Première étude de cas

« Réduire simplement la combinatoire »

On souhaite mettre en place un nouveau serveur de données multimédia au niveau de la DSI. Les données sont de quatre types distincts : html, avi, wave et mp3. Les clients de ce serveur seront des navigateurs WEB de type Intenet Explorer 5 (IE5), Netscape 7 (Ne7) ou FireFox (FFx). Les systèmes clients peuvent être des systèmes Linux-Suse 9.1 (LS), Windows 2000 (W2000), Windows Me (WMe), Mac OSX (Mac) ou Linux-RedHat-Fedora 7 (LR). Avant de déployer le serveur vous êtes chargé de vérifier rapidement qu'il y a une bonne prise en compte des différents formats dans les différents cas d'utilisations potentiels.

Question 1) Si l'on souhaite être exhaustif, combien de configurations différentes faut-il tester ? Si chaque test demande une préparation de 1/2 heures, quel temps mettrez-vous ?

Réponse :

Une première solution pour réduire cette combinatoire est d'adopter la stratégie dite « All-Singles ». Celle-ci consiste à construire des jeux de tests couvrant au moins chaque possibilité pour chaque donnée ; ici, il faudra que les jeux de tests couvrent au moins une fois chacun des OS, chacun des navigateurs et chacun des types de fichiers multimédia. Par exemple, les deux configurations (WMe, Ne7, avi) et (W2000, FFx, mp3) couvrent deux types de systèmes, deux types de navigateurs et deux types de fichiers.

Question 2) Donner les configurations de tests permettant de couvrir selon la stratégie All-Singles.

Réponse :

Question 3) Quel est le principal inconvénient de cette stratégie ?

Réponse :

Afin d'améliorer cette stratégie on recourt souvent à la stratégie dite « All Pairs » (qui se décline aussi en « All triple » et ainsi de suite). Cette stratégie vise à construire des jeux de tests de telle sorte que toutes les paires de possibilités soient couvertes au moins par un jeu de test. Par exemple, il faudra que tous les couples (système / navigateurs), (système / type de donnée) et (navigateur / type de donnée) apparaissent au moins dans un jeu de test.

Différentes approches sont possibles pour construire ces jeux de tests. Nous en étudions une fort simple. Pour cela, on commence par ordonner en ordre décroissant les variables par rapport à leur nombre de valeurs possibles. Notons V1 et V2 le nombre respectif de valeurs possibles pour les deux premières variables.

Ici cela donne Système (5 valeurs), Données (4 valeurs), Navigateur (3 valeurs).

On construit alors une table avec une colonne par variable et au moins V1*V2 lignes (ici cela donne 5*4=20 lignes).

Sur les V2 premières lignes, on met la première valeur de la première variable, puis une ligne blanche, puis la seconde valeur de la première variable sur V2 lignes, une ligne blanche et ainsi de suite. On a ainsi rempli la première colonne.

Pour la seconde colonne, on alterne successivement les V2 valeurs de la seconde variable (en passant la ligne blanche). On a ainsi par ligne toutes les paires des 2 premières variables.

Pour la troisième colonne, on alterne les différentes valeurs de la troisième variable en faisant en sorte de construire des paires avec la colonne 2.

On procède ainsi avec les autres colonnes. Chaque ligne définit alors un jeu de test.

Question 4) Appliquer cette méthode sur l'exemple initial en complétant la table suivante. Combien de jeux de test seront générés par cette stratégie ? Commenter.

Système	**Type Données**	**Navigateur**	
Wme	Html	IE5	
Wme	Avi	Ne7	
Wme	Wave	FFx	
Wme	Mp3	IE5	
W2000	Html	Ne7	
W2000	Avi	FFx	
W2000	Wave	IE5	
W2000	Mp3	Ne7	
LS	Html		
LS	Avi		
LS	Wave		
LS	Mp3		
Mac			
Mac			
Mac			
Mac			
LR			
LR			
LR			
LR			

Question 5) Ajouter une variable binaire (par exemple gestion https OUI/NON). Cela modifie t-il le nombre de tests à générer ? Que pouvez-vous conclure de cette stratégie ?

Réponse :