Testarea Orientată Obiect

Generaţia a patra de software este rezultatul analizei, proiectării şi programării orientate obiect. Tehnologiile orientate obiect generează particularităţi care se reflectă în toate etapele ciclului de viaţă software, inclusiv în testare. Pornind de la testarea empirică, până la utilizarea unor tehnici şi metode moderne, viziunea orientată obiect generează noi abordări. Testarea se face atât pentru un produs software luat ca un întreg, cât şi pentru componentele acestuia. Dacă se testează sistemul de programe procesul se desfăşoară da la simplu spre complex. Se testează mai întâi modulele apoi programul şi în final sistemul de programe ca ansamblu complex. Rezultatele asamblării, software integrat, este testat în continuare (integration testing).

Programarea orientată obiect pesupune definire de clase şi referire de obiecte.

Clasele sunt definite, caz în care reutilizarea generează efecte pozitive. Când clasele sunt subdefinite este necesară construirea de clase derivate de completare. Când clasele sunt supradefinite apar restricţii de referire şi de reutilizare. Testarea claselor este permisă reutilizării necondiţionate. Testarea claselor evidenţiază raportul operanzi–operatori, gradul de moştenire şi de acoperire probabilă a tipologiilor de clase de probleme prin constructori/destructori(în funcție de limbaj) definiţi. Programarea orientată obiect este caracterizată printr-un nivel foarte ridicat al reutilizării software.

Numeroase aplicaţii complexe se prezintă sub forma unor programe apelatoare (**main**) care conţin lungi secvenţe de directive şi ca instrucţiuni, structuri liniare de expresii cu referire la funcţiile(metodele) membre ale obiectelor folosite.

Testarea software orientat obiect presupune două planuri:

- testarea construcţiilor proprii;
- testarea construcţiilor incluse pentru reutilizare.

Testarea software-ului orientat obiect are pe lângă obiectivul general al stabilirii măsurii în care produsul software realizează sarcinile date în specificaţii, obiective specifice legate de:

- testarea funcţiilor(metodelor) membre ale fiecărei clase;
- testarea gradului de încapsulare şi a efectelor acestuia;
- testarea efectelor induse de nivelele de moştenire şi de derivare;
- testarea efectelor induse de polimorfismul funcţiilor membre.
- testarea interacţiunilor dintre clase.

Spre deosebire de software-ul dezvoltat prin alte metode, în cazul programării orientate obiect, testarea vizează şi măsura în care clasele sunt proiectate în vederea reutilizării. Adică, se evidenţiază gradul de generalitate şi mai ales concordanţa dintre specificaţiile fiecărei metode(funcţii) şi ceea ce efectiv metoda(funcţia) realizează.

Rezultatul testării vizează două aspecte şi anume:

- secvenţa referirilor determină pentru exemplele de test rezultate acceptabile sau nu ceea ce se răsfrânge asupra produsului ca atare;
- rezultate privind măsura în care clasele definite sau referite acoperă cerinţele unei reutilizări confortabile, fără nevoia de a construi interfeţe sau de a realiza derivări(extinderi) în obţinerea de clase noi cu un nivel de saturare redus, create în mod special şi destinate unor utilizări cu totul particulare.

Dacă produsul poate fi acceptat pentru calităţile lui în raport cu problema de rezolvat, nu este obligatorie şi acceptarea claselor definite. În acelaşi fel, clasele definite pot îndeplini condiţiile de reutilizare, fără ca programul să fie acceptat în urma testării.

Metoda de testare ierarhică stă la baza sistemului de testare orientat obiect. Această metodă de testare utilizează şi construieşte pe baza unor tehnici de testare cunoscute, legându-le împreună într-un sistem de testare corespunzător. Metoda defineşte şi aplică standarde de testare pentru câteva nivele ale componentelor software: obiecte, clase, componente de bază şi sisteme.

Metoda de testare ierarhică indică ca fiind "sigure" acele componente care îndeplinesc standardele de testare pentru tipul respectiv de componentă. Odată ce o componentă a fost desemnată ca "sigură", ea poate fi integrată cu alte componente "sigure" pentru a realiza împreună componentele de pe nivelul următor.

"Sigur" este o stare relativă. Depinde în întregime de:

- standardele alese pentru realizarea aplicaţiei;
- atitudinea faţă de risc;
- riscurile specifice şi practicile de management al riscului adoptate în proiect.

Metoda de testare ierarhică se axează pe componentele de bază. O componentă de bază poate fi o ierarhie completă de clase sau anumite clustere de clase care realizează o funcţie de bază sau care reprezintă o componentă arhitecturală logică sau fizică.

După ce este testată o componentă de bază la un nivel "sigur", ea poate fi integrată cu alte componente de bază. Testarea de integrare a componentelor de bază "sigure" necesită accesarea doar a interconexiunilor dintre componentele de bază şi orice funcţionalitate complexă nouă. Metoda ierarhică elimină obligativitatea testării tuturor combinaţiilor de stări în timpul testării de integrare, ceea ce duce la creşterea productivitaţii.

În figura 1 este indicat un model de reprezentare grafică a metodei ierarhice.

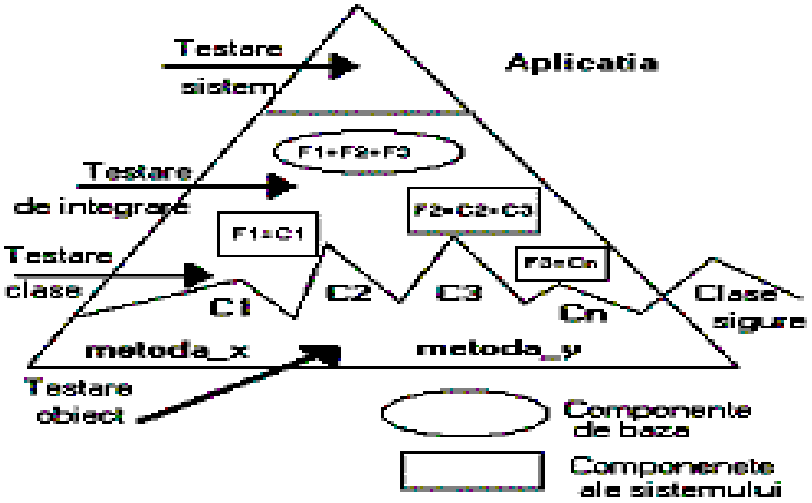

Fig. 1. Metoda ierarhică de testare

La baza piramidei sunt testate complet metodele individuale pentru a fi denumite "sigure". Metodele se integrează în clase şi se va testa întreaga ierahie de clase pentru a se atinge un nivel "sigur". Colecţia de clase "sigure" (componentele de bază) formează baza produsului final. Aria largă de integrare a piramidei este locul în care se integrează componentele de bază in diferite structuri arhitecturale, funcţionale şi logice. Ele formează un sistem.

Vârful piramidei este un test de sistem scurt şi concentrat. Următoarea suită de teste completează structura piramidei:

- **suita de teste condiţionale** testează clasele utilizând modelul de testare condiţională şi aserţiunile, excepţiile, operaţiile de testare concurente adiţionale;
- **suita de teste ierarhice incrementale** testează componentele de bază utilizând diverse modele de testare şi scenarii;
- **suita de teste de integrare** testează combinaţii de componente de bază utilizând modelul ierarhic incremental cu scenarii care utilizează toate componentele, nu doar stub-uri;
- **suita de teste de sistem** testează sistemele componentelor de bază utilizând modele de testare pentru sisteme;
- **suita de teste de regresie** testează componentele de bază şi sistemul.

Suita de teste ierarhice incrementale este cea mai complexă dintre suitele de testare. Se poate construi suita utilizând oricare din următoarele modele de testare:

- modelul de testare bazat pe stări de tranziţie;
- modelul de testare bazat pe fluxul de tranzacţii;
- modelul de testare bazat pe excepţii;
- modelul de testare bazat pe fluxul de control;
- model de testare bazat pe fluxul datelor.

Suita de teste ierahice incrementale este un set de scenarii de test, care tratează obiectele ca instanţe ale claselor şi ca parte a unui sistem corelat cu alte sisteme. Termenul ierahice şi incrementale exprimă scopul de a testa obiectul în contextul ierahiei de moşteniri şi a piramidei sistemului. Astfel orice obiect nu constă numai în proprietăţile care i se atribuie, dar de asemenea şi în acelea pe care le moşteneşte de la ascendenţii săi mai generali şi mai abstracţi. Trebuie deci ca operaţiunea de testare a obiectului să aibă în vedere acest lucru.

La o extremă poate exista o singură serie de test ierarhice şi incrementale pentru întregul sistem; la cealaltă extremă o suită pentru fiecare clasă. Soluţia optimă este undeva la mijloc.

O modalitate potrivită de a începe este aceea de concentrare asupra sistemului, acesta conţinând de obicei toate clasele ce pot fi subiectul unui singur grup de teste. Este mai logic în unele cazuri să se lucreze cu subseturi de clase într-un sistem, acest lucru fiind similar cu conceptul de testare a clusterelor în timpul integrării.

Este important să se facă diferenţa dintre clasele concrete şi cele abstracte. Putem testa clasele concrete direct, lucru pe care nu îl putem face în cazul claselor abstracte datorită faptului că acestea nu pot fi instanţiate. Structural, scenariile se construiesc separat pentru clasele abstracte, dar se rulează împreună cu testele pentru subclase concrete. Păstrarea scenariilor de testare separat permite reutilizarea testelor pentru subclase diferite profitând din plin de moştenire pentru reutilizarea lor. Când se instanţează un obiect şi se execută teste asupra sa, acestea testează şi clasa a cărei instanţă este obiectul şi superclasa sa urcând în piramidă.

Esenţa metodei ierarhice şi incrementale de testare este ansamblul de relaţii dintre clase. În lumea orientat obiect, când se testează o clasă se testează în acelaşi timp şi clasele părinte, construind teste, acestea pot fi reutilizate ulterior şi pentru testarea subclaselor.

Fiecare subclasă este rezultatul combinării structurii şi comportamentului claselor părinţi cu atribute şi metode noi. În figura 2 este o reprezentare grafică din care se observă cum prin moştenire, clasa derivată D se obţine prin combinarea clasei de bază B cu un modifcator, M, ce cuprinde metodele şi proprietăţile specifice clasei derivate. Astfel se poate scrie că $D = B \oplus M$

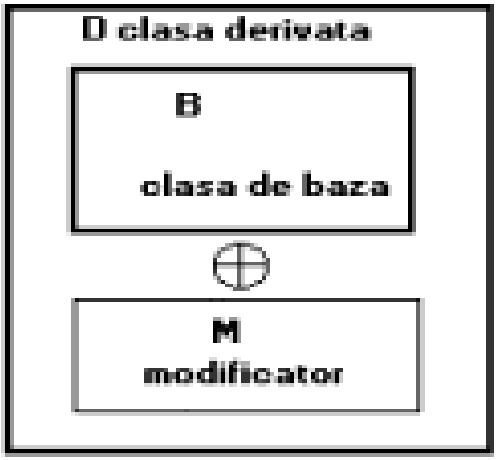

Fig. 2. Moştenirea, o tehnică de modificare incrementală

În cadrul unei clase derivate au fost distinse trei metode:

- **metode noi,** în subclase, incluzându-le pe acelea cu acelaşi nume ca metoda din superclasă dar cu parametri diferiţi;
- **metode recursive,** definite într-o superclasă şi care nu sunt supradefinite sau supraîncărcate în subclasă;
- **metode redefinite,** definite într-o superclasă şi supradefinită sau supraîncărcată într-o subclasă.

Metodele abstracte sunt metode de orice tip asociate cu clase abstracte. Metodele abstracte pot sau nu să aibă o implementare. Dacă nu au, subclasle trebuie să redefinească metoda, supraîncărcând-o. Fiecare din aceste metode are o cerinţă diferită de testare în contextul clasei, de aceea este importantă această tipologie. Astfel:

- metodele noi – necesită o testare completă;
- metodele recursive – necesită o retestare limitată dacă metoda interacţionează cu funcţii(metode) membre noi sau redefinite. Este nevoie doar să fie retestat obiectul care interacţionează, nu toate obiectele;
- metodele redefinite – se retestează prin reutilizarea modelelor de teste şi obiecte dezvoltate din specificaţii mai degrabă decât din controlul logic intern.

Acest lucru înseamnă că nu este nevoie să se testeze fiecare tip de obiect în parte dacă acesta ar fi unul complet nou; va fi nevoie să se testeze doar părţile într-adevăr noi. Acest lucru creşte simţitor productivitatea testării, precum şi productivitatea programării şi a proiectării.

Reutilizarea structurală a acestui tip este partea cheie a paradigmei orientării obiect.

Planul de testare ar trebui să corespundă listei detaliate de clase şi metodele de testare.

Moştenirea multiplă face clasificarea mai dificilă, dar nu afectează categoriile în sine. Anumite modele de moştenire multiplă, cum ar fi cele din C++, fac să existe aceeaşi superclasă în două sau mai multe instanţe, în obiect, creând astfel mai mult decât o metodă recursivă. Astfel se poate numi metodă "diferită" o metodă care este de fapt aceeaşi doar atribuind-i calea superclasei. Atâta timp cât metoda este recursivă nu este nevoie ca acest lucru să fie testat. Dacă metoda este redefinită, nu va mai fi nevoie să fie testate cele ale superclaselor, ci doar redefinirea. Singura situaţie în care ar putea fi nevoie de o testare a mai multor căi diferite cu aceeşi metodă este atunci când metoda interacţionează cu metode virtuale redefinite într-o subclasă.

Este de remarcat faptul că legarea dinamică joacă un rol în determinarea cerinţelor testării, dar numai în contextul testării de integrare. Legarea dinamică creează posibilitatea unui set de mesaje (combinaţii de obiecte emiţătoare şi receptoare de mesaje), ceea ce înseamnă că va fi nevoie de mai multe teste în locul unuia singur pentru a testa un mesaj specific. Când doar se testează metode, ca fiind opuse mesajelor, nu prezintă interes altceva decât varietatea de valori posibile ale datelor pe care mesajele le pot introduce în metode.

În funcţie de tipul modelului de test utilizat şi de dezvoltarea suitei de test, ar putea fi necesară adăugarea unor obiecte de test condiţilor testelor care apar în metodele recursive. De exemplu dacă o metodă recursivă utilizează unele atribute şi subclasele adaugă o nouă serie de valori atributelor, ar trebui adăugate obiecte de testare şi scenarii pentru a testa noua serie ca parte a unui model de flux de control sau model de flux de date. Această situaţie nu este tocmai obişnuită, dar întotdeauna trebuie reexaminate modelele de testare pentru superclase pentru a fi sigur faptul că acoperirea este adecvată.