VRSTE, TEHNIKE I METODE TESTIRANJA

Ručno i automatsko testiranje - osnovni pojmovi

Testiranje softvera je postupak proveravanja softvera sa ciljem da se utvrdi da li se ponaša u skladu sa zahtevima i očekivanjima. Tokom testiranja pokušavamo da pronađemo probleme pre nego što oni dođu do korisnika. Zbog toga testiranje doprinosi kvalitetu, pouzdanosti i sigurnijem korišćenju softvera.

Testiranje nije samo pokretanje programa i traženje grešaka. Pre izvršavanja testa potrebno je znati šta proveravamo, sa kojim podacima, kakav rezultat očekujemo i kako ćemo zabeležiti ono što se zaista dogodilo. Zato se u testiranju koriste unapred osmišljeni test slučajevi i test podaci.

Jedna ista funkcionalnost može se proveravati na različite načine. Neke provere obavlja čovek direktnim korišćenjem aplikacije, dok druge može da izvršava programski kod. Na osnovu načina izvršavanja razlikujemo ručno i automatsko testiranje.

1. Ručno testiranje (Manual Testing)

Kod ručnog testiranja tester sam izvršava korake testa: unosi podatke, klikće na dugmad, bira opcije, posmatra ponašanje aplikacije i upoređuje dobijeni rezultat sa očekivanim.

Ručno testiranje je pogodno kada želimo da istražimo novu funkcionalnost, proverimo izgled i jednostavnost korišćenja ili kada se određena provera izvršava samo nekoliko puta. Njegova prednost je što tester može da reaguje na neočekivano ponašanje koje primeti tokom rada.

Nedostatak je što ponavljanje velikog broja istih testova zahteva vreme i pažnju. Čovek može da preskoči korak ili pogreši prilikom ponavljanja testa.

2. Automatsko testiranje (Automated Testing / Test Automation)

Kod automatskog testiranja deo provere obavlja programski kod. Pišemo test koji poziva funkciju ili drugi deo programa, prosleđuje mu unapred određene podatke i proverava dobijeni rezultat.

Automatski test se može pokrenuti mnogo puta na isti način. Zbog toga je naročito koristan kada često menjamo program i želimo brzo da proverimo da li postojeće funkcionalnosti i dalje rade ispravno.

Automatizacija ipak nije zamena za sve ručne provere. Ne mora svaki test biti automatizovan. Biramo provere koje se često ponavljaju, imaju jasno definisan očekivani rezultat i kod kojih automatizacija donosi uštedu vremena.

Najjednostavniji automatski test zasniva se na tri osnovna elementa:

Korak	Šta se dešava
1.	Zadaju se test podaci (test data).
2.	Poziva se kod koji testiramo i dobija stvarni rezultat (actual result).
3.	Stvarni rezultat se poredi sa očekivanim (expected result).

Ako se očekivani i stvarni rezultat poklapaju, test je uspešan - PASSED. Ako se ne poklapaju, test je FAILED. FAILED ne govori automatski gde je greška u programu; on nam govori da dobijeno ponašanje nije ono koje smo očekivali.

3. Ručno ili automatsko?

Izbor zavisi od toga šta testiramo. Ručno testiranje je fleksibilno i važno kada je potrebna procena čoveka. Automatsko testiranje je brzo i dosledno kada iste, jasno definisane provere treba ponavljati. U realnom razvoju softvera ova dva pristupa se dopunjuju.

Ručno testiranje	Automatsko testiranje
Tester izvršava korake.	Kod izvršava unapred napisanu proveru.
Pogodno za istraživanje i vizuelne provere.	Pogodno za česte i ponovljive provere.
Sporije pri velikom broju ponavljanja.	Brzo ponavlja isti skup testova.
Ne zahteva pisanje test koda.	Zahteva pisanje i održavanje test koda.

4. Framework za automatsko testiranje

Kada testova ima mnogo, koriste se test framework-i - alati koji olakšavaju pisanje, organizovanje, pokretanje testova i prikaz rezultata.

Za različite programske jezike postoje različiti framework-i. Na primer, za Javu se često koristi JUnit, dok se za JavaScript može koristiti Jest.

Jest omogućava da kratko zapišemo šta testiramo, koji rezultat očekujemo i da automatski dobijemo informaciju da li je test prošao. U ovom prvom primeru još ne koristimo Jest. Proveru pišemo običnim JavaScript-om da bismo razumeli šta framework kasnije radi umesto nas.

```
test("Pozdrav za Anu", function () {
    expect(Pozdrav("Ana")).toBe("Zdravo Ana");
});
```

Kasnije ćemo ovaj zapis povezati sa onim što već znamo: Pozdrav(„Ana“) daje stvarni rezultat, „Zdravo Ana“ je očekivani rezultat, a expect(...).toBe(...) vrši poređenje.

5. Naš prvi automatski test u JavaScript-u

Kod aplikacije i kod testa možemo držati u odvojenim fajlovima. U script.js nalazi se funkcija koju testiramo:

```
function Pozdrav(ime) {
    return "Zdravo " + ime;
}
```

U fajlu test.js nalazi se kod kojim testiramo funkciju:

```
var stvarniRezultat = Pozdrav("Ana");
var ocekivaniRezultat = "Zdravo Ana";
```

Test podatak je „Ana“. Stvarni rezultat je ono što funkcija vrati, a očekivani rezultat unapred definišemo.

6. Kompletan test u test.js

```
function testPozdrav() {
    var stvarniRezultat = Pozdrav("Ana");
    var ocekivaniRezultat = "Zdravo Ana";

    if (stvarniRezultat == ocekivaniRezultat) {
        document.getElementById("rezultat").innerHTML = "PASSED";
    }
    else {
        document.getElementById("rezultat").innerHTML = "FAILED";
    }
}
```

Klikom na dugme Pokreni test poziva se testPozdrav(). Ako funkcija vrati očekivani tekst, rezultat je PASSED. Ako promenimo funkciju, a očekivani rezultat ostane isti, test može da prikaže FAILED.

7. Zapamti

Pojam	U našem primeru
Test data	„Ana“
Expected result	„Zdravo Ana“
Actual result	vrednost koju vraća Pozdrav(„Ana“)
PASSED	expected = actual
FAILED	expected != actual

Automatski test ne „zna“ sam šta je ispravno. Mi unapred definišemo očekivani rezultat, a test kod automatski izvršava proveru i poredi ga sa stvarnim rezultatom. Framework poput Jest-a kasnije olakšava i organizuje ovaj isti postupak.

Literatura

Miodrag Živković, Testiranje softvera, Univerzitet Singidunum, Beograd, 2018.