Découpe et contrats en génie logiciel

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 14 janvier 2021

LAMSADE
UMR CNRS 7243
laboratoire d'analyse et modélisation de systèmes pour l'aide à la décision

DAUPHINE
UNIVERSITÉ PARIS

PSL
RESEARCH
UNIVERSITY

Découpe

- Problèmes résolus par découpe en sous-problèmes
- Logiciels : pouvoir s'appuyer sur des blocs dont on oublie les détails
- Exemple : calcul d'une racine carrée
- Organisation d'une entreprise commerciale ?

- Organisation d'une société humaine ?

- Découpe parfois peu étanche !
- Raisonnement mathématique

Découpe

- Problèmes résolus par découpe en sous-problèmes
- Logiciels : pouvoir s'appuyer sur des blocs dont on oublie les détails
- Exemple : calcul d'une racine carrée
- Organisation d'une entreprise commerciale ?
- Découpe en services
- Exemple : secrétariat fournit des accessoires
- Organisation d'une société humaine ?

- Découpe parfois peu étanche !
- Raisonnement mathématique

Découpe

- Problèmes résolus par découpe en sous-problèmes
- Logiciels : pouvoir s'appuyer sur des blocs dont on oublie les détails
- Exemple : calcul d'une racine carrée
- Organisation d'une entreprise commerciale ?
- Découpe en services
- Exemple : secrétariat fournit des accessoires
- Organisation d'une société humaine ?
- Système judiciaire, exécutif, législatif
- Découpe parfois peu étanche !
- Raisonnement mathématique

Découpe en informatique

- En fait, que se passe-t-il lors du calcul d'une racine carrée, d'une lecture de fichier, . . . ?

Découpe en informatique

- En fait, que se passe-t-il lors du calcul d'une racine carrée, d'une lecture de fichier, ... ?
- Appel à la JDK
- Appel système d'exploitation (éventuellement)
- Appel au processeur
- Chacun résoud des sous-problèmes pour qqn d'autre
- Exemples de gros sous-problèmes déjà résolus pour vous dans tout langage de programmation de haut niveau ?

Découpe en informatique

- En fait, que se passe-t-il lors du calcul d'une racine carrée, d'une lecture de fichier, ... ?
- Appel à la JDK
- Appel système d'exploitation (éventuellement)
- Appel au processeur
- Chacun résoud des sous-problèmes pour qqn d'autre
- Exemples de gros sous-problèmes déjà résolus pour vous dans tout langage de programmation de haut niveau ?
- Structures de données : listes, etc.

Interfaces

- Interface entre l'implémentation et le monde extérieur
- Penser à l'inter-face entre une cellule et le monde (sa membrane)
- Analogue à une entreprise : interface secrétariat
- GUI ?
- Multiples interfaces possibles pour un même service
- Exemple entreprise ?
- Exemple radiateur ?

- Exemple voiture ?

Interfaces

- Interface entre l'implémentation et le monde extérieur
- Penser à l'inter-face entre une cellule et le monde (sa membrane)
- Analogue à une entreprise : interface secrétariat
- GUI ? Graphical User Interface
- Multiples interfaces possibles pour un même service
- Exemple entreprise ?
- Exemple radiateur ?

- Exemple voiture ?

Interfaces

- Interface entre l'implémentation et le monde extérieur
- Penser à l'inter-face entre une cellule et le monde (sa membrane)
- Analogue à une entreprise : interface secrétariat
- GUI ? Graphical User Interface
- Multiples interfaces possibles pour un même service
- Exemple entreprise ? Formulaire, demande orale, ...
- Exemple radiateur ?

- Exemple voiture ?

Interfaces

- Interface entre l'implémentation et le monde extérieur
- Penser à l'inter-face entre une cellule et le monde (sa membrane)
- Analogue à une entreprise : interface secrétariat
- GUI ? Graphical User Interface
- Multiples interfaces possibles pour un même service
- Exemple entreprise ? Formulaire, demande orale, ...
- Exemple radiateur ? Boutons différents, à différents endroits, commande vocale, ...
- Exemple voiture ?

Interfaces

- Interface entre l'implémentation et le monde extérieur
- Penser à l'inter-face entre une cellule et le monde (sa membrane)
- Analogue à une entreprise : interface secrétariat
- GUI ? Graphical User Interface
- Multiples interfaces possibles pour un même service
- Exemple entreprise ? Formulaire, demande orale, ...
- Exemple radiateur ? Boutons différents, à différents endroits, commande vocale, ...
- Exemple voiture ? Boite automatique VS boite manuelle

Interfaces pour différents utilisateurs

- Une même entité peut avoir différents niveaux d'interface
- Interface plus simple, interface plus complète
- Exemples ?

- Interface pour utilisateur final, interface pour assembleur
- Exemples ?

Interfaces pour différents utilisateurs

- Une même entité peut avoir différents niveaux d'interface
- Interface plus simple, interface plus complète
- Exemples ?
- Accès difficiles (réservés aux parents) sur jouet pour enfant
- Deux systèmes sur lave-vaisselles
- Menu de configuration de votre télévision
- Interface pour utilisateur final, interface pour assembleur
- Exemples ?

Interfaces pour différents utilisateurs

- Une même entité peut avoir différents niveaux d'interface
- Interface plus simple, interface plus complète
- Exemples ?
- Accès difficiles (réservés aux parents) sur jouet pour enfant
- Deux systèmes sur lave-vaisselles
- Menu de configuration de votre télévision
- Interface pour utilisateur final, interface pour assembleur
- Exemples ? Voiture (conduire VS entretenir) ; Système d'exploitation (utiliser VS installer des logiciels)

Interfaces pour le programmeur

- Vous développez des sous-routines
- Ces sous-routines sont accessibles à des programmeurs
- Y compris vous-même !
- Elles peuvent être combinées pour créer différents programmes pour utilisateur final
- Elles peuvent être inspectées en cas de bug
- API ?

- Une interface peut aussi être un ensemble d'interfaces !
- Exemple ?

Interfaces pour le programmeur

- Vous développez des sous-routines
- Ces sous-routines sont accessibles à des programmeurs
- Y compris vous-même !
- Elles peuvent être combinées pour créer différents programmes pour utilisateur final
- Elles peuvent être inspectées en cas de bug
- API ? Application Programming Interface
- Accessible par programme (API $\neq$ End-user Interface)
- Une interface peut aussi être un ensemble d'interfaces !
- Exemple ?

Interfaces pour le programmeur

- Vous développez des sous-routines
- Ces sous-routines sont accessibles à des programmeurs
- Y compris vous-même !
- Elles peuvent être combinées pour créer différents programmes pour utilisateur final
- Elles peuvent être inspectées en cas de bug
- API ? Application Programming Interface
- Accessible par programme (API $\neq$ End-user Interface)
- Une interface peut aussi être un ensemble d'interfaces !
- Exemple ? API de Java

Contrat

- Découpe en sous-problèmes résolus par des services
- Service fonctionne sous certaines conditions
- Contrat : clarification des devoirs de l'utilisateur et du fournisseur de service
- entre appelant et programmeur de la sous-routine
- Devoirs appelés *préconditions*
- Exemple : entier fourni en paramètre > 0
- Sous ces conditions, méthode fournit un service
- Si conditions non remplies : pas de garanties offertes !
- *Postconditions* : garanties offertes en retour
- Exemple : renvoie un nombre aléatoire entre 0 et l'entier fourni, exclu

Contrat à expliciter

- Contrat facilite l'implémentation de la sous-routine
- Contrat facilite la vie de l'utilisateur
- À condition de rendre le contrat explicite
- Documenter les préconditions et postconditions
- Utilisateur averti : pensera plus probablement à vérifier les préconditions

Échec rapide

- Principe de l'*échec rapide* (*fail-fast*)
- Mieux vaut une erreur immédiate qu'une action inattendue
- Évite les conséquences catastrophiques
- Facilite les corrections de bug
- Deux mises en œuvre : programmation défensive (erreur de l'utilisateur) ; programmation prudente et explicite (erreur du programmeur)

Programmation défensive

- Aider les utilisateurs imprudents
- Échec rapide si précondition non satisfaite
- En pratique : tester les préconditions en entrée de sous-routine

  (sauf si très couteux en temps)

Programmation prudente

- Tester vos déductions à des endroits cruciaux
- Échec rapide si non valide
- Exemple : je sais qu'ici telle valeur devrait être positive

Exceptions

- Interrompt le flux normal de traitement
- Pour gérer une situation exceptionnelle
- Exemple ?

- Différence par rapport à un test (if / else) ?

- Est une forme d'*interruption*
- Exemple d'interruption qui n'est pas une exception ?

Exceptions

- Interrompt le flux normal de traitement
- Pour gérer une situation exceptionnelle
- Exemple ? Erreur utilisateur ; Erreur programmeur ; Mémoire vive épuisée
- Différence par rapport à un test (if / else) ?

- Est une forme d'*interruption*
- Exemple d'interruption qui n'est pas une exception ?

Exceptions

- Interrompt le flux normal de traitement
- Pour gérer une situation exceptionnelle
- Exemple ? Erreur utilisateur ; Erreur programmeur ; Mémoire vive épuisée
- Différence par rapport à un test (if / else) ? Service « normal » non rendu
- Est une forme d'*interruption*
- Exemple d'interruption qui n'est pas une exception ?

Exceptions

- Interrompt le flux normal de traitement
- Pour gérer une situation exceptionnelle
- Exemple ? Erreur utilisateur ; Erreur programmeur ; Mémoire vive épuisée
- Différence par rapport à un test (if / else) ? Service « normal » non rendu
- Est une forme d'*interruption*
- Exemple d'interruption qui n'est pas une exception ? Préemption du processeur par un OS multi-tâches

Problèmes résolus par les exceptions

- Opérations communes souvent faillibles
- Exemple : allouer de la mémoire, écrire sur un fichier
- Si échec, souhait d'interrompre le flux normal
- Par défaut : interrompt totalement le programme
- Mais avec opportunité de traiter l'exception pour se rétablir
- Éviter de devoir écrire une commande spécifique de traitement à chaque invocation

Gestion d'exceptions

- Certaines opérations peuvent lancer une *exception*
- Si pas de gestion spécifique prévue, envoi de l'exception à l'échelon supérieur
- L'appelant peut, alternativement, *gérer* l'exception
- Si pas de gestion du tout : arrêt du programme

En Java (aperçu)

- Java utilise des `Throwable` pour représenter tout ce qui peut être lancé pour interrompre le flux de traitement
- Choisir le type de `Throwable`, par exemple `IllegalArgumentException`
- Créer une exception qui représente votre problème : `IllegalArgumentException exc = new IllegalArgumentException("Positive value required.");`
- La variable `exc` contient alors des détails concernant le problème (qu'est-ce qui a été appelé)
- Lancer l'exception : `throw exc;`

Choix du type d'exception

- Programmation défensive : interruption pour cause de précondition violée

`IllegalArgumentException` L'utilisateur vous fournit un argument incorrect (exemple : précondition *chaîne non vide* violée)

`IllegalStateException` L'utilisateur appelle votre sous-routine alors que l'état de votre programme ne le permet pas (exemple : précondition *appel préalable d'une autre sous-routine* violée)

- Programmation prudente : interruption pour cause d'erreur de déduction détectée (vérification violée)

`VerifyException` (Guava) Vous détectez une erreur de logique dans votre programme (exemple : une valeur devrait être paire à une certaine étape de calcul mais ne l'est pas)

À faire

- Découper en sous-routines
- Documenter vos contrats
- Échouer rapidement (avertir l'utilisateur de vos sous-routines) s'il y a un problème

Intérêt de la découpe en sous-routines

- Clarté du code : auto-documentation ; boite noire
- Factorisation : application conçue comme assemblage de blocs élémentaires
- Éviter la duplication de code (DRY)
- Bugs : correction à un seul endroit
- Partage du travail entre développeurs
- Estimation quantité de travail
- Réusinage facilité (trouver tous les endroits où routine est appelée)

Factorisation

- Code peut se ressembler sans être identique
- Modifier pour qu'il soit identique mais paramétré
- Exemple : Échecs, dessin du plateau vu du côté noir ou blanc

$\Rightarrow$ Une seule routine de dessin, paramétrée selon couleur

Deux sortes de throwable

- Rappel : Throwable en Java représente ce qui peut être lancé pour interrompre le flux normal de traitement
- À utiliser pour échouer rapidement
- Deux sortes de throwable
- Certains problèmes sont réparables (ne requièrent sans-doute pas de quitter le programme)
- Exemple ?
- Certains problèmes sont difficilement réparables
- Exemple ?

Deux sortes de throwable

- Rappel : Throwable en Java représente ce qui peut être lancé pour interrompre le flux normal de traitement
- À utiliser pour échouer rapidement
- Deux sortes de throwable
- Certains problèmes sont réparables (ne requièrent sans-doute pas de quitter le programme)
- Exemple ? Échec lors écriture sur fichier
- Certains problèmes sont difficilement réparables
- Exemple ? Erreur de logique ; Échec lors allocation mémoire

Licence