1

Uvod

Testiranje je veoma važna aktivnost u razvoju softvera koja je posebno dobila na značaju kada je vrednost softvera počela da raste. Greške u softveru koji je vredan nekoliko miliona dolara mogu prouzrokovati ogromne novčane štete, tako da se moraju otkloniti što je moguće ranije. Zbog toga se u poslednje vreme ne štedi na aktivnostima testiranja. Postepeno testiranje postaje aktivnost koja je važna kao i sam razvoj softvera tako da je potrebno ozbiljno prići ovoj temi.

Međutim, softverski timovi često ne shvataju proces testiranja i složenost uloga koje postoje u procesu testiranja. Često se smatra da su testeri manje-više sporedno osoblje koje čeka da se softverski sistem napravi, dobije sistem i onda rade sa njim šta god im je volja kako bi pronašli sve probleme. Međutim, proces testiranja je znatno složeniji, sa ozbiljnim planovima, strukturom test tima i metodologijom rada.

Ovo poglavlje daje uvid u osnovne pojmove testiranja softvera.

Pojam testiranja

U ovoj sekciji će biti objašnjeno šta je u stvari testiranje i zašto je ono uopšte važno u softveru. Jedan zanimljiv citat za početak može pokazati šta je u stvari testiranje.

„Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous." – ***James Bach***

(Testiranje je beskonačan proces poređenja nevidljivih stvari sa nejasnim, s ciljem da se izbegnu nezamislive stvari ljudima koje ni ne znamo.)

Greške koje se prave tokom rada su normalna i svakodnevna stvar, i nisu direktno vezane za izradu softvera. Greške se dešavaju u svim oblastima rada i uzrokovane su kako ljudskim, tako i mašinskim faktorima. Imajući u vidu da su greške realna pojava, u svim oblastima rada se uvodi sistem kontrole kvaliteta kojim se greške identifikuju i otklanjaju pre nego što izazovu veće probleme u radu.

Konkretno, u izradi softvera testiranje predstavlja pokušaj da se pronađu greške u softveru koji je napravljen. Softver je implementiran prema korisničkim zahtevima kojima se rešava neki realni problem ili se kreira neka korisna funkcionalnost koja predstavlja nešto što je potrebno krajnjim korisnicima. Kada se implementira, sof-

tver može u većoj ili manjoj meri da odgovara originalnim zahtevima prema kojima je i napravljen. Svako ponašanje softvera koje se ne slaže sa originalnim zahtevima predstavlja grešku koju je potrebno identifikovati i otkloniti. U užem smislu, testiranje predstavlja upravo proveru da li je određeni softver u potpunosti implementiran prema originalnim korisničkim zahtevima.

U širem smislu testiranje predstavlja sistem kontrole kvaliteta (QA – *Quality Assurance)* kojim se ne proverava samo softver već i sve njegove prateće komponente i karakteristike. Kvalitet softvera se može definisati na različite načine kao na primer:

1. Usaglašenost sa zahtevima i potrebama korisnika – jedan od najbitnijih uslova da bi se softver ocenio kao kvalitetan je da pomaže krajnjim korisnicima u radu. Ovo je moguće samo ako softver što je moguće više odgovara potrebama korisnika.
2. Dobri atributi proizvoda kao što su brzina rada, malo zauzeće memorije i prostora na disku, brzina pokretanja.
3. Lakoća održavanja i promena u softveru, kao i prenošenja na druge platforme.
4. Kvalitet dokumentacije, zahteva, dizajna, uputstava za upotrebu i bilo kojih drugih pratećih dokumenata kojima se opisuje softver.
5. Usaglašenost sa standardima što podrazumeva usaglašenost sa organizacionim standardnima pisanja programskog koda, poštovanje opštih standarda (ISO), usaglašenost sa zakonima i slično.
6. Rad u ekstremnim uslovima sa ogromnim količinama podataka, slabim vezama, ograničenim resursima koji su na raspolaganju i slično.

Bez obzira na to kako se definiše kvalitet, svaka definicija predstavlja skup nefunkcionalnih zahteva kojima se opisuje šta se očekuje od softvera. Cilj testiranja kao kontrole kvaliteta je upravo provera da li su ovi zahtevi ispunjeni. Primer provere kvaliteta softvera danas uključuje i proveru smislenosti originalnih zahteva, jasnoće specifikacije i usaglašenosti sa originalnim zahtevima, lakoće korišćenja softvera i slično. Danas se sve manje govori o testiranju softvera kao izolovanoj aktivnosti provere funkcionalnosti i teži se opštoj kontroli kvaliteta svih softverskih komponenti. Za ljude koji rade ovakve poslove često se umesto naziva tester više koristi termin kontrolor kvaliteta (QA).

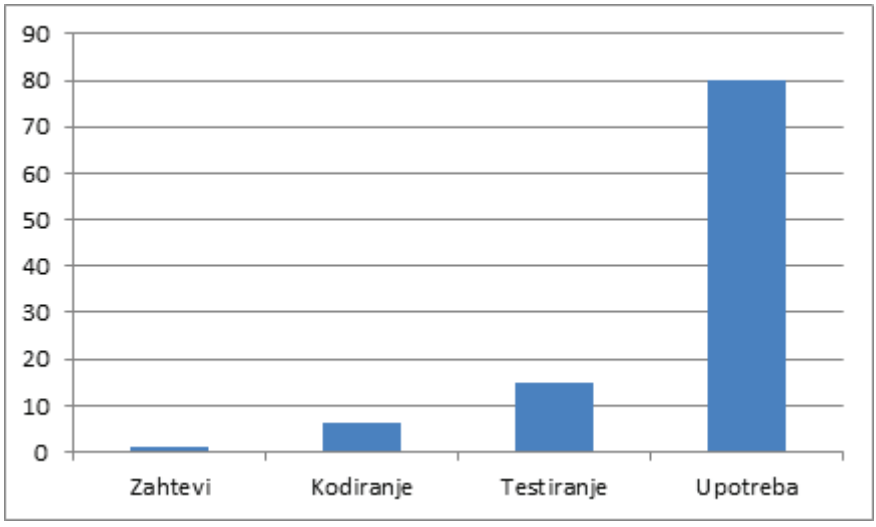
U istoriji tehnike je poznato mnogo grešaka koje su uzrokovane softverskim problemima. Ovi problemi su često uzrokovali višemilionske štete. Primeri takvih grešaka su:

- 1998. Pad Marsovog orbitera. Orbiter je prišao Marsu pod pogrešnim uglom i umesto u orbiti završio je na površini planete. Uzrok je nekompatibilnost podataka koje su moduli slali jedan drugom, čime je uništen projekat vredan 327 miliona dolara.
- 2002. Studija Nacionalnog instituta za standarde i tehnologiju (NIST) je pokazala da softverske greške koštaju američku ekonomiju 59,5 milijardi dolara godišnje. Studija govori da bi se 22,2 milijarde dolara mogle uštedeti boljim testiranjem.

- 2007. Blokiranje aerodroma u Los Anđelesu. Usled greške u softveru, pogrešne informacije su bile poslate u mrežu carine Sjedinjenih Američkih Država, što je uslovilo da 17.000 aviona budu osam sati zarobljeni na aerodromu.
- 2010. Usled greške u softveru kojim su se unosile informacije o organima koji mogu biti izvađeni iz donatora, pogrešni organi izvađeni iz 25 donatora u Velikoj Britaniji. Softver za prikupljanje podataka se koristio od 1999. godine i pronađeno je još 400.000 grešaka.
- 2011. Investicioni fond AXA je morao da plati preko 200 miliona dolara zbog štete koju je nanela greška u softveru zbog koje su investitori izgubili investicije. Fond je znao za grešku, ali je tvrdio da je problem u tržištu i drugim faktorima.

Kao što se može videti u slučaju da se radi o velikim i skupim projektima nikada nije dovoljno testiranja pošto i najmanji problem može da uništi ogroman napor koji je uložen na projektu. Današnji softverski projekti postaju sve skuplji tako da je danas sve veći fokus na testiranju i proverama softvera kako bi se izbegli problemi koji u poslednjem trenutku mogu da unište ceo projekat.

S obzirom da se greške ne mogu izbeći, potrebno ih je što je moguće ranije otkriti kako bi njihovo otklanjanje bilo brže i jeftinije. Kao što se vidi iz prethodnih primera otkrivanje grešaka tek kada sistem uđe u upotrebu može biti suviše kasno, uzrokovati ogromne štete, pa čak i propast celog projekta. Efikasnije je pronaći greške ranije kada ih je lakše ispraviti. Na slici 1.1. prikazani su odnosi cene ispravke grešaka zavisno od trenutka u projektu kada je greška identifikovana.



Slika 1.1. Cena ispravljanja greške zavisno od faze projekta u kojoj je problem identifikovan.

Bilo kakav napor uložen na početku projekta kako bi se identifikovale greške višestruko se vraća kasnije pošto će se greška brže ispraviti i izbeći će se mogućnost da one naprave mnogo veće štete kasnije tokom upotrebe.

Terminologija

U ovoj knjizi ćete naići na dosta termina specifičnih za oblast testiranja. Neki termini koji su bitni za dalje praćenje sadržaja knjige su:

Proizvod rada (engl. ***Work Product*****) –** bilo koji dokument, programski kôd, plan, izveštaj ili test koji naprave članovi projektnog tima tokom rada na projektu. Proizvodi rada su predmet testiranja.

Testiranje – proces evaluacije softvera i svih pratećih proizvoda rada koji se kreiraju tokom razvoja projekta kako bi se utvrdilo da li su u skladu sa očekivanim zahtevima, da li odgovaraju svojoj svrsi, kao i da li postoje neki defekti u njima.

Test slučaj – stanje opisano akcijom koju treba izvršiti, podacima koje treba poslati sistemu, kao i očekivanim odgovorom od strane softvera. Umesto ovog termina koriste se i termini test primer ili slučaj testiranja.

Test scenario – sekvenca akcija koje treba izvršiti tokom testiranja sistema kako bi se proverili jedan ili više test slučaja. Često se umesto termina test scenario koristi termin test procedura ili test skript.

Defekt – greška u softveru koja uslovljava da se softver ne ponaša kao što je očekivano. Defekti u programskom kodu često se nazivaju bagovi (engl. *bug*).

Pronalaženje defekata – proces analize softvera kako bi se pronašlo šta uzrokuje pronađenu programsku grešku. Često se naziva debagovanje (engl. *Debugging*). Pošto se programska greška pronađe i ispravi, sistem postaje stabilniji.

Programski kôd – tekst napisan na nekom jeziku koji se može ili izvršiti direktno od strane računara ili se može prevesti u oblik razumljiv računaru.

Dizajn testa – proces analize i specifikacije test slučajeva i scenarija koji će biti korišćeni tokom testiranja. U dizajn testa pored funkcionalnih test slučajeva mogu da se uključe dizajn testa sigurnosti, opterećenja, platforme, lakoće korišćenja i slično. U literaturi to se još naziva projektovanje test slučajeva.

U slučaju da naiđete na neki od termina koji nije naveden u ovoj listi, možete ga potražiti u rečniku termina u dodacima na kraju knjige. U praksi ćete verovatno naići na različite definicije pojmova pošto postoji nekoliko različitih standarda kojima se definišu pojmovi u testiranju. Neki standardni su IEEE 610.12 standard (standardni IEEE rečnik podataka), Britanski BS-7925-1 standardni rečnik pojmova u testiranju, International Software Testing Qualification Board (ISTQB) ili Test Maturity Model integration (TMMi) rečnik pojmova.

Istorija testiranja

Kao i svi ostali procesi, testiranje se menjalo kroz istoriju sve dok nije dobilo današnji oblik. Testiranje kroz istoriju se može posmatrati kroz nekoliko perioda opisanih u nastavku.

Testiranje radi pronalaženja grešaka

Na samom početku razvoja programiranja nisu postojali projektni timovi u kojima su se nalazili ljudi različitih uloga (analitičari, programeri, testeri, administratori i slično). Čak nije ni postojalo zanimanje programera kao nekog ko se isključivo bavi programiranjem. Programiranje je u to vreme bilo samo alat stručnjaka u nekoj oblasti koji su na taj način rešavali neke probleme iz svoje uže struke. U takvoj situaciji „programeri" su bili većinom naučnici, studenti ili stručnjaci za neku oblast (na primer matematiku, medicinu, vojsku) koji su pokušavali da automatizuju neke poslove u svakodnevnom radu programiranjem računara. S obzirom na to da je često ista osoba i programirala i dorađivala svoj softver, a da je malo ljudi bilo u mogućnosti da to proveri, testiranje su uglavnom vršili sami programeri kako bi ispravili greške koje su našli.

Ovaj period, koji se može posmatrati kao vreme do sredine šezdesetih godina dvadesetog veka, obeležavalo je testiranje koje se uglavnom zasnivalo na pronalaženju programskih grešaka. To je bila usputna aktivnost koja je pratila normalni rad tokom razvoja programa.

Demonstraciono testiranje

Od sredine šezdesetih godina do kraja sedamdesetih godina dvadesetog veka cilj testiranja je bio da se demonstrira da softver radi ono za šta je namenjen. Testeri su se profilisali kao osobe koji demonstriraju rad softvera. U ovom periodu se pojavila jasna granica između testiranja i pronalaženja grešaka:

1. Testiranje je bilo proces kojim se demonstriralo da softver radi upravo ono za šta je originalno namenjen. Svako odstupanje je predstavljalo grešku koju je trebalo ispraviti.
2. Pronalaženje grešaka je predstavljalo proces kojim se softver vraćao u okvire namenjene funkcionalnosti.

U ovom periodu je počela specijalizacija testera koji se isključivo bave proverama funkcionalnosti sistema za razliku od programera koji se bave pravljenjem programa i pronalaženjem mesta u programima koja su uzrokovala greške.

Testiranje koje se vrši samo da bi se dokazalo da nema problema je u većini slučajeva samo pronalaženje programskih putanja u kojima se očekuje da neće biti pronađene greške. U ovom slučaju softver se koristio u nekim očekivanim scenarijima gde su se ispravljali problemi koji zaustavljaju scenario rada. Međutim, ovo je loša praksa pošto to ne garantuje da nema problema u softveru, nego samo da će pronalaženje problema u alternativnim tokovima korišćenja sistema biti prepušteno krajnjim korisnicima.

Destrukciono testiranje

Krajem sedamdesetih i početkom osamdesetih godina dvadesetog veka testiranje je definisano kao proces „izvršavanja programa sa ciljem da se pronađu greške". Uloga testera nije da demonstrira da program radi u skladu sa zahtevima – uloga testera je da dokaže da program ne radi. Testiranje postaje destrukciono – tester počinje test sa predubeđenjem da sistem ne radi i ima samo jedan cilj – da to i dokaže. Opet se promenila definicija testiranja i pronalaženja grešaka:

1. Testiranje je proces otkrivanja grešaka u programu.
2. Pronalaženje grešaka je proces otkrivanja gde se greške nalaze u programskom kodu.

Testiranje se vrši pronalaženjem onih test slučajeva (akcija i podataka koji se koriste prilikom rada sa sistemom) kojim se najlakše dokazuje da stvarno postoje greške u programu. Cilj testera je da pronađe najbolji način da te greške i otkrije.

Ovaj period koji se može posmatrati oko osamdesetih godina dvadesetog veka dovodi testiranje u krajnju fazu razvoja softvera. Kada se program kompletira, testeri ga preuzimaju kako bi dokazali da ne radi. Tek kada testeri ne mogu više da dokažu da program ne radi, on je potpuno stabilan. Ovo odgovara testiranju u modelu vodopada koji je opisan u sledećoj sekciji.

„Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence." – Edsger Dijkstra

(Testiranje može da bude efektivan način dokazivanja da postoje programske greške, ali ne potpuno neadekvatan način da dokaže da greške u softveru ne postoje)

Evaluaciono testiranje

Sredinom osamdesetih godina dvadesetog veka pokazano je da testiranje koje se vrši kao poslednja faza u razvoju softvera ima velikih nedostataka. Naime, praksa je pokazala da je skupo ispravljati greške kada se sistem kompletira pošto ispravke grešaka često dovode do novih grešaka čime se smanjuje stabilnost programa i povećava cena ispravljanja grešaka. Veliki broj grešaka često nije ni posledica nekonzistentnosti sa zahtevima, nego inicijalni zahtevi prema kojima je rađen sistem nisu bili validni. Takvi problemi u zahtevima koji se prenose u programski kôd prave znatno više štete od onih koji su napravljeni u kodu. Primećeno je da bi bilo dobro da se greške identifikuju što je moguće ranije kako ne bi ni prošle do krajnje faze testiranja.

Kao posledica ovih ideja, testiranje je postalo evaluaciono – proces testiranja se prožimao kroz ceo proces razvoja. Zavisno od faze projekta svaki dokument ili zahtev se proveravao kako bi se što je moguće ranije pronašli problemi i kako bi se sprečilo da se rašire u sistemu tokom razvoja. U ovom periodu testiranje je postalo deo svake aktivnosti u procesu razvoja softvera i postalo proces koji se odvija paralelno sa procesom

razvoja. Svaki put kada se napravi novi zahtev, komponenta ili dokument, on ne može ići dalje ako se na neki način ne testira i potvrdi da je spreman za dalju upotrebu.

Preventivno testiranje

Sledeći korak u testiranju je bilo preventivno testiranje koje se razvilo od kraja osamdesetih godina. Za razliku od evaluacionog testiranja kojim se testira svaka komponenta pošto se napravi, u prevencionom testiranju se testovi prave i pre nego što se komponente naprave. Na primer, za programske komponente koje su planirane u sistemu se prvo prave testovi koji će proveriti da li će one raditi, tako da se testovi mogu primenjivati i tokom izrade samih komponenti. Ideja testiranja više nije bila pokušaj da se pronađe gde se greška nalazi u postojećem kodu, nego je cilj otkriti gde se greška može pojaviti u kodu koji još nije ni napravljen. Ovaj princip se održao do danas i prerastao u kompletan proces osiguravanja kvaliteta kojim se obezbeđuje kvalitet svake komponente čak i pre nego što se napravi.

„Intellectuals solve problems, geniuses prevent them"– Albert Ajnštajn.
(Intelektualci rešavaju probleme, geniji ih sprečavaju.)

Ajnštajnova izreka je idealan primer i za preventivno testiranje pošto pokazuje koliko je zahtevan posao testera. Programiranje je intelektualan posao koji je podložan greškama ali programeri su tu da poprave sve probleme koji su nastali. Međutim, da bi tester sprečio da uopšte dođe do problema, on mora da bude izuzetno sposoban što uključuje moć imaginacije i predikcije. Tester mora da se uživi u ulogu korisnika, da razmišlja kao on i da pronađe alternativne tokove rada koji ne bi pali na pamet nikom drugom (verovatno čak ni samom korisniku dok jednom slučajno ne bi izvršio taj tok). Da bi preduhitrio u razmišljanju čak i samog korisnika i tako izvršio prevenciju problema koji se mogu desiti, izuzetnom testeru je potreban određen nivo genijalnosti.

Proces testiranja

Testiranje je proces koji obuhvata veliki broj aktivnosti i usko je vezan sa procesom razvoja softvera. U sledećem poglavlju opisano je kako se proces testiranja softvera uklapa u različite procese razvoja softvera, ali ovde će biti kratko opisan uobičajeni proces testiranja koji se može prilagođavati konkretnim potrebama. Primer aktivnosti testiranja koje se primenjuju tokom faza razvoja projekta je prikazan na slici 1.2.

U ranijim fazama projekta koje predstavljaju upoznavanje sa konkretnim projektom i problemima koje je potrebno rešiti, planira se šta će se testirati i šta je sve potrebno test timu, što predstavlja osnovu za dalji rad. Planiranje se ne završava na početku projekta – zavisno od potreba i zahteva na projektu planiranje se nastavlja do kraja.

Slika 1.2. Aktivnosti u procesu testiranja tokom projekta.

Kada se preciznije definišu zahtevi, počinje proces dizajna testova. Dizajn testova predstavlja definisanje na koji način će se konkretno testirati softver. Kao i planiranje, dizajn se nastavlja sve do kraja projekta pošto svaka izmena ili prepravka sistema uključuje izmenu dizajna testova.

Izvršavanje testova i rešavanje problema se vrše kada su na raspolaganju konkretni delovi sistema koji se mogu testirati. Ove aktivnosti će detaljnije biti opisane u narednim poglavljima ali okviran pregled je prikazan u nastavku.

Paralelno sa ovim aktivnostima obavljaju se prateće aktivnosti kao što su praćenje procesa testiranja, evaluacija i kontrola kvaliteta testiranja, unapređenja procesa testiranja i slično.

Planiranje testa

Planiranje predstavlja prvi korak u svakom procesu rada. Planiranje predstavlja pripremu za ceo proces testiranja i služi da test tim sagleda šta je potrebno uraditi i na koji način. Tokom planiranja se definiše koje vrste testova će biti sprovedene (testovi funkcionalnosti, opterećenja, sigurnosti), metode testiranja (ručno, automatsko, regresivno), strategije (crne kutije, bele kutije), kao i kriterijum završetka testiranja.

Pored toga, tokom planiranja se definiše koliko članova test tima je potrebno, kao i ko će i kada sprovesti aktivnosti testiranja. Kao rezultat planiranja dobija se skup dokumenata koji predstavljaju najopštiji pogled na sistem koji će biti testiran, aktivnosti koje će biti sprovedene tokom testiranja, kao i strategije i alati koji ce biti korišćeni. Pored toga, bitno je isplanirati i proces komunikacije koji će biti korišćen tokom testiranja, kojim se definiše ko će kome prijavljivati probleme, na koji način će se dodeljivati pronađeni problemi projektnom timu, na koji način će se vraćati test timu radi provere i slično. Detaljniji opis aktivnosti u procesu planiranja testa može se naći u poglavlju 3.

Dizajn testova

Kada se definiše šta je potrebno uraditi, pristupa se detaljnoj specifikaciji načina na koji će se aktivnosti predviđene planom izvršiti, i formulišu se konkretna uputstva kako će se vršiti testiranje sistema. Tokom ove faze se analizira sistem koji će biti testi-

ran, identifikuje se šta je potrebno testirati i na koji način. Kao rezultat ove aktivnosti kreira se skup test slučajeva i procedura koje će biti korišćene tokom testiranja sistema. Dizajn testova detaljnije je objašnjen u poglavlju 4.

Postoji razlika u procesu testiranja u kome se haotično proveravaju funkcionalnosti sistema i slučajno ili srećno pronalaze problemi, i procesa u kome se sistematično proverava softver najoptimalnijim metodama. Ova razlika je upravo u postojanju dizajna testa.

Izvršavanje testova

Izvršavanje testova (testiranje) je proces konkretne primene specificiranih test slučajeva i test procedura u skladu sa planom i dizajnom. U slučaju da se izvršavanje testova vrši neplanski i bez specifikacije, testiranje predstavlja aktivnost u kojoj tester dolazi u kontakt sa softverskim sistemom i pokušava da nađe greške. Uspešnost ovakvog procesa zavisi od iskustva i sposobnosti testera, a veoma često i od sreće. Bez obzira da li je ovakvo testiranje uspešno ili ne, uvek postoji veliki rizik da je tester propustio neke bitne funkcionalnosti, ili da ih bar nije dovoljno proverio.

U zrelijim procesima testiranja se pre testiranja precizno definiše šta će biti testirano i na koji način (dizajn testa). Pored toga, planom je definisano kako će testeri primeniti varijacije u testovima kojima se odstupa od test plana, u kojim slučajevima će biti razvijeni automatski skriptovi i procedure za testiranje i slično. Ove aktivnosti se obično ne vrše proizvoljno nego je precizno isplanirano kako će se i kada izvršavati. Izvršavanje testova prate i dve dodatne aktivnosti kojima se zaokružuje ceo proces testiranja:

1. Praćenje statusa problema – ovo je aktivnost koja se često izvršava paralelno sa fazom izvršavanja testova i obuhvata praćenje životnog ciklusa prijavljenih problema što uključuje rešavanje problema od strane programera, potvrđivanje da je problem rešen od strane testera ili reaktiviranje problema i vraćanje programerima ako nije korektno rešen.
2. Izveštavanje – završna aktivnost u procesu testiranja je kreiranje izveštaja kojim se opisuje šta je testirano i potvrđuje da je softver spreman za korišćenje u skladu sa kriterijumima kvaliteta definisanim u test planu.

Prateće aktivnosti

Paralelno sa opisanim aktivnostima vrše se i posebne aktivnosti kojima se kontroliše proces testiranja i analizira na koji način se on može poboljšati. Neke prateće aktivnosti koje se vrše paralelno sa navedenim aktivnostima testiranja su:

1. Praćenje i kontrola procesa testiranja koja predstavlja skup aktivnosti kojima se utvrđuje da li se sve aktivnosti u procesu testiranja izvršavaju po planu. Ovo je izuzetno važno pošto nekada trajanje izvršavanja testova može da probije rokove zato što se u sistemu otkrije veliki broj nedostataka koje je potrebno testirati ponovo.

2. Kontrola kvaliteta testiranja – ova aktivnost predstavlja proveru da li se kontrola kvaliteta izvršava na zadovoljavajući način. Kada se sistem testira, nije dovoljno samo proći kroz funkcionalnosti i ostale zahteve. Bitno je uveriti se da je sistem testiran na dovoljno dobar način i da projektni tim može da bude siguran da se kvalitetan proizvod isporučuje klijentima.
3. Kontrola promena kojom se prate promene u sistemu i analizira koje promene bi trebalo da se urade u testovima. Ova aktivnost uključuje i kontrolu verzija testova, kao i usaglašenost verzija testova sa verzijama programskog koda.
4. Evaluacija performansi tima za testiranje predstavlja skup aktivnosti kojima se na osnovu određenih metrika i indikatora performansi utvrđuje kojim tempom se testira sistem.
5. Poboljšanje procesa testiranja predstavlja skup mera kojima se na osnovu rezultata kontrole i evaluacije aktivnosti uvode određene izmene kako bi se proces poboljšao.
6. Trening članova tima za testiranje predstavlja skup aktivnosti kojima se vrši obuka članova test tima.

Prateće aktivnosti koje će se izvršavati tokom procesa testiranja definišu se tokom planiranja testa i zavise od potreba na projektu, pravila i politike organizacije po pitanju testiranja, kao i zrelosti procesa testiranja.

Uloge u timu za testiranje

U slučaju da je implementiran potpun proces testiranja potrebno je okupiti veći broj ljudi sa različitim ulogama, znanjima i sposobnostima koji će zajedno uspešno testirati softver. U procesu testiranja se mogu uključiti članovi test tima sa različitim ulogama zavisno od njihovog znanja.

Svaka softverska organizacija ima svoju klasifikaciju pozicija u test timu. Ovde će biti izvršena podela uloga na osnovu znanja i sposobnosti koje moraju da imaju pojedini članovi test tima što ne znači da se ove uloge ne mogu kombinovati u skladu sa organizacionom strukturom tima. U ovoj knjizi će se koristiti sledeće uloge u timu za testiranje:

- Tester je svako ko ima dovoljno znanja u korišćenju softvera ili znanja u određenoj oblasti primene softvera i ko može da koristi softver i ispituje da li se ponaša kao što je očekivano.
- Test inženjer je neko ko ima više tehničkog iskustva i znanja koja mu omogućavaju da koristi test alate, prilagođava okruženje potrebama testa, i više se bavi tehničkim delovima testa.
- Test analitičar je neko ko je u stanju da analizira originalne zahteve i definiše test slučajeve koje je potrebno izvršiti kako bi se proverilo da li su zahtevi ispravno implementirani.

- Test menadžer ili vođa test tima je osoba sa dosta iskustva u testiranju, onaj ko može da sagleda kompletan proces testiranja, menja ga u skladu sa potrebama projekta i vodi ceo test tim tokom provere softvera.

Ove uloge su detaljnije opisane u narednim sekcijama.

Microsoft ima poziciju koju naziva „Software development engineer in test (SDET)" koja po opisu funkcije obuhvata uloge testera, test analitičara i test inženjera. Bez obzira na naziv, ovakva podela olakšava shvatanje šta se očekuje od zaposlenih koji rade u test timu.

Tester

Tester je bilo koji član tima koji proverava validnost sistema ili njegovih komponenti. U praksi tester ne mora da ima neko tehničko predznanje o aplikaciji ili platformi koja se testira pošto mu je fokus na poslovnom procesu i korisničkom pogledu na sistem. Često su testeri aplikacija ljudi koji imaju neko znanje o problemu koji rešava aplikacija i mogu da prođu kroz funkcionalnosti kako bi proverili da li su one ispravno implementirane. Na primer, u slučaju da se implementira neka finansijska aplikacija, često testiranje vrši neki ekonomista koji dovoljno dobro poznaje problematiku i koji može da korišćenjem aplikacije proveri da li su sva poslovna pravila ispravno implementirana. Kod testera najvažnija osobina je snalažljivost, brzo shvatanje rada kako softvera tako i korisnika.

Osnovni zadatak testera je da shvati na koji način će korisnici koristiti softver, da što je moguće verodostojnije „simulira" krajnjeg korisnika, da razmišlja kao krajnji korisnik, radi iste akcije, i načini iste greške koje bi i stvarni korisnik aplikacije načinio kada bi mu se dodelila aplikacija.

U nedostatku testera, često se testeri „privremeno regrutuju" iz ostalih delova tima tako da ponekad u kritičnim situacijama svi programeri, analitičari, menadžeri koji su na raspolaganju postaju „testeri" koji započinju ili ad-hoc testove u sistemu ili prate pripremljene skripte za testiranje. Ovo bi trebalo da bude izuzetak a ne pravilo.

Neki ljudi misle da je tester „nekvalifikovani" član test tima koji samo radi ručno testiranje na osnovu procedura i specifikacije koja mu je dodeljena. Još gore je predubeđenje da je tester samo neko ko će malo da „prođe kroz aplikaciju" i proveri da li ona ispravno radi, ili da slepo prati definisane test slučajeve. Tester je osoba koja unosi „ljudsku" komponentu u proces testiranja, identifikuje se sa korisnikom, razmišlja kao on i pokušava da vidi sistem na način na koji ga ostali, „tehnički orijentisani" članovi projektnog tima ne vide. Uklapanje u pojedini domen (finansije, novinarstvo) je izuzetno težak zadatak, tako da je sasvim neopravdano smatrati da je tester po znanju ili sposobnostima lošiji od ostalih članova tima. Često je tester pravi ekspert u određenoj oblasti koji testira ono što ni jedan analitičar ni inženjer ne mogu da zamisle.

Jedina stvar koja je bitna za testera jeste da je upoznat sa aplikacijom koju testira i da je ispravno testira i prijavljuje probleme. Osnovne aktivnosti koje tester radi su:

1. Izvršavanje testova – testeri ili prate pripremljene scenarije za testiranje ili vrše ad-hoc testove kako bi pronašli greške koje ne pokrivaju planirani scenariji testiranja.
2. Analiza pronađenih problema – svaki problem koji je pronađen se analizira kako bi se prikupilo što je moguće više informacija o tome zašto je problem nastao i pod kojim uslovima se može reprodukovati.
3. Izveštavanje o pronađenim problemima – pošto su pronađeni problemi analizirani i potpuno opisani oni se prijavljuju u sistem za prijavljivanje grešaka gde se dodeljuju razvojnom timu kako bi bili rešeni.
4. Učestvovanje u analizi i dizajnu testova – testeri zajedno sa ostalim članovima test tima identifikuju šta će biti testirano i na koji način. Iako njihova primarna aktivnost nije identifikacija i specifikacija test slučajeva koji će se koristiti, oni zajedno sa analitičarima učestvuju u definisanju načina na koji će testiranje biti sprovedeno.
5. Pregled i evaluacija kvaliteta i smislenosti zahteva i specifikacije softvera.

U nekim timovima testeri nisu aktivno uključeni u evaluaciju zahteva, analizu i dizajn testova. Ove aktivnosti često radi isključivo analitičar test slučajeva koji prosleđuje rezultate testeru. Međutim, u pravim timovima za testiranje svi su uključeni u analizu zahteva, pošto i analitičari i inženjeri i testeri iz svog ugla mogu da otkriju probleme u zahtevima i da aktivno učestvuju u analizi.

U aplikativnom softveru i poslovnim aplikacijama testeri su dominantna uloga u test timu. U manjim projektima sa manjim budžetom često su i jedina uloga.

Analitičar testova

Analitičar testova ili test analitičar, predstavlja osobu koja je dobro upoznata sa sistemom, razume zahteve koji su implementirani i može da identifikuje metode kojima će ti zahtevi biti provereni. Test analitičar bi trebalo da ima dovoljno znanja iz domena u kome radi softver kako bi ravnopravno sa analitičarima sistema i krajnjim korisnicima mogao da raspravlja o funkcionalnostima, da ispravno definiše na koji način će zahtevi biti testirani i pod kojim uslovima. Test analitičar predstavlja spregu između korisnika i analitičara sistema koji znaju šta su zahtevi, i testera i test inženjera koji bi trebalo da provere da li su ti zahtevi ispravno implementirani u sistemu. Pored toga, test analitičar mora da ima iskustva u tehnikama dizajna test slučajeva koje mu omogućavaju da na osnovu funkcionalnih zahteva identifikuje kako se na najoptimalniji način ti zahtevi mogu proveriti. Osnovne aktivnosti koje radi test analitičar su:

1. Analiza sistemskih zahteva u cilju validacije zahteva i razumevanja potreba korisnika.
2. Specifikacija test slučajeva koji će se koristiti od strane test tima kako bi se verifikovalo da softverski sistem radi ono što je predviđeno.

3. Pregled i evaluacija kvaliteta zahteva i specifikacija.

Test „analitičar" je pomalo nesrećno izabran naziv pošto izgleda kao da je kompletan proces analize i razmišljanja dodeljen analitičaru, dok ostali slepo prate rezultate analize. Ovakav stav je pogrešan – svako ko se bavi testiranjem mora da utroši neko vreme na analizu posla koji će raditi. „Analiza" koju vrši test analitičar predstavlja analizu metoda testiranja koje je potrebno primeniti, ali i svi ostali članovi test tima moraju da analiziraju i shvate problem koji testiraju pre nego što počnu da rade.

Treba imati na umu da je svako ko dolazi u kontakt sa nekim problemom u stvari, na neki način, test analitičar. Gde god se pojavi neki zahtev ili problem kog treba proveriti, i kad god je potrebno razmisliti na koji način treba izvršiti test član test tima postaje test analitičar, bez obzira da li je nominalno tester ili test inženjer.

Inženjer testova

Uloga test inženjera je slična testeru uz razliku u tome što test inženjer ima više tehničkog iskustva. Test inženjer kao i tester izvršava testove definisane od strane test analitičara, ali je više usredsređen na tehničke aspekte testiranja. Test inženjer mora da zna veliki broj tehničkih termina vezanih za platformu, okruženje, korišćeni programski jezik i alate kako bi izvršio specifične testove koje funkcionalni tester ne izvršava. Konkretno, test inženjer je pravi radnik u procesu testiranja koji pravi komponente kojima se olakšava i ubrzava testiranje.

Kao što je tester dominantna uloga u testiranju aplikativnog softvera, tako je test inženjer dominantna uloga u testiranju sistemskog softvera pošto se sve sistemske komponente testiraju uglavnom programski.

Tipičan posao test inženjera je kreiranje automatskih testova kojima se testiraju iste funkcionalnosti koje tester testira ručno, ali se ovi testovi mogu pokretati po želji. Da bi implementirao automatske testove, test inženjer mora da bude upoznat sa korišćenjem alata ili jezika koji se koriste za kreiranje automatskih testova, tako da su mu potrebna određena programerska znanja. Pored toga, test inženjer mora da poznaje fizičke karakteristike sistema pošto se njemu obično dodeljuje testiranje performansi, sigurnosti, opterećenja i slično. Testiranje pojedinih komponenti ili podsistema, koje takođe vrši test inženjer, obavlja se pisanjem skriptova ili programa na nekom standardnom programskom jeziku tako da je neophodno da test inženjer ima određeno programersko iskustvo.

Kao što se test analitičar može posmatrati kao sprega između korisnika i zahteva sa jedne strane i test tima sa druge strane, test inženjer predstavlja spregu između programerskog tima i tehničkih karakteristika sistema sa jedne strane i test tima sa druge strane. Osnovna zaduženja test inženjera su:

1. Implementacija skriptova i programa za automatsko testiranje.
2. Testiranje komponenti za koje je potrebno poznavanje tehničkih protokola komunikacije.

3. Testiranje nefunkcionalnih karakteristika sistema kao što su performanse, sigurnost, rad pod opterećenjem, promena platforme i slično.
4. Pregled tehničkih komponenti kao što je tehnički dizajn softvera i programski kôd.

Test inženjer je po sposobnostima sličan softverskom inženjeru s obzirom da zaposleni na obe pozicije moraju da budu tehnički potkovani i da budu sposobni da samostalno programiraju određene komponente (bilo kôd koji implementira funkcionalnosti, bilo kôd koji testira kôd u kome su implementirane funkcionalnosti). Softverski inženjer i test inženjer mogu se menjati na svojim pozicijama u timu pošto mogu da rade slične stvari.

Test menadžer/vođa test tima

Test menadžer, menadžer testiranja ili vođa test tima je lider u test timu i njegov osnovni zadatak je da kontroliše rad test tima, da mu obezbedi sve potrebne uslove za rad i da se postara da se primenjuje proces testiranja. Uloga test menadžera je komunikacija sa ostalim timovima koji imaju veze sa sistemom koji se testira, organizacija članova test tima, nadgledanje komunikacije, definisanje generalnih ciljeva testiranja i slično. Test menadžer je često sprega između menadžmenta i test tima. Osnovna zaduženja test menadžera su:

1. Procena vremena potrebnog za testiranje, broja ljudi u test timu i definicija test plana.
2. Obezbeđivanje resursa potrebnih za testiranje (ljudi, alati, hardver, laboratorijsko okruženje).
3. Nadgledanje i kontrola test aktivnosti.
4. Podnošenje krajnjeg izveštaja o verifikaciji sistema koji je testiran.

Bilo bi dobro da test menadžer ima jake veze sa višim menadžmentom pošto je njegov glavni zadatak da obezbedi potrebno vreme i resurse ostalim članovima test tima. Veoma često neki projekt menadžer preuzima ulogu test menadžera.

Test menadžer se razlikuje od drugih članova test tima i po svojim „političkim" sposobnostima pošto on mora da ima dovoljno znanja i sposobnosti da se snađe u organizaciji u kojoj se razvija sistem. Tokom projekta se dešavaju razni problemi, konflikti i neočekivane stvari koje utiču na tok razvoja projekta što često utiče i na test tim – test menadžer mora da zna kako da se izbori sa ovim problemima.

Nekada projekt menadžer postaje i test menadžer i tako kontroliše ceo tim. Ovo može da bude problem pošto ove dve uloge imaju različite interese. Primarni cilj projekt menadžera je da dovede projekat kraju u okviru rokova i budžeta, tako da je često na iskušenju da „skraćuje" vreme za testiranje ako projekat kasni. Naravno, njegov cilj je da se isporuči kvalitetan softver, ali ako zna da će projekat propasti ako se ne dostigne rok ili ako nema više novca za programere, onda je često kvalitet sekundaran (iako to

niko neće priznati dok ne dođe u realnu situaciju). S druge strane, primarni cilj test menadžera je kvalitet i sigurnost sistema bez obzira na budžet i rokove. Ako je potrebno još mesec dana rada sa dva nova programera inače će sistem biti neprihvatljivo loš, test menadžer će primorati tim da odloži rokove. U uvodu je naveden primer orbitera oko Marsa koji se srušio. Projekat je bio završen u roku i budžetu, ali da li je na kraju bio i uspešan? Konflikt između test menadžera i projekt menadžera i kompromis koji na kraju moraju da postignu dovodi do sistema sa optimalnim rokovima, budžetom i kvalitetom.

Ono što uvek mora da bude na umu svim članovima tima je da se softverski projekti rade po striktnim planovima i rokovima koji se moraju ispoštovati. Jednom potpisan ugovor i utvrđen rok mora da se ispuni, inače projektni tim, pa čak i cela softverska organizacija, trpi posledice (npr. u vidu plaćanja novčanih penala, gubitka posla ili poverenja klijenata). U praksi je uvek veliki pritisak na sve članove tima da ispoštuju rokove uprkos svim problemima koji mogu da se dese. Probleme je potrebno interno rešiti kako se to ne bi odrazilo globalno na projekat i pomeranje rokova, koliko god je to moguće. Test menadžer mora da se izbori sa tim problemima kako bi test tim normalno funkcionisao.

Treba imati na umu da su ovo uloge a ne zanimanja. Članovi test tima mogu menjati uloge tako da tokom faze dizajna testova mogu da budu analitičari, a onda tokom faze izvršavanja mogu da budu test inženjeri ili testeri (naravno pod uslovom da su kvalifikovani za sve uloge). Isto tako test menadžer može da postane analitičar ako treba analizirati test slučajeve ili čak i tester ako je potrebno.