TESTAREA FLUXULUI DE DATE

Autor: Mihaela LOZOVANU
Conducător ştiinţific: magistru în TI, lect. sup. Mariana CATRUC
Universitatea Tehnică a Moldovei
Email: mihaela.lozovanu@gmail.com, micatruc@yahoo.com

Abstract: *Testarea fluxului de date expune erori de software prin trasarea de date prin intermediul fluxului de control al programului de la definirea variabilelor la utilizarea lor. Testarea fluxului de date este o tehnica de testare a fluxului de control, care examinează, de asemenea, ciclul de viaţă al variabilelor de date. Utilizarea testării fluxului de date duce la o suită de teste mai bogată concentrându-se pe folosirea neadecvată a datelor din cauza erorilor de codificare. Scopul principal al acestei lucrări este de a discuta despre conceptul de testare a fluxului de date şi beneficiile sale.*

Cuvinte cheie: *flux de date, black box, white box, strategii de testare, anomalii, diagrama fluxului de control.*

1. Introducere

Testarea software este procesul de analiză a unui element software pentru a detecta diferenţele dintre condiţiile existente şi necesare (care sunt, bug-uri), precum şi pentru a evalua caracteristicile elementelor software-ului testat. Principalele obiective ale testării software sunt descoperirea erorilor şi asigurarea că sistemul în curs de elaborare este în conformitate cu cerinţele clientului. Pentru a face testarea eficientă, este recomandat ca testul de planificare / dezvoltare să înceapă la debutul proiectului. Tehnicile de testare software pot fi împărţite în 2 grupuri: testarea funcţională (black box) şi testarea structurală(white box). Testarea funcţională este în principiu o tehnică de validare care controlează dacă produsul întruneşte cerinţele clientului. Testarea structurală este o tehnică de verificare care utilizează codul sursă pentru a ghida o selecţie de date de test.

Testarea fluxului de date este o tehnică de testare structurală care poate fi folosită pentru a detecta utilizarea improprie a valorilor de date ca urmare a erorilor de codificare. Erorile sunt introduse din greşeală în program de către programatori. De exemplu, un programator ar putea folosi o variabilă fără a o defini. În plus, se poate defini o variabilă, fără a o iniţializa şi apoi se poate folosi ca o variabilă într-un predicat.

Testarea fluxului de date construieşte şi extinde tehnicile testării fluxului de control. Ca atare testarea fluxului de control ar trebui să fie folosită pentru toate modulele de cod care nu pot fi testate suficient prin revizii şi inspecţii. Limitele sale se referă la faptul că testerul trebuie să aibă suficiente abilităţi de programare care să înţeleagă codul, fluxurile sale de control şi variabilele sale. Asemenea testării fluxului de control, testarea fluxului de date poate fi uneori consumatoare din cauza tuturor modulelor, a căilor şi a variabilelor care compromit sistemul.

La testarea fluxului de date, primul pas este de a modela programul ca un graf al fluxului de control. Acest lucru ajută la identificarea fluxului de control al informaţiei în program. În pasul 2 sunt stabilite asociaţiile între definiţiile şi utilizările variabilelor care sunt necesare pentru a fi acoperite cu un anumit criteriu de acoperire. În pasul 3, suita de test este creată folosind un număr finit de căi de la pasul 2.

Testarea fluxului de date monitorizează ciclul de viaţă al unei piese de date şi se îngrijeşte de utilizarea necorespunzătoare a datelor în timpul definiţiei, utilizarea în predicate, calcule şi terminare (uciderea). Acesta identifică bug-uri potenţiale examinând modelele în care acea bucată de date este utilizată.

De exemplu, un model care indică utilizarea unei variabile într-un calcul după ce a fost ştearsă, cu siguranţă este un bug care trebuie să fie abordat.

Pentru a examina modelele, avem nevoie de a construi un graf al fluxului de control a codului. Un graf al fluxului de control este un graf orientat în care nodurile reprezintă prelucrarea declaraţiilor de definire, de calcul şi predicate, în timp ce vârfurile reprezintă fluxul de control între declaraţiile de prelucrare. Deoarece testarea fluxului de date îndeaproape examinează starea datelor în graful fluxului de control, el rezultă într-o set de teste mai bogată decât cel obţinut de la tradiţionalul graf al fluxului de control.

2. Anomaliile fluxului de date

Anomaliile fluxului de date reprezintă modele de utilizare a datelor care pot duce la o executare incorectă a codului. Notația pentru modele este:

d- definit, creat, inițializat
k- distrus, realizat, nedefinit
u- utilizat
c- utilizat în calcule
p- utilizat în predicate
~ x- indică faptul că toate acțiunile anterioare nu prezintă interes pentru x
x ~- indică faptul că toate acțiunile ulterioare nu prezintă interes pentru x.

Utilizarea variabilelor sau a elementelor de date se numeste U-utilizarea în dependență de utilizarea lor în predicate sau calcule. Utilizarea variabilelor în predicate senumeşte P-utilizare, iar cele utilizate în calculi matematice se numeşte C-utilizare. Pentru utilizarea acestor modele trebuie de menționat că acestea trebuie definite mai devreme, astfel încât să se poată determina tipurile lor, să se poată obține valorile lor, şi să fie utilizate în scopuri diferite. Formal, putem defini aceste modele, după cum urmează:

- definirea de date prin crearea datelor, inițializarea, atribuirea, toate în mod explicit, sau uneori prin reacții adverse, cum ar fi prin locații de memorie partajată, căsuțe poştale, parametri read/write, etc Acesta este de obicei abreviat ca D-operație, sau doar D. Cheia caracteristică a operațiunii D este faptul că este distructivă, ceea ce înseamnă că orice ar fi fost stocat în elementul de date este distrus după această operațiune şi nu poat fi recuperate cu excepția cazului în care este utilizsat un mecanism de recuperare de specialitate
- Utilizarea de date în calcul general sau în predicat, denumit în mod obişnuit ca C-utilizare sau P-utilizare. Ambele aceste tipuri de utilizări sunt numite colectiv U-operație, sau doar U. Cheia caracteristică a operațiunii U este faptul că ea este non-distructivă, ceea ce înseamnă că valoarea elementului de date rămâne acelaşi după această operație. Cu toate acestea, P-utilizare a unui element de date într-un predicat ar putea afecta calea de execuție care urmează să fie selectată şi urmată. C-utilizare a elementelor de date apare de obicei sub forma de variabile şi constant într-o expresie de calcul sau ca parametri într-o funcție sau o procedură de program. Astfel de C-utilizare de obicei afectează unele rezultate de calcul, cu variabile-rezulate fiind definite anterior.

Tabelul 1 enumeră combinațiile de utilizare a acestora şi consecințele corespunzătoare. Se poate observa că nu toate anomaliile a fluxului de date sunt dăunătoare, dar ele toate sunt suspecte şi indică faptul că poate apărea o eroare. De exemplu, modelul de utilizare "ku" indică faptul că o variabilă este utilizată după ce a fost ucisă, ce este un defect serios.

Tabelul 1

Anomalii		Explicații
~d	Mai întâi definiți	Permis
du	Definiți-utilizați	Permis. Caz normal.
dk	Defineşte- distruge	Bug potențial. Variabila este distrusă fără a fi folosită după definire.
~u	Mai întâi utilizează	Bug potențial. Variabila este utilizată fără a fi definită.
ud	Foloseşte - defineşte	Permis. Variabila este folosită şi apoi redefinită.
uk	Foloseşte - distruge	Permis
~k	Mai întâi distrusă	Bug potențial. Variabila este distrusă înainte de a fi definită.
ku	Distruge - foloseşte	Defect serios. Variabila este folosită după ce a fost ştearsă.
kd	Distruge - defineşte	Permis. Variabila este distrusă apoi redefinită.
dd	Defineşte -defineşte	Bug potențial. Definire dublă.
uu	Foloseşte -foloseşte	Permis. Caz normal
kk	Distruge - distruge	Bug potențial
d~	Defineşte la sfârşit	Bug potențial
u~	Foloseşte la sfârşit	Permis
k~	Distruge la sfârşit	Permis. Caz normal

2.1 Testarea fluxului de date static

Cu analiza statică, codul sursă este analizat fără a fi executat. Testarea fluxului de date static nu este eficientă în situații în care starea unei variabile de date nu poate fi determinată doar analizând codul. Acest lucru este posibil atunci când variabilele de date sunt utilizate ca un index pentru o colecție de elemente de date. De exemplu, în cazul matricelor, indicii ar putea fi generați dinamic în timpul execuției, prin urmare, noi nu putem garanta ce statut are elementul de matrice care este menționat de indicele dat. Mai mult decât atât, testarea fluxului de date poate să atribuie unei anumite bucăți de cod o anumită anomalie, iar această bucată poate să nu fie executată niciodată, deci prin urmare nu e completă.

Analiza statica deasemenea nu poate fi folosită în cadrul sistemelor în care procesele sunt întrerupte. Deoarece unele acțiuni din aceste sisteme definește – utilizează – distruge s-ar putea produce la nivelul întrerupt, în timp ce altele ar putea apărea la nivelul de procesare principal. În plus, dacă sistemul foloseşte nivele multiple de priorități de executare, analiza statică a nenumăratelor interacțiuni posibile este, pur şi simplu, prea dificilă pentru a putea fi executată manual.

Din acest motiv revenim acuma la testarea fluxurilor de date dinamic.

2.1 Testarea fluxului de date dinamic

Scopul principal al testării fluxului de date dinamic este de a descoperi posibile bug-uri în utilizarea datelor în timpul executării codului. Pentru a realiza acest lucru, cazurile de testare sunt create pentru a urmări fiecare definire pentru fiecare utilizare şi fiecare utilizare este urmărită pentru fiecare definire a sa. Diferite strategii sunt implementate pentru crearea cazurilor de testare. Strategiile reprezintă o familie de tehnici de testare bazate pe selectarea cu atenție a unei mulțimi de căi din program. Dacă mulțimea căilor este aleasă corespunzător, atunci se va atinge o anumită măsură a profunzimii testului. Pentru utilizarea acestor tehnici este necesară cunoaşterea completă a structurii programului şi accesul la codul sursă. Fiecare definiție a tuturor strategiilor este explicată de exemplele următoare.

Toate căile ***du*** – este cea mai puternică strategie de testare a fluxului de date. Aceasta constă din toate căile de la fiecare definire a fiecărei variabile la orice utilizare a definirii date.

Toate căile ***u*** – reprezintă cel puțin o cale de la fiecare definire a fiecărei variabile la fiecare utilizare care poate fi atinsă de acea definire.

Toate căile ***pu*** – pentru fiecare variabilă, este o cale de la fiecare definire la fiecare pu a definiri date.

Toate căile ***cu*** – pentru fiecare variabilă, există o cale de la fiecare definire la fiecare ***cu*** a acestei definiri.

Toate căile ***pu***/câteva căi ***cu*** – pentru fiecare variabilă şi fiecare definire a acestei variabile, este inclusă cel puțin o cale de la definire la fiecare predicat utilizat; dacă sunt definiri de variabile care nu există atunci se adaugă cazuri de test computaționale.

Toate căile ***cu***/câteva căi ***pu*** – pentru fiecare variabilă şi fiecare definire a acestei variabile, există cel puțin o cale de la definire la fiecare caz computațional, iar dacă nu sunt se adaugă cazuri de test utilizând predicatele.

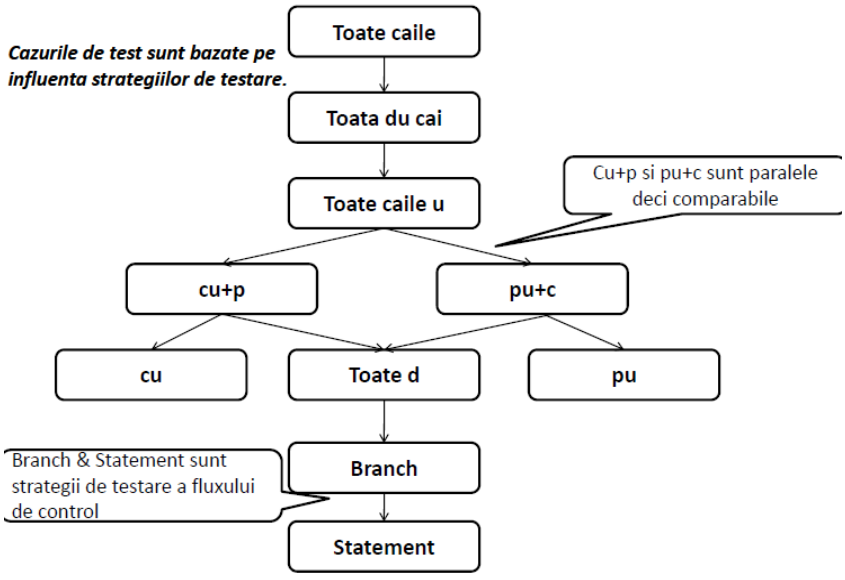

Figura 1 reprezintă influența strategiilor de testare. Potrivit acestei figuri, influența strategiilor de testare se reduce odată cu parcurgerea ei în jos. Prin urmare, strategia tuturor căilor este cea mai puternică(influentă). De asemenea, observăm că căile cu+p şi pu+c se desfăşoară în paralel prin urmare, ele sunt comparabile. Strategiile de testare sunt aplicate pentru alcătuirea grafelor adnotate a fluxului de control pentru fiecare variablă, iar cazurile de test sunt derivate ulterior.

3. Concluzie

Am prezentat un studiu asupra testării fluxului de date concentrându-ne asupra criteriului de testare a fluxului de date şi anomaliile ce pot surveni. Varietatea strategiilor de testare pot dezvălui eficient problemele în codul programului, ajutând testerul să evalueze codul uniform, concentrându-se pe locurile în care pot apărea probleme – anume acele care implică variabile de atribuire şi utilizare.

Selectarea strategiilor fluxului de date pentru un anumit cod sursă a unui program se face în dependență de complexitate şi de timpul disponibil pentru testare. Testarea fluxului de date static este puternică, însă dacă codul programului are mai multe cicluri, aceasta ar pute fi neeficient şi neproductiv. Testarea fluxului de date dinamic s-a dovedit a fi un compromis eficient, şi utilizat în analiza valorilor şi echivalența claselor ierarhice. Acest tip de testare poate oferi o acoperire excelentă a codului.

Bibliografie

1. B. Beizer, *Software Testing Techniques*, New York, second edition, 1990.
2. L. Copeland, *A Practitioner's Guide to Software Test Design*, Artech House Publishers, Boston, 2003.
3. E. J. Weyuker, More experience with data flow testing, Software Eng., 1993.
4. J. Tian, *Software Quality Engineering*, IEE Computer Science, 2005.
5. L. Copeland, *A Practitioner's Guide to Software Test Design*, STQE Publishing, 2004.