Tipuri generice de testare

Scopurile și obiectivele sunt diferite pentru diferite nivele de testare prezentate în lecțiile precedente. Din această cauză pentru fiecare nivel de testare sunt relevante diferite tipuri de testare.
Pot fi distinse următoarele tipuri generice de testare:

- Testarea funcțională
- Testarea non-funcțională
- Testarea structurală
- Testarea statică
- Testarea dinamică

Testarea funcțională

Testarea funcțională include toate tipurile de teste care verifică comportamentul sistemului pentru diferite date de intrare. Pentru a proiecta cazuri de test funcționale, sunt utilizate metodele de testare a cutiei negre (black box technics)care vor fi discutate în următoarele teme, iar baza de testare este specificația cerințelor (dacă există ☺).
Este foarte important de înțeles ce este o cerință software (funcțională sau nonfuncțională). Câteva definiții care pot clarifica aceasta și ușura lucrul testerului atunci când lipsesc specificațiile sau nu sunt complete.

Cerinţa software este:

1. o ***condiţie*** sau ***capabilitate*** necesar a fi îndeplinită de către un sistem, pentru ca un *utilizator* să poată rezolva o anumită *problemă* sau să atingă un anumit *obiectiv*;

2. o ***condiţie*** sau ***capabilitate*** pe care un *sistem* trebuie să o realizeze sau să o posede pentru a satisface un contract, standard, specificaţie sau alt *document* formal impus;

3. un ***document*** – reprezentarea unei *condiţii* sau *capabilităţi*, aşa cum este descrisă la punctele 1 şi 2 de mai sus.

Capabilităţile desemnează ceva ce un sistem software furnizează utilizatorilor, fie un anumit comportament fie un anumit atribut. Deşi termenul provine din englezescul "capability", limba română are aici privilegiul de a avea pentru capabilitate o descriere intuitivă: ea desemnează ceva ce un sistem este capabil să facă, ceva ce sistemul poate.

Printr-o capabilitate a unui sistem putem desemna fie un comportament fie un atribut. De exemplu, o funcţionalitate de genul „sistemul validează formatul datei atunci când utilizatorul înregistrează factura în sistem” este un comportament al sistemului, în timp ce „poziţia unui buton pe ecran” este un atribut. (Mai pe româneşte, în final, un câmp dintr-o bază de date sau o proprietate a unui obiect poate corespunde unui atribut al unei entităţi, în timp ce o metodă a unui obiect înseamnă comportament.)

Se pot stabili următoarele niveluri ale cerinţelor software :

A. Cerinţe de business: acestea sunt cerinţele de pe cel mai înalt nivel şi sunt obiectivele (sau problemele) de nivel înalt ale clientului.

B. Cerinţe utilizator: pe acest nivel sunt task-urile pe care utilizatorul le va putea îndeplini utilizând produsul software. Aceste cerinţe sunt specificate de obicei sub formă de use case-uri.

C. Cerinţe funcţionale: sunt cerinţele adresate direct viitorului sistem, funcţionalităţile care trebuie dezvoltate pentru ca utilizatorii să îşi poată îndeplini task-urile. Acest nivel este nivelul cel mai apropiat de arhitectura sistemului.

Un alt element cheie sunt caracteristicile cerințelor. Pot fi evinențiate următoarele caracteristici de bază:

1. Necesară

O cerinţă există *dacă şi numai dacă este necesară*. Amintind de prima parte a definiţiei cerinţei, putem spune că cerinţa există dacă există o problemă reală de la care porneşte. *În caz contrar cerinţa nu există*.

Doar pe baza problemei reale vom putea şti dacă o cerinţă se încadrează sau nu în proiect.

Pentru a demonstra că o cerinţă este necesară este nevoie de existenţa unei legături către cerinţele de pe nivelul A.

2. Corectă

Atunci când spunem că o cerinţă trebuie să fie corectă, ne apropiem deja de a doua parte a definiţiei de mai sus (sau mai degrabă le cuprindem pe amândouă). O cerinţă este corectă dacă faţeta denumită *soluţie* este, nimic altceva decât soluţia corectă la problema dată.

De exemplu, un *use case* care este gândit pentru realizarea unui anumit task este corect dacă șirul pașilor descriși conduce la realizarea, fără dubii, a task-ului. În caz contrar cerința descrisă astfel nu este corectă.

În general pentru a determina corectitudinea unei cerinţe este necesar să se facă referire la cerinţele de pe nivelul A.

3. Completă

O cerinţă este completă dacă reprezintă o soluţie completă pentru rezolvarea completă a problemei. Deşi cazurile de incompletitudine sunt greu de descoperit, organizarea de review-uri formale cu participarea personalului din echipa de dezvoltare, de obicei dă rezultate.

4. Consistentă

O cerinţă este considerată consistentă dacă nu intră în contradicţie cu altă cerinţă. De exemplu, următoarele cerinţe sunt inconsistente:

- autovehiculul se va deplasa cu viteza maximă de 100 km/h;
- autovehiculul va parcurge 200 de km în maximum o oră.

5. Verificabilă

O cerinţă este verificabilă (testabilă) dacă permite realizarea validarea fără echivoc a soluţionării ei prin măsurare sau testare. De exemplu, „sistemul va permite derularea optimă a activităţii", „timpul de răspuns va fi cât mai mic posibil" sau „sistemul va putea fi accesat de un număr mare de utilizatori simultan" sunt cerinţe care nu pot fi verificate în mod cert, fără dubii.

Oricând se poate pune întrebarea ce înseamnă derularea optimă a activităţii, când se poate spune că obiectivul a fost atins?

6. Clară (fără ambiguităţi)

O cerinţă poate fi considerată lipsită de ambiguităţi atunci când poate fi interpretată într-un singur fel. Dacă mai mulţi cititori înţeleg lucruri diferite atunci cerinţa este ambiguă.

Pentru a ţine sub control fenomenul existenţei ambiguităţilor trebuie organizate review-uri ale specificaţiei. De asemenea, specificaţiile vor fi folosite ca sursă primară pentru crearea planurilor de teste şi a manualului de utilizare.

7. Trasabilă

Trasabilitatea se referă la posibilitatea de a reface traseul pe care o cerinţă a luat naştere, pornind de la solicitarea iniţială a unui reprezentant al clientului. Acest mod de abordare asigură informaţia care justifică existenţa sau nu a cerinţei, precum şi posibilitatea de a reface drumul pe care a apărut o cerinţă, atunci când apar dubii asupra acesteia, asupra sursei sau asupra raţiunii ei.

Testarea funcțională prevede crearea cel puțin a unui caz de test pentru fiecare cerință a produsului. Iar pentru aceasta sunt folosite tehnicile black-box de proiectare a cazurilor de test.

Testarea non-funcțională

Cerinţe non funcţionale

Una dintre greşelile cele mai frecvente în specificaţiile software este tratarea superficială a cerinţelor nefuncţionale.

Acestea pot fi cerinţe legate de atributele de calitate a produsului, cerinţe privind performanţa, respectarea unor standarde, regulamente, contracte, interfeţe externe sau alte constrângeri de design.

Determinarea *cerinţelor nefuncţionale* trebuie să urmeze un proces sănătos, de la determinarea lor şi până la specificare.

În primul rând, determinarea acestor cerinţe este un lucru dificil, pentru că, în general utilizatorii nu sunt interesaţi, şi nici puşi în temă, în legătură cu costurile cerinţelor lor şi au tendinţa să exagereze: "sistemul trebuie să lucreze 24 de ore din 24", "timpul de răspuns?... păi sistemul trebuie să răspundă instantaneu la orice comandă", "nu se admite nici un bug în funcţionarea sistemului".

În realitate nu este deloc important, nici măcar util, să se arunce banii pe obţinerea unor caracteristici care, de fapt, sunt excepţionale, ale produselor software. Şi asta pentru că nu toate aplicaţiile software sunt folosite pentru ghidarea navetelor spaţiale sau pentru controlul centralelor nucleare.

Dimpotrivă, am văzut cu toţii site-uri de succes care nu răspund chiar instantaneu la solicitări, sau aplicaţii de calcul tabelar, extrem de utile şi practice dealtfel, care nu pot gestiona fără pierderi de performanţă cantităţi foarte mari de date.

Prin urmare, ceea ce trebuie neapărat identificat sunt cerinţele cu adevărat importante, specifice business-ului respectiv, care pot produce pierderi sau dificultăţi reale. De pildă, în unele cazuri, ar putea fi necesară disponibilitatea sistemului 24 de ore din 24 pentru facturare, chiar dacă sunt acceptabile unele deprecieri ale timpului de răspuns în anumite intervale orare. În fiecare caz în parte există un raport optim între performanţe şi costuri.

Astfel cerinţele non-funcţionale explică comportamentul aplicaţiilor în practică.

Exemple de cerinţe non-funcţionale:

- Instalabilitatea - proceduri de instalare.
- Interoperabilitatea - execuţia aplicaţiei în diferite medii de lucru.
- Stabilitatea - posibilitatea de a face schimbări în sistem.
- Performanţa - comportamentul normal, aşteptat.
- Încărcarea (loading) - comportamentul sistemului când resursele sunt încărcate la maxim.
- Portabilitatea - utilizarea pe diferite platforme.
- Capacitatea de recuperare - restabilirea sistemului după încheerea necorespunzătoare a execuţiei.
- Fiabilitatea - capacitatea softului de a-si păstra funcţionalităţilor după o perioadă de timp.
- Utilizabilitatea - simplitatea de comunicare a utilizatorului cu sistemul.

Câteva tehnici specifice testării cerințelor non funcționale:

1. Testarea pentru determinarea capacităţii de recuperare este un test de sistem care, printr-o multitudine de căi, forţează aplicaţia software să îşi încheie execuţia în mod necorespunzător. Prin acestea se testează capacitatea aplicaţiei de revenire dintr-o situaţie necorespunzătoare.

2. Testarea securităţii se realizează pentru a verifica eficienţa mecanismelor de protecţie dintr-un sistem software şi dacă acesta se comportă corespunzător la atacurile externe.

3. Testarea de solicitare se execută astfel încât sistemul să funcţioneze cu un volum mai mare de resurse decât cel normal, cu scopul de a determina fiabilitatea aplicaţiei şi momentul în care funcţionarea aplicaţiei devine critică şi pentru a lua măsurile corespunzătoare, astfel încât să nu se ajungă în exploatarea de zi cu zi a sistemului informatic la astfel de situaţii.

4. Testarea de încărcare constă în rularea sistemului software în condiţiile în care resursele (memorie, microprocesor, reţea) sunt încărcate la maxim astfel încât se ajunge la situaţii critice, în care o parte din resurse nu mai sunt disponibile. Se urmăreşte capacitatea sistemului de a-şi întrerupe funcţionarea fără pierderea sau coruperea datelor.

5. Testarea performanţelor se realizează pentru determinarea conformităţii sistemului cu cerinţele de performanţă impuse. De exemplu sistemul este încărcat într-un interval de timp pornind de la capacitatea minimă până la capacitatea maximă şi se verifică dacă resursele sistemului se află în limitele corespunzătoare şi nu sunt întârzieri în executarea funcţiilor aplicaţiei.

Securitatea poate fi considerată o cerinţă funcţională a sistemului.