# Découpe et contrats en génie logiciel

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 19 janvier 2018

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 ?

# 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

- 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 ?

# Découpe en informatique

- En fait, que se passe-t-il lors du calcul d'une racine carrée ?
- Appel au processeur
- Lui-même un solveur de sous-problèmes
- Votre environnement d'exécution fait également appel au système d'exploitation
- Exemples de gros sous-problèmes déjà résolus pour vous ?

# Découpe en informatique

- En fait, que se passe-t-il lors du calcul d'une racine carrée ?
- Appel au processeur
- Lui-même un solveur de sous-problèmes
- Votre environnement d'exécution fait également appel au système d'exploitation
- Exemples de gros sous-problèmes déjà résolus pour vous ?
- Structures de données : listes, etc.

# Interface

- 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 ?

# Interface

- 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 ?

# Interface

- 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 simple, . . .
- Exemple radiateur ?

# Interface

- 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 simple, . . .
- Exemple radiateur ? Boutons différents, à différents endroits, commande vocale, . . .
- Exemple voiture ?

# Interface

- 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 simple, . . .
- 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 ?

# 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 ?

# 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
- Exemple : entier fourni en paramètre $> 0$
- Appelées *préconditions*
- *Contrat* entre appelant et programmeur de la sous-routine
- Sous ces conditions, méthode fournit un service
- Exemple : renvoie un nombre aléatoire entre 0 et l'entier fourni, exclu
- Si conditions non remplies : pas de garanties offertes !

# 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
- 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 (cf. exemple)
- Évite les conséquences catastrophiques
- Évite que l'erreur passe inaperçue
- 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
- Si précondition non satisfaite : le faire savoir
- 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 si non valide
- Exemple : je sais qu'ici telle valeur devrait être positive

# Sous-routines

- Sous-routine : entrée (facultative), ensemble d'instructions, sortie (facultative)
- Fait qqch
- On peut l'appeler sans savoir comment elle procède (boite noire)
- `Math.random();` `Math.sqrt(4);`
- Sous-routine sqrt statique se trouvant dans classe `Math`
- Classes regroupent des variables et sous-routines statiques
- `System.out` : variable statique `out` dans classe `System`
- Permet d'*organiser* et de *nommer*

# Intérêt

Découper 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)

# À faire

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

# 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

# Illustration

- Calculateur qui additionne
- Deux interfaces pour un même service (plusieurs étapes ou une étape)
- Deux implémentations pour l'interface en plusieurs étapes
- Et si addition seulement de nombres positifs ?

# Licence