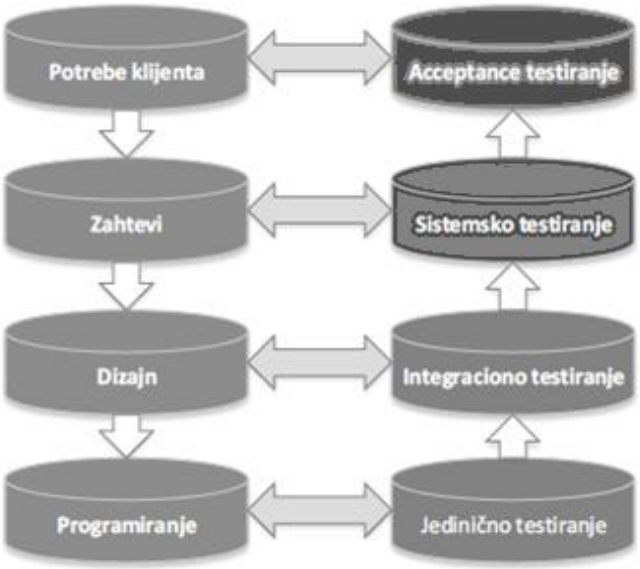
20. OCENA KVALITETA SOFTVERSKIH PROIZVODA

Pojam osiguranje kvaliteta softvera SQA (Engl. Software Quality Assurance) se odnosi na sistematizovane aktivnosti koje obezbeđuju dokaz podobnosti krajnjeg softverskog proizvoda za upotrebu. Obezbeđenje kvaliteta obuhvata skup aktivnosti potrebnih da bi se obezbedilo adekvatno poverenje u sprovedeni proces i stalno unapređenje u cilju stvaranja softvera koji zadovoljava početne specifikacije i odgovara svrsi, osnovnoj nameni i cilju izrade. **Kontrola kvaliteta je proces koji se sprovodi nikako samo na kraju izrade softvera, kontrolom i testiranjem kada se upoređuje kvalitet urađenog softvera sa polaznim specifikacijama.**

Provera kvaliteta softvera je ispitivanje koje verifikuje slaganje softvera sa ciljevima, namenom, planovima, politikom i procedurama različitim metodama, tehnikama i postupcima testiranja.

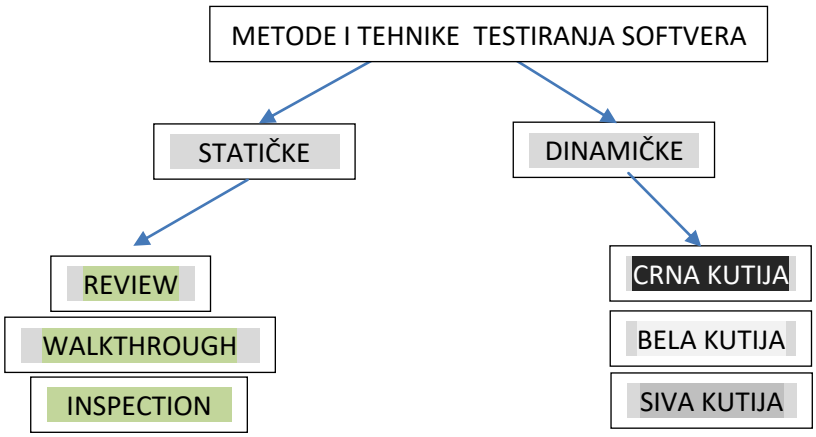


Obezbeđenje kvaliteta softvera je planiran napor da softver ispuni određene kriterijume i da ima dodatne specifične atribute npr. portabilnost, efikasnost, fleksibilnost. SQA aktivnosti i funkcija koje se koriste za praćenje i kontrolu projekta omogućavaju da specifični ciljevi budu postignuti sa željenim stepenom sigurnosti.

Objedinjene aktivnosti SQA čine sistem osiguranja kvaliteta sofvera koji se izvodi u svim fazama razvoja softvera!

Testiranje softvera se deli u dve velike grupe tehnika prema načinu izvršavanja:

- **Ručno** – vrši se manuelnim izvršavanjem test slučajeva prema definisanom scenariju i planu testiranja,
- **Automatsko** – vrši se uz koda, tj. skripta koji se izvršava u nekom specijalizovanom softverskom alatu ili razvojnom okruženju (testRigor, Ranorex Studio, Kobiton, LambdaTest, Avo Assure, Subject7, Selenium)



Statičko testiranje je način testiranja softvera u kom se program testira bez izvršavanja koda odnosno bez njegovog pokretanja, manuelno ili putem specijalizovanih softverskih alata. Ovakva vrsta testiranja može se izvoditi u ranim fazama razvoja softvera, pre izvršavanja dinamičkih testova. Mogu se sprovoditi na specifikaciji zahteva, dizajnu softvera, modelima, arhitekturi, funkcionalnim zahtevima, ali i na izvornom kodu, a tokom sprovođenja testa je moguće veoma rano otkriti razne greške i probleme kao što su: greške u arhitekturi sistema, greške u dizajnu, odstupanja od standarda, neodrživi i komplikovani kod, odstupanja od specifikacije i zahteva. Glavni cilj testiranja je da se u što ranijoj fazi razvoja softvera pronađu defekti.

Dinamičko testiranje se sprovodi tako što se testira softver koji se razvija i može se već izvršavati, odnosno testira se ponašanje softvera sa raznim varijablama u skupu ulaznih podataka i postavki programa. Testovima se nastoje pronaći slabe tačke softvera i defekte u stvarnom okruženju, gde se nakon unošenja ulaznih podataka i izbora ulaznih postavki, rezultati upoređuju s očekivanim rezultatima.

Testiranje softvera se može klasifikovati na dva načina:
- Prema pristupu testiranja,
- Prema nivou testiranja.

Prema pristupu, testiranje se deli na:
- Funkcionalno – testiranje bazirano na specifikaciji,
- Strukturno – nefunkcionalno testiranje bazirano na izvornom kodu softvera.

Prema nivou, testiranje se deli na:
- Jedinično testiranje
- Integraciono testiranje
- Sistemsko testiranje

Funkcionalno testiranje posmatra softver kao zatvorenu, crnu kutiju i implementacija programa u ovom slučaju nije poznata. Zbog toga i potiče naziv testiranje metodom crne kutije. Softver se posmatra kao funkcija koja mapira ulazni skup vrednosti u izlazni skup vrednosti. Testovi se određuju i formiraju na osnovu specifikacije softvera, nezavisni su od implemetacije, upotrebljivi su i ako dođe do promene implementacije. Razvoj testova može da se odvija paralelno sa razvojem softvera. Nedostatak funkcionalnog testiranja je to što se deo implementiranih funkcionalnosti neće pokriti testovima ukoliko nisu navedeni u specifikaciji. Vrste funkcionalnih testova:
- Unit testing – jedinično testiranje,
- Integration testing - integraciono testiranje,
- System testing – sistemsko testiranje,
- Interface testing - testiranje povezanosti (provera kako delovi softvera koji su povezani međusobno komuniciraju - preko servisa, API-ja, servera, drajvera i sl.),
- Regression testing – regresiono testiranje (provera da li softver funkcioniše očekivano nakon izmena u kodu, ubacivanja dodataka (Engl. „Updates“), poboljšanja funkcionalnosti, novih zahteva i sl.),
- User acceptance testing – testiranje prihvatljivosti za korisnika (izvršava se od strane krajnjeg korisnika, klijenta kako bi se verifikovalo da li su zahtevi ispoštovani, da li je sve specificirano i realizovano, radi se nakon jediničnih, integracionih i sistemskih testova).

Strukturno testiranje se fokusira na implementaciji programa i raspoloživom, dostupnom, izvornom kodu. Zbog je dobilo drugi naziv - testiranje metodom bele kutije. Fokus ovog testiranja je na izvršavanju svih programskih struktura i struktura podataka u softveru koji se testira i prema tome se određuju testovi. U ovoj vrsti testiranja se ne proverava specifikacija

softvera, funkcija i zahteva korisnika. Zbog toga nije moguće otkriti da li su specificirane funkcionalnosti zaista implemtirane u programu.

Jedinično testiranje (Engl. "Unit Testing") - Odnosi se na testiranje pojedinačnih jedinica izvornog koda, kao što su klase ili delovi klasa, mada to zavisi od tehnike i načina izrade programa. Najmanja funkcionalna jedinica izvornog koda je najčešće jedna metoda unutar klase. Unit testove mogu da koriste i programeri, a ne samo testeri softvera, kako bi testirali sopstveni napisani kod. Za ovo testiranje se najčešće koristi određeno okruženje za pisanje jediničnih testova (npr. Junit u okviru Java programskog jezika). Jedinični test je deo koda koji testira neki drugi deo koda programa. Izvršavanje testova je najšešće automatizovano i može se izvršiti i ponoviti nekoliko puta, tj. onoliko puta koliko je potrebno i neophodno.

Integraciono testiranje (Engl. "Integration Testing") – Nakon završetka jediničnog testiranja, jedinice se integrišu u celinu. Glavni fokus Integracionog testiranja je na verifikaciji funkcionalnosti i interfejsa između povezanih i integrisanih modula.

Sistemsko testiranje (Engl. "System Testing") **-** Kada se na kraju svi delovi softvera integrišu u kompletan sistem, sistemsko testiranje proverava ponašanje tog sistema kao celine u odnosu na specifikaciju sistema. Kada je većina funkcionalanih zahteva proverena na nižim nivoima testiranja (integracionim testovima), tada se akcenat stavlja na nefunkcionalne zahteve, kao što su brzina rada softvera, njegova sigurnost, pouzdanost, tj. robusnost i sl.