UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

tesi di laurea

Testing black box di web service: sperimentazione su di un servizio senza stato

Anno Accademico 2005/2006

relatore
Ch.mo prof. Porfirio Tramontana

candidato
Alfredo Monaco
Matr. 534/001519

Obiettivi

Risulta importante poter valutare se un web service abbia o non abbia uno stato, per poterne dedurre il comportamento e poter scegliere tra le varie alternative di servizi offerti quello che meglio soddisfa le nostre esigenze.
Si vuole vedere se è possibile risolvere questo problema tramite lo studio delle classi di equivalenza (CE) e l'analisi delle tracce di esecuzione, ottenute dopo aver effettuato un testing black box su di un servizio senza stato.

Web services e loro importanza

- *"Un servizio Web è un sistema software identificato da una URI le cui interfacce pubbliche e i binding sono descritti da XML. La sua definizione può essere rintracciata da altri sistemi software. Questi sistemi software possono interagire secondo quanto scritto nella definizione, usando messaggi XML veicolati da protocolli internet."* (W3C)
- Consentono accesso a software in rete
- Pay per use
- Transazioni automatiche economiche

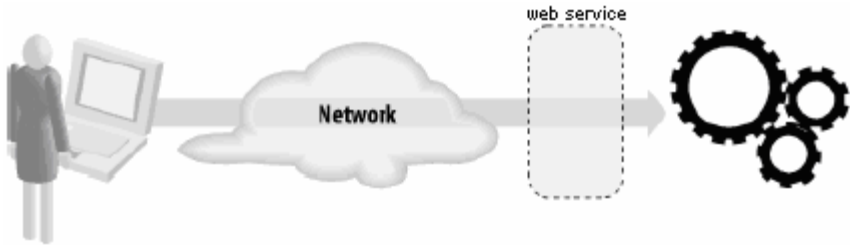
Strumenti utilizzati

•Tomcat - funge da web server

•Axis - motore SOAP

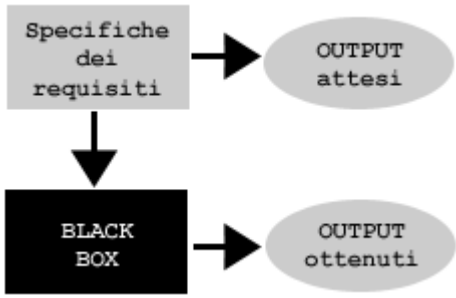
•JWSDP - tecnologie di supporto

•WSDD - deployment del servizio



Testing black box di web services

- Le tecniche di testing black box non richiedono la conoscenza dei dettagli di implementazione del servizio da testare ma progettano test cases sulla base di funzionalità specifiche o output attesi.
- I testers devono avere una discreta conoscenza dei meccanismi di programmazione perche i web service sono UI-less**.**
- La ***suddivisione in classi di equivalenza*** è un metodo di collaudo black-box che suddivide il dominio dei dati di un servizio in classi di dati, dalle quali derivare casi di prova.



Esperimento

E'stato considerato un servizio del quale si conoscono bene le specifiche.

Da esse sono state ricavate le classi di equivalenza, grazie alle quali è stato possibile derivare casi di test significativi.

Infine analizzando le tracce di esecuzione si sono formulate le opportune deduzioni…

Contesto

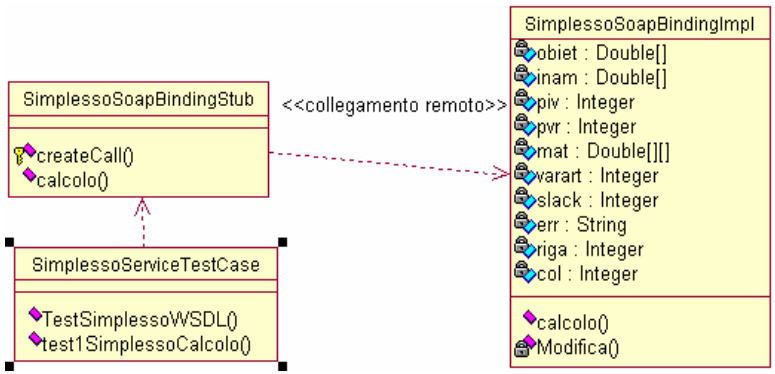
Come tecnica di collaudo è stata scelta la suddivisione in classi di equivalenza.

Per effettuare un testing black box ***corretto***, è necessario sapere se il servizio è con o senza stato perchè il suo comportamento potrebbe dipendere in maniera significativa dallo stato dei dati gestiti e dall' input fornito dall'utente.

Nel nostro caso è stato considerato il web service Simplesso, senza stato.

Web service Simplesso

- Risolve problemi di programmazione lineare usando l'algoritmo del simplesso
- Input: numero vincoli, numero variabili (max 10), matrice dei vincoli, funzione obiettivo
- Output: numero righe, numero colonne, matrice ottima, valore ottimo della funzione obiettivo
- 17 CE di input
- 3 CE di output
- Se il problema è inconsistente restituisce una matrice vuota



- Se ci sono errori nel passaggio dei dati di input restituisce una matrice di zeri

Classi di equivalenza

Le classi di equivalenza (CE) sono state ricavate a partire da un attento studio delle specifiche del problema e da una buona conoscenza della teoria del simplesso.

Partendo dalle CE ricavate sono stati creati ed eseguiti gli opportuni test (in ognuno dei quali sono indicate le CE di input e di output coperte).

Le tracce di esecuzione sono state raccolte per poterne effettuare un'attenta analisi. Siccome Simplesso è stateless ci si aspetta che dati appartenenti alle stesse CE restituiscano sempre output appartenenti alla stessa CE.

Esempio di CE:

Input = n° variabili

Valida $\in$ [1,2,3,4,5,6,7,8,9,10]

Non valida $\notin$ [1,2,3,4,5,6,7,8,9,10]

Output = "Errore nei valori di input"

Esempio di esecuzione e test

In questi due test i dati di input sono coperti dalle medesime classi (C1, C2, C5, C7, C11, C13) e si nota che in output entrambi hanno dei dati coperti dalla classe C1.

Proprio come ci si aspettava…☺

Ma…

TEST 2	
INPUT	n. vincoli :2 n. variabili :3 Matrice vincoli : 252.4273087685 128.6646616867 1 433.712657041 42.5239780281 133.4229441959 1 684.639055578 Funzione obiettivo : 890.0308912205325 352.62638840914906 max
Classi Equivalenza Input coperte	c2 c1 c13 c11 c5 c7
OUTPUT	n. righe :3 n. colonne :5 Matrice vincoli : 1.0 0.509709754916 0.0039615365107 0.0 1.718168526050 0.0 111.7480577771 -0.1684602915414 1.0 611.575694928 Funzione obiettivo : 0.0 -101.0310390232 -3.52588987127 0.0 -1529.22306450
Classi Equivalenza Output coperte	c1

TEST 5	
INPUT	n. vincoli :2 n. variabili :3 Matrice vincoli : 723.402721618537 277.94706712369464 1 656.7513804405293 699.4931403430138 12.748555168029085 1 138.4119745127932 Funzione obiettivo : 742.9486230705647 193.3171721777014 max
Classi Equivalenza Input coperte	c2 c1 c13 c11 c5 c7
OUTPUT	n. righe :3 n. colonne :5 Matrice vincoli : 0.0 1.0 0.00377696636189 -0.00390606796274 1.939881292745 1.0 0.0 -6.8836792321905 0.001500796308593 0.162519522031 Funzione obiettivo : 0.0 0.0 -0.6790102564 -0.359904538086 -495.75602098939
Classi Equivalenza Output coperte	c1

Test e deduzioni

Qui si vede che le stesse CE di input danno due CE di output differenti (C1 e C2). Conoscendo la programmazione lineare si sa che un problema inconsistente (coperto dalla classe C2) dipende dal dominio (e quindi dai vincoli). Nel test 12 infatti si vede che i vincoli hanno tutti e 3 l'operatore "="; ciò fa sì che le CE di input possano essere uguali a quelle del test 2, ma restituiscano, tuttavia, un problema inconsistente. ☹

TEST 2	
INPUT	n. vincoli :2 n. variabili :3 Matrice vincoli : 252.4273087685 128.6646616867 1 433.712657041 42.5239780281 133.4229441959 1 684.639055578 Funzione obiettivo : 890.0308912205325 352.62638840914906 max
Classi Equivalenza Input coperte	c2 c1 c13 c11 c5 c7
OUTPUT	n. righe :3 n. colonne :5 Matrice vincoli : 1.0 0.509709754916 0.0039615365107 0.0 1.718168526050 0.0 111.7480577771 -0.1684602915414 1.0 611.575694928 Funzione obiettivo : 0.0 -101.0310390232 -3.52588987127 0.0 -1529.22306450
Classi Equivalenza Output coperte	c1

TEST 12	
INPUT	n. vincoli :3 n. variabili :3 Matrice vincoli : 632.8711415422677 600.9073817168364 0 930.9919301980375 238.24662662766792 965.8799527954845 0 683.4971934842021 93.15346387440925 501.8836169790046 0 324.42704253863076 Funzione obiettivo : 789.53221397567 287.4357915018522 max
Classi Equivalenza Input coperte	c2 c1 c13 c11 c5 c7
OUTPUT	n. righe :0 n. colonne :0 Problema inconsistente
Classi Equivalenza Output coperte	c2

Conclusioni

E'stato testato, tramite la suddivisione in classi di equivalenza, il web service stateless Simplesso. Grazie alla conoscenza delle sue specifiche si può affermare che è deterministico.
Osservando i risultati, però, ci siamo resi conto che alcuni dati di input appartenenti alle stesse classi di equivalenza hanno dato in output dati appartenenti a classi di equivalenza diverse. Da ciò, dunque, dovremmo dedurre che Simplesso è un servizio stateful ☹
Quindi con questo studio non siamo in grado di dedurre che Simplesso è un web service stateless.