Automatisation et Tests en Programmation

Séance 2 de cours/TD

IUT SD Aurillac

Objectifs:

— Etude des tests fonctionnels.

Tests fonctionnels [Cours]

Au contraire des tests structurels que nous avons vu la dernière fois, les tests fonctionnels se font sans la connaissance du code, juste de sa spécification. L'intérêt est de pouvoir développer les tests en parallèle du développement du code, voire de comparer plusieurs implémentations pour vérifier lesquelles correspondent le mieux au comportement attendu.
La question devient alors comment analyser la spécification pour produire des jeux de test pertinent. Nous allons voir quelques méthodes qui proposent des stratégies pour cela. Ces méthodes sont souvent utilisées ensemble .

Analyse Partitionnelle [Cours]

L'hypothèse de départ de cette méthode est que deux valeurs d'entrées similaires auront un effet similaire sur le programme. Il convient donc de partitionner (découper) l'ensemble des valeurs d'entrées possibles en sous-classes telles que l'ensemble des valeurs d'une sous-classe aient des propriétés similaires. On procède en trois phases :

1. Pour chaque paramètre en entrée, on calcule les sous-classes de l'ensemble des possibles.
2. Pour chacune des sous-classes, on choisit un représentant.
3. On effectue ensuite un produit cartésien entre les ensembles de représentants de chaque paramètre pour obtenir le jeu de test.

La question devient alors comment construire les sous-classes. Il n'y a pas de solution parfaites, mais voici quelques règles :

— Si le paramètre appartient à un intervale $[a, b]$. On découpe l'intervalle en n classes de même taille (avec n choisi en fonction du niveau de détail voulu) auxquelles on ajoute une classe pour les valeurs inférieures à a et une classe pour les valeurs supérieures à b.
— Si le paramètre est une liste de valeurs. On construit n classes valides, auxquelles on ajoute une classe pour la liste vide, et une classe pour les listes contenant trop de valeurs.
— Si le paramètre doit vérifier une certaine propriété. On construit une classe de valeurs vérifiant la propriété, et une classe de valeurs ne la vérifiant pas.

Cette méthode est efficace car elle permet de couvrir beaucoup de cas si les sous-classes sont bien construites. Son principal inconvénient tient dans l'explosion combinatoire quand le programme gère plusieurs paramètres. Par exemple, un programme à 4 paramètres ayant chacun 5 classes possibles de valeurs produira $5^4 = 625$ jeux de test.

La méthode suivante permettra de contourner ce problème.

Exercice 1 (Triangles, $\star$)
Spécification : Le programme prend en entrée trois réels, interprétés comme étant les longueurs des côtés d'un triangle. Si ces longueurs forment un triangle, le programme retourne la propriété du triangle correspondant (scalène, isocèle ou équilatéral) ainsi que la propriété de son plus grand angle (aigu, droit ou obtus).

Donner les classes d'équivalence sur les entrées de ce programme, ainsi qu'un cas de test pour chacune des classes. □

Méthode Pairwise [Cours]

Le point de départ de cette méthode est le fait que l'augmentation du nombre de paramètres d'un programme rend de plus en plus compliquée l'approche systématique de l'analyse partitionnelle. La proposition est donc de ne pas générer de tests pour chacune des combinaison possibles, mais uniquement le nombre nécessaire de test permettant de tester toutes les paires possibles de paramètres.

Prenons un exemple : Soit trois variables x, y, z prenant les valeurs suivantes $x \in \{1, 2\}, y \in \{Q, R\}, z \in \{5, 6\}$. Il existe 8 combinaisons possibles de ces trois paramètres, et 12 paires. On peut toutefois couvrir ces 12 paires avec seulement quatre tests : $(1, Q, 5), (1, R, 6), (2, Q, 6), (2, R, 5)$.

Il n'est pas toujours aisé de trouver les tests permettant de couvrir toutes les paires, mais il existe des outils en ligne permettant de générer ces jeux.

Exercice 2 (, $\star$)
On veut tester un script d'impression sur plusieurs OS, imprimantes, applications et réseaux. Voici les choix possibles pour chacun de ces paramètres.

OS	Réseau	Imprimante	Application
Windows7	IP	HP35	Word
Linux	WIFI	Canon900	Excel
Mac OS X	Bluetooth	Canon-EX	Powerpoint

Combien y-a-t-il de combinaisons possibles ? Proposez une série de jeux de test en vous basant sur la méthode pairwise.

□

Test aux limites [Cours]

Le point de départ de cette méthode est le constat que la plupart des erreurs dans un programme sont dûs à l'oubli dans la gestion des cas limites telles que les valeurs très grandes ou petites, les variables de boucles nulles ou négatives, voire les données non valides.
La proposition est alors de générer des tests pour tester les comportements du programmes près des limites des paramètres.

Par exemple, si un paramètre doit appartenir à l'intervalle $[a, b]$, on teste $a-1, a, a+1, b-1, b, b+1$.

Exercice 3 (, $\star$)
Spécification : Ecrire un programme statistique analysant un fichier comprenant les noms et les notes des étudiants d'une année universitaire. Ce fichier se compose au maximum de 100 entrées. Chaque entrée comprend le nom de chaque étudiant (20 caractères), son genre (1 caractère) et ses notes dans 4 UE (décimaux compris entre 0 et 20). Le but du programme est de :

- *calculer la moyenne pour chaque étudiant*

— *calculer la moyenne générale (par genre et par UE)*
— *calculer le nombre d'étudiants validant leur année.*

En appliquant les tests aux limites, donnez une séries de jeux de test pour cette spécification. □