence système précédent avec des activités internes intéressantes, et quelques réponses du Publiphone à l'appelant.
   Pour simplifier, on ne représentera plus la conversation, afin de se concentrer sur les *opérations système*.
4. Sur les 2 Diagrammes de séquence système précédents, répertorier les messages échangés entre le système et les acteurs, et les généraliser en ajoutant des paramètres si nécessaire.
   Réaliser alors le **diagramme de contexte dynamique** du Publiphone.
5. Réaliser un premier **Diagramme d'états** qui décrive le comportement nominal du Publiphone, d'après le diagramme de séquence système.
6. Sur le Diagramme d'états précédent, comment représenter le fait que l'appelant peut raccrocher à tout moment, et pas seulement dans l'état *Conversation*?
   N.B.: On introduit ici la notion de *super-état*.

128

EXERCICE 2

On considère un réveille-matin simplifié:

1. On peut mettre l'alarme *en* ou *hors fonction*.
2. Quand l'heure courante devient égale à l'heure d'alarme, le réveil sonne (sans s'arrêter, dans un premier temps).
3. On peut interrompre la sonnerie.

QUESTIONS

1. Dessiner le **Diagramme d'états** correspondant.
2. Compléter le Diagramme d'états précédent pour prendre en compte le fait que la sonnerie du réveil s'arrête d'elle-même au bout d'un certain temps.
3. En déduire un **diagramme de contexte statique étendu** qui fasse apparaître les **attributs** et les **opérations** de la classe qui représente le système *Réveil* (vu comme une boîte noire).

Conception avec U.M.L.

IV - Analyse: Point de vue dynamique (*Diagrammes d'états*)

EXERCICE 1:

On considère une montre digitale simplifiée:

Bouton Mode **17:35:22** Bouton Avance

1. Le mode courant est le mode *Affichage*.
2. Quand on appuie une fois sur le bouton **Mode**, la montre passe en mode *modification de l'heure*.Chaque pression sur le bouton **Avance** incrémente l'heure d'une unité;
3. quand on appuie une nouvelle fois sur le bouton **Mode**, la montre passe en mode *modification des minutes*. Chaque pression sur le bouton **Avance** incrémente les minutes d'une unité.
4. quand on appuie une nouvelle fois sur le bouton **Mode**, la montre repasse en mode *Affichage*.

QUESTIONS:

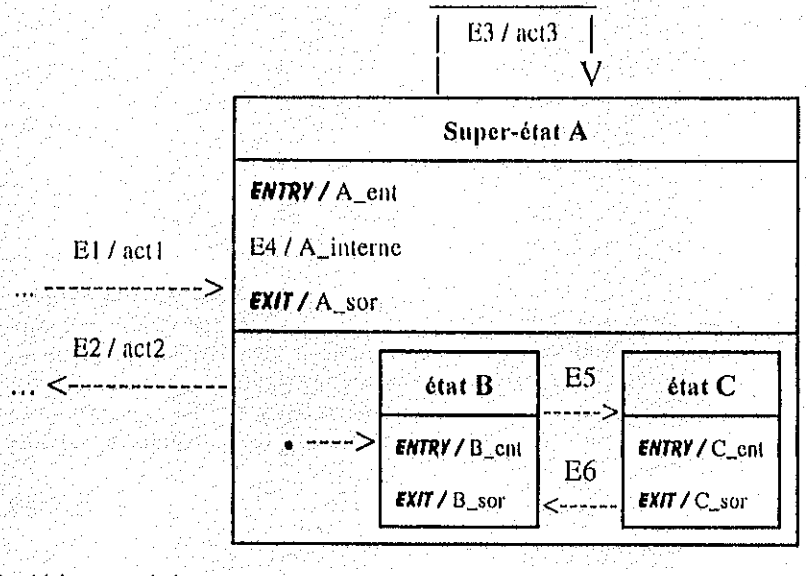
1. Réaliser un **Diagramme d'états** correspondant aux spécifications ci-dessus, sans oublier les actions accompagnant certaines des transitions.
2. On enrichit maintenant les spécifications initiales de la manière suivante:
   Quand on appuie sur le bouton **Avance** plus de 2 secondes, les heures ou les minutes (selon le cas) s'incrémentent rapidement jusqu'à ce qu'on relâche la pression sur le bouton.
   Modifier le Diagramme d'états précédent en conséquence, en introduisant les nouveaux événements (notamment temporels) et états nécessaires.
3. On revient aux spécifications initiales (Q. 1), mais l'on ajoute à la montre digitale 2 nouveaux boutons:
   - un bouton d'éclairage: en le pressant, on éclaire le cadran de la montre jusqu'à ce qu'on relâche ce bouton;
   - un bouton d'alarme, qui confère à la montre une fonctionnalité classique d'alarme: quand l'heure courante devient égale à l'heure d'alarme, la montre sonne (la sonnerie s'arrête d'elle-même au bout d'un certain temps, ou lorsqu'on l'interrompt).

Donner le nouveau **Diagramme d'états** correspondant, incluant tous les comportements de la montre (gestion de: l'affichage, l'alarme, l'éclairage). On envisagera 2 solutions:

- à l'aide d'un Diagramme de classes composite,
- ou avec un Diagramme d'états à **régions concurrentes** (ou: à vecteur d'états).

EXERCICE 2: Diagramme d'états hiérarchique complexe

Soit le fragment de diagramme d'états suivant, contenant un certain nombre d'actions:



où **En** désigne un *événement*, et **actn** (ou L_xxx) une *action* accompagnant une transition.

QUESTION

Compléter le tableau ci-après. On suppose que l'on part de l'état à gauche du diagramme symbolisé par ... ; à partir de la ligne 2, l'état de départ est l'état d'arrivée de la ligne précédente.

État de départ	Événement	Action(s)	État d'arrivée
...	E1	?	*? (dans A)*
?	E5	?	?
?	E4	?	?
?	E6	?	?
?	E3	?	?
?	E5	?	?
?	E3	?	?
?	E2	?	...

TABLEAU CORRIGÉ DE L'EXERCICE 2

État de départ	Événement	Action(s)	État d'arrivée
...	E1	**act1, A_ent, B_ent**	**B** *(dans A)*
B	E5	**B_sor, C_ent**	**C**
C	E4	**A_interne**	(C)
C	E6	**C_sor, B_ent**	**B**
B	E3	**B_sor, A_sor, act3, A_ent, B_ent**	**B**
B	E5	**B_sor, C_ent**	**C**
C	E3	**C_sor, A_sor, act3, A_ent, B_ent**	**B**
B	E2	**B_sor, A_sor, act2**	**...**

Conception avec U.M.L.

Un organisme de recherche est constitué de **départements scientifiques** couvrant les grands domaines de la recherche. À chaque département sont rattachées un certain nombre d'**unités de recherche** dont il assure la gestion.

Les activités considérées sont regroupées en trois secteurs:

- la gestion proprement dite des unités;
- la gestion de l'activité scientifique et technique des unités;
- la gestion des personne(l)s des unités.

On n'envisage ici que le secteur *Gestion des unités*.

Chaque **unité** se caractérise par un type d'unité (unité *propre*, *associée* ou *mixte*), un numéro, un intitulé, la date de proposition de création, la date de création effective et la date de fermeture. Elle possède en outre une adresse principale et, dans certains cas, une autre adresse dite secondaire. Une adresse est constituée des données suivantes: code du type d'adresse (principale ou secondaire), adresse postale, adresse électronique (*mel*, ou *courriel* pour nos amis et cousins québécois), téléphone, nom de l'organisme et nom de la délégation régionale.

À une unité donnée sont affectées une à plusieurs personnes, dont un à trois **responsables** pilotant l'unité.

Enfin, une unité est rattachée à une ou plusieurs sections d'évaluation de la recherche. Chaque section, caractérisée par un numéro et un libellé, peut évaluer une ou plusieurs unités.

La **gestion des unités** consiste, pour un département scientifique, à créer ou modifier ou supprimer une unité, et à en visualiser toutes les données.
Pour une unité, la gestion de l'unité consiste à modifier l'adresse secondaire et à visualiser toutes ses données.

On demande d'élaborer une modélisation par objets se ce Système d'Information à l'aide d'U.M.L.

QUESTIONS:

1. Identifier les **acteurs**; identifier et représenter les **cas d'utilisation** du système *Gérer les unités*.
2. Identifier puis décrire les différents **scénarios** des cas d'utilisation de la Q. 1. On pourra utiliser, pour représenter graphiquement les scénarios, un **Diagramme de Séquence système**, ou un **Diagramme de Collaboration**.
3. Identifier et nommer les **Classes d'objets** mises en jeu dans le modèle.
4. Élaborer le **Diagramme de classes** du modèle.
5. Donner le **diagramme d'états-transitions** décrivant le comportement d'une Unité.

EXAMEN DE GÉNIE LOGICIEL 2002

EXERCICE DE L.U.M. / UML

On souhaite modéliser en L.U.M./UML le comportement (très simplifié...) d'une calculette - convertisseur francs / euros.

On suppose (pour.. compliquer quand même!) que le taux de conversion entre les monnaies n'est pas figé en "dur": l'appareil doit initialement être configuré par un "opérateur" qui enregistre une fois pour toutes ce taux dans la calculette.

La calculette dispose, en plus de cette fonction de conversion, des fonctions de calcul arithmétique de base usuelles (+, -, *, /).

QUESTIONS:

1. Quels sont les (deux) **acteurs** du modèle? Identifier les (trois) **cas d'utilisation** principaux de ces acteurs, puis donner le **diagramme de contexte** de la calculette.

2. Donner une description textuelle précise, mais succincte (moins d'une demi-page...) de chacun des cas d'utilisation de la Q. 1. On considérera les "exceptions" (scénarios alternatifs) suivantes: "Taux de conversion non fourni" pour une conversion francs / euros, et "Annulation" (par une touche d'annulation) dans tous les cas (on ignorera les "divisions par zéro", et autres erreurs de calcul).

3. Déduire de la description précédente un **diagramme de séquence-système** pour un scénario de **calcul** nominal, ainsi que pour un scénario de calcul alternatif "Annulation".

4. Établir un **diagramme d'etats - transitions** (à 2 états...) pour les scénarios de calcul.

5. Quels sont, selon vous, les (3) classes du "système" indispensables pour modéliser le déroulement du scénario de "calcul" (la calculette est un "petit ordinateur"...).

TROISIÈME PARTIE

TEST du LOGICIEL

(Méthode des Couvertures)

avec une large sélection de corrigés

22/09/99

Cnam

Génie Logiciel
EXAMEN 93-94

1ère session
15 Juin 1994

Durée de l'épreuve : 3 heures
Tous documents autorisés

Chaque sujet doit être développé sur une copie séparée
vous devez donc rendre 3 copies

Notation :
- Synthèse de cours : /5
- Exercice de modélisation de système : /7
- Exercice de programmation : /5
+ note obtenue en ED/5
troncature à 20

1. Question de cours

Étude de cheminement dans un programme Ada

A partir du programme Ada ci-joint, vous réaliserez les opérations suivantes:

1. Construire le graphe de contrôle associé à ce programme.
2. Calculer le nombre associé au graphe de contrôle.
3. Déterminer, en examinant les conditions de chemins, l'ensemble des valeurs initiales qu'il faut fournir au programme de façon à vérifier que:

a) tous les noeuds du graphe sont parcourus au moins une fois (Couverture CO).

b) Tous les arcs du graphe sont parcourus au moins une fois (Couverture C1).

Dans les 2 cas, qu'en déduisez vous par rapport au nombre cyclomatique.

136

```
pragma title("Tri de SHELL");
        --*****************************************
        --*    TRI PAR LA METHODE DE   S H E L L    *
        --*****************************************

generic
  type ELEMENT is private;                    -- type des elements a trier.
  type VECTEUR_D_ELEMENTS is array (NATURAL range <>) of ELEMENT;
  with function "<" (elt1,elt2: ELEMENT) return BOOLEAN is <>;

procedure TRI_SHELL (VECTEUR: in out VECTEUR_D_ELEMENTS);

pragma title("Tri de SHELL");
        --*****************************************
        --*    TRI PAR LA METHODE DE   S H E L L    *
        --*****************************************

procedure TRI_SHELL (VECTEUR: in out VECTEUR_D_ELEMENTS) is

  incr: natural:=VECTEUR'length;   -- suite des increments degressifs.

  inf: constant natural:=VECTEUR'first;
  L_element: ELEMENT;
  K:integer;

begin           -- corps du tri.

  while incr > 1 loop
        if incr < 5 then incr:=1;
                    else incr:= (5*incr-1) / 11;
        end if;

     -- Invariant de la boucle L:
     --   Les INCR sous_files imbriquees de VECTEUR (inf:L-1)
     --   sont ordonnees.

        for L in inf+incr .. VECTEUR'last loop
            L_element:= VECTEUR (L);              -- sauvegarde.

                -- Recherche sequentielle retrograde de la place
                -- de L_ELEMENT, et decalage en parallele.

            K:=L-incr;
            while K >= inf loop
                if L_element < Vecteur (k) then
                   VECTEUR (k+incr):= VECTEUR (k);
                   k:=k-incr;
                else
                   exit;                          -- fin du decalage.
                end if;
            end loop;    -- k.
            VECTEUR (k+incr):=L_element;          -- insertion.
        end loop;        -- l.
   end loop;             -- incr.

end  TRI_SHELL;
```