PROJEKTNI RAZVOJ APLIKACIJA

EPIZODA:10

Testiranje

Projektni razvoj aplikacija

Predavanje 10

Testiranje aplikacija

Teorija


PROJECT MANAGER
CLIENT
TESTER
DEVELOPER

Uvod

- Ponekad se svima dogodi da naruše ispravnost koda
- “Sve radi na mom računalu”
- Dobra praksa:
  - Repozitorij koda
  - Dnevni buildovi
- Je li to dovoljno?

Što je testiranje aplikacija?

- Što slučajni prolaznik misli – što je to testiranje?

    1. Postupak pronalaženja grešaka
    2. Postupak provjere da li aplikacija zadovoljava sve zahtjeve

- Kada se testiranje zbiva?
    - Na kraju, nakon što je neka funkcionalnost razvijena
        - Stari način
        - Testeri
    - Cijelo vrijeme tijekom razvoja i nakon razvoja
        - Novi način, agilno programiranje
        - Programeri i testeri

Vrijeme pronalaska grešaka

- Što je kasnije greška pronađena
  - Skuplje njezino rješavanje

		Vrijeme kada je greška otkrivena				
		Prikupljanje zahtjeva	Dizajn sustava	Programiranje	Testiranje prije puštanja u produkciju	Nakon puštanja u produkciju
Vrijeme kada je greška nastala	Prikupljanje zahtjeva	1×	3×	5–10×	10×	10–100×
	Dizajn sustava	-	1×	10×	15×	25–100×
	Programiranje	-	-	1×	10×	10–25×

Testiranje funkcionalnosti i značajki

- Provjera da li određena funkcionalnost radi
  - “Da li korisnik aplikacije može …”
- Zahtjevi koji nisu vezani za funkcionalnost – značajke aplikacije
    - Brzina odziva aplikacije
    - Broj korisnika koji istovremeno mogu koristiti aplikaciju
    - Memorijski zahtjevi
    - Sigurnosni zahtjevi
  - Provjera cijelo vrijeme

Kompatibilnost

- Programeri rade (žele raditi) na najnovijim:
  - Tehnologije, alati, OS, baze, razvojne okoline, preglednici, …
  - Tipovi i verzije
  - Ne provjerava se je li to kompatibilno s prethodnim verzijama
- Greške mogu nastati zbog nedostatka kompatibilnosti
  - Operativni sustav
  - Baza podataka
  - Hardverska platforma
  - Ostale aplikacije

Životni ciklus greške

Životni ciklus greške

Životni ciklus greške

- **(1) Uočavanje**
  - Rušenje aplikacije
  - Nefunkcionalno sučelje
  - Zbunjujuće / nekonzistentno sučelje
  - Neusklađenost s dokumentacijom
  - Nepostojanje funkcionalnosti

Životni ciklus greške

- **(2) Prijava**
  - Ne čuvati na raznim mjestima
  - Aplikacije za praćenje
  - Važno zapisati:
    - Sažetak
    - Koraci za reprodukciju
    - Što je očekivano ponašanje / što se stvarno dogodilo
    - Verzija, okolina, lokacija
    - Utjecaj i prioritet rješavanja

Životni ciklus greške

- **(3) Analiza greške**
  - Određivanje prioriteta
  - Procjena posla
  - Određivanje u kojoj iteraciji će se rješavati
  - Određivanje tko će rješavati

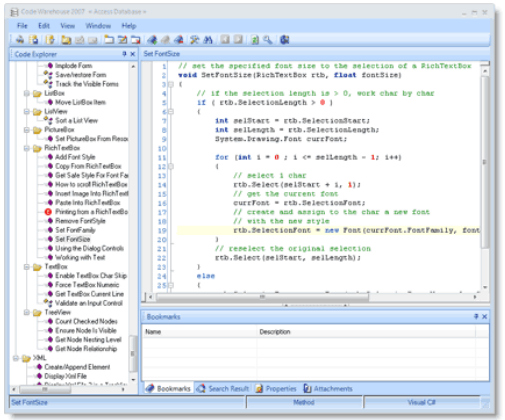
Životni ciklus greške

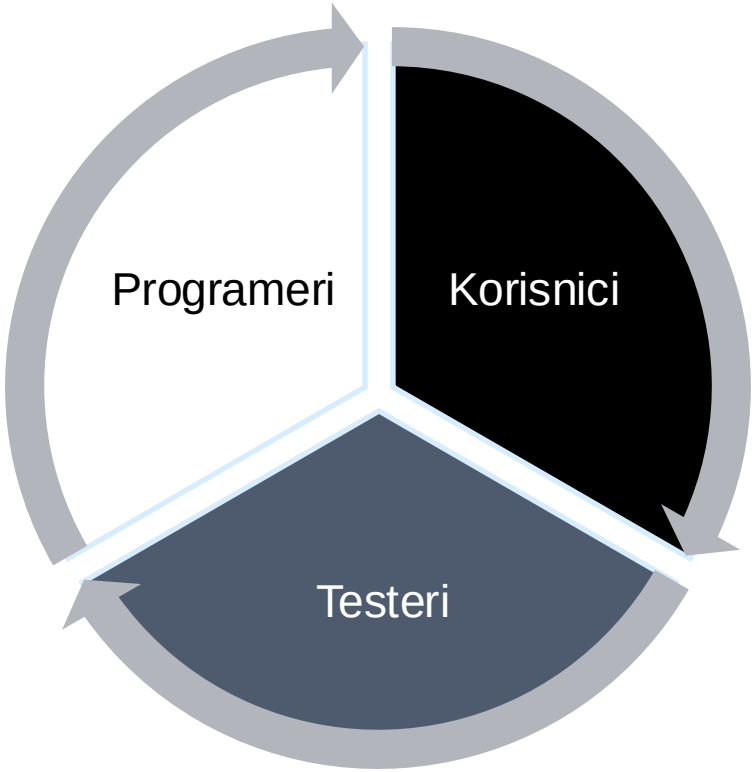
- **(4) Ispravljanje greške**
  - Poput bilo kojeg zadatka
  - Sastavni dio iteracije
  - Programiranje upravljeno testiranjem
  - Označavanje da je *riješeno* u aplikaciji za praćenje zadataka i grešaka
  - Statusi: Provjereno, Potvrđeno od strane korisnika
- **(5) Provjera je li greška riješena**
  - Osoba koja je originalno prijavila
  - Promjena statusa: Provjeren ili Aktivan

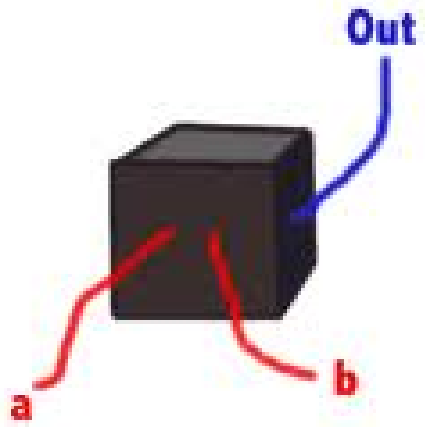

WHEN YOU FIND OUT THE
BUG YOU FIXED
CREATED A NEW BUG

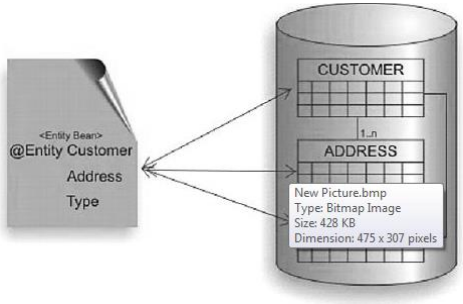
Tipovi testiranja

Tipovi testiranja









Tipovi testiranja

- Korisnici
  - Ne gledaju kod, bazu podataka
  - Ne analiziraju algoritam
  - Radi li ono što njima treba?
- Testeri
  - Gledaju radi li ono što treba
  - Ali, malo zavire ispod haube (log datoteke, baza, …)
    - Da li se zapisalo, izmijenilo, obrisalo očekivano
- Programeri
  - Analiza koda – nekonzistentnost, multipliciranje, …
  - Obrada grešaka

Tipovi testiranja

Korisničko testiranje

- Funkcionalnost
- Unos
- Izlaz
- Promjena stanja

Izvode testeri

- Bilježenje podataka
- Razmjena podataka
- Ispravnost kontrolnih podataka
- Čišćenje “smeća”

Programersko testiranje

- Sva izdanja koda
- Obrada grešaka
- Usklađenost dokumentacije i koda

Aplikacija kao crna kutija (1/4)

- **(1) Funkcionalnost**
  - Najvažnije
  - Nije važno kud se spremaju podaci (datoteke, baza, …)
  - Važno:
    - Je li moguće unijeti podatke na očekivani način
    - Vraćaju li se podaci na očekivani način (ekran, printer, …)
      - Točni? Ispravno prikazani?
  - NIJE :: *Da li se u kolekciju studenata može spremiti jedan ili više objekata klase Student*
  - JE :: Omogućuje li *aplikacija upis jednog ili više studenata u godinu*

Aplikacija kao crna kutija (2/4)

- (2) Kontrola unosa
  - Provjera da li polje prihvaća predviđeni tip podataka
  - Bez rušenja u slučaju neispravnog formata unosa
  - Upozorenje s objašnjenjem, bez brisanja ispravnih unosa
  - Primjeri:
    - Slova u datumsko polje
    - HTML kod u unos imena i prezimena
    - Iznos koji je prevelik ili premali
      - Korisnici uglavnom ne griješe puno, provjeriti
    - Vrijednost koja ne spada u rang vrijednosti
    - Izostavljen unos na mjestu na kojem je to potrebno
    - Međusobno povezani podaci
      - Regija + Grad

Aplikacija kao crna kutija (3/4)

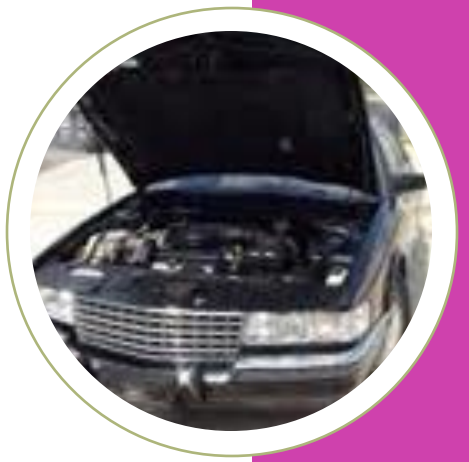
- **(3) Kontrola izlaznih informacija**
  - Je li nešto na temelju unesenih podataka ispravno izračunato?
  - Je li izvještaj točan?
  - Ručna provjera (ili korištenjem druge aplikacije)
  - Provjera izlaznih informacija u slučaju neispravnih podataka
    - Kod programera ovo dolazi zadnje na red, a važno je
    - Korisnici to prvo primijete

Aplikacija kao crna kutija (4/4)

- (4) Promjena stanja
  - Primjeri
    - Dokument
      - Kreiran, potvrđen, ovjeren/odbijen, dovršen
    - Godišnji odmor
      - Poslan na zahtjev, odobren, realiziran
    - Email poruka
      - Kreirana, spremljena radna kopija, poslana, primljena, obrisana
  - Treba provjeriti kretanje iz stanja u stanje

Aplikacija kao siva kutija (1/4)

- (1) Provjera bilježenja podataka
  - Bilježenje podataka
  - Bilježenje kontrolnih podataka kod kritičnih sustava
    - Oprez – ne smije biti dostupno svakome
    - Npr. Bankarski sustavi
    - Ovi podaci nisu dostupni “normalnom” korisniku preko korisničkog sučelja
    - Koriste se u slučaju razrješavanja problema
  - Što vide administratori?

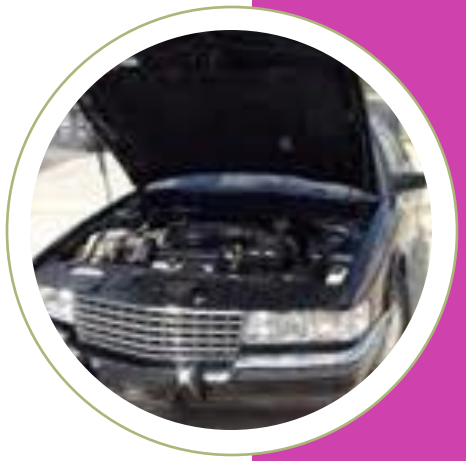


Aplikacija kao siva kutija (2/4)

- (2) Provjera podataka koje se šalje drugim sustavima / aplikacijama
  - Npr. podaci za sinkronizaciju poslovnog sustava i web shopa
  - Provjeravati:
    - **Strukturu** podataka
    - **Ispravnost** samih podataka
    - Način na koji sustavi razmjenjuju **pogreške**
    - Obrada **nedostupnosti** – u slučaju da dohvaćamo podatke iz druge aplikacije (servisa)

Aplikacija kao siva kutija (3/4)

- (3) Provjera dodatnih sistemskih podataka
  - Svrha: dodatne informacije za provjeru ispravnosti podatka
  - Primjer:
    - OIB – primjer podatka koji ima kontrolnu brojku
    - Zaporke: Hash/Solt
  - Provjeriti da li sve ispravno radi
    - Npr. Da li se zaporka nakon što je zapisana na ovaj način može provjeriti i koristiti u sustavu



Aplikacija kao siva kutija (4/4)

- (4) Čišćenje “smeća”
  - Često se događa da se prilikom brisanja podataka iz aplikacije ne pobrišu podaci iz baze
  - Provjera da li se podaci
    - Brišu kada trebaju
    - Ne brišu kada ne trebaju
      - **Problem s prostorom** – tijekom vremena
  - Deinstalacija aplikacije
    - Vraćanje u prvobitno stanje
      - Baza
      - Korisnici dodani prilikom instalacije
      - Mape, datoteke
      - Registry

Aplikacija kao bijela kutija (1/3)

- (1) Provjera svih izdanja aplikacije
  - Potrebno poznavanje koda
  - Postoji najčešće 1 grana s kojom programer češće radi
    - Samim time je ta grana bolje testirana
  - Testiranjem se mora proći po svim granama
    - OS, baza, izdanje, …

Aplikacija kao bijela kutija (2/3)

- (2) Obrada grešaka
  - Da li **postoji** obrada grešaka
  - Je li greška koju korisnik dobije u slučaju neispravnih podataka **ispravna i korisna**

Aplikacija kao bijela kutija (3/3)

- (3) Kod i dokumentacija su usklađeni
  - Ako u dokumentaciji piše da će metoda vraćati cjelobrojni podatak
    - Je li to zaista tako?
  - Važno u:
    - Samoj aplikaciji
    - Dijelu servisa koji se koriste za sinkronizaciju s drugim sustavima
  - Tko koristi dokumentaciju?
    - Novi članovi tima
    - Naši krajnji korisnici
    - Mi sami
    - Kolege iz drugih firmi (su-izvođači aplikacije)

Organizacija testiranja

Nastavak

Uvod

- Par dana prije isporuke
  - Implementirana tražena funkcionalnost
    - Provjeravano s korisnikom
  - Stalne provjere koda
  - Dnevni buildovi
  - Dijagram obavljanja posla

  - Što još treba napraviti?

… drugi puta ću pametnije …

- Uvijek postoji još stvari koje se mogu napraviti
- … da smo bar neke stvari napravili drugačije …
  - Nakon obavljena posla smo pametniji, iskusniji
  - Iterativni način - omogućava da učimo iz svake iteracije
    - Što možemo bolje sljedeći put?
      - Unapređenja u procesu
      - Unapređivanje koda
      - Izmjene u razvojnoj okolini
        - Novi alati
        - Nove tehnologije

Što još (ne) učiniti danas?

- Učiniti
  - Testiranje sustava
    - Mi testiramo redovito, a korisnik?
  - Ažuriranje dokumentacije
- Ne učiniti
  - Promjena koda u zadnji tren
    - Radi unapređenja koda
    - Super ideja u zadnji tren

Ako ostane vremena - priprema za sljedeću iteraciju

- Rješavanje greška
  - Provjeriti s korisnikom
- Priprema za neki zadatak iz sljedeće iteracije
  - Provjeriti s korisnikom, prioriteti su se možda promijenili
- Edukacija
  - Članovi tima
    - Pregled korištene tehnologije
  - Korisnici
    - Korištenje aplikacije
- Zabava

Pitanja

- Ostane li ikada viška vremena?
- Prvih par iteracija traju duže?
  - Procjene / usputni važni zadaci
- Što ako imamo previše vremena na kraju iteracije?
  - Jesu li procjene precijenjene? Zašto?

Testiranje sustava

- Testiranje koje je što sličnije pravom korištenju sustava
- Testiranje sustava u cijelosti (ne izoliranih dijelova / modula)
  - Testovi pojedine klase – test dijelova aplikacije
  - Potrebno provjeriti kako sve funkcionira kao cjelina
    - Iz korisničke perspektive – da li je sve izvedeno i radi
  - Da li je predviđen period za testiranje unutar iteracije:
    - Da - izvodimo mi: programeri, testeri
    - Ne – izvodi odabrani skup korisnika – beta testiranje

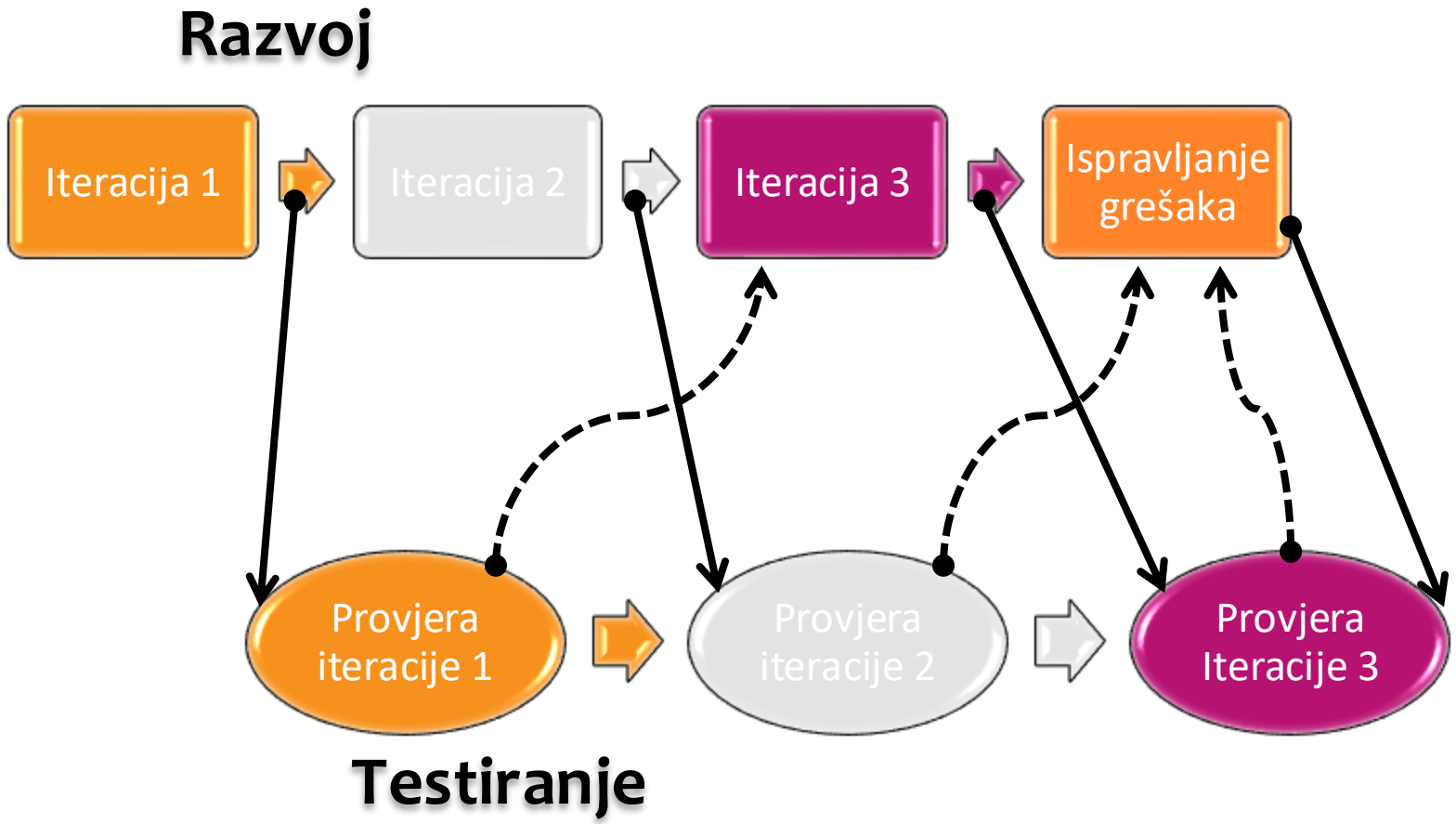
Tko izvodi testiranje sustava?

- Najbolje: nekoliko različitih ljudi
- Programer
  - Zna puno interno o izvedbi sustava
  - Lošija opcija - podsvjesno izbjegava zamke
  - Teško se prebacuje u mentalni sklop korisnika
  - **Nikako: programer koji testira svoj dio koda**
- Tester
  - Svjež pogled na aplikaciju kao crnu kutiju
  - Bolja opcija: Pokušavaju pronaći greške i nelogičnosti
  - Ne zanima ih tehnička savršenost, samo da li sustav radi

Kada je moguće testiranje sustava

- Testiranje dijelova – komponenti, klasa
  - Cijelo vrijeme, paralelno s razvojem
- Testiranje cijele aplikacije
  - Moguće tek na kraju pojedine iteracije
    - Gotove cjeline koje su pogodne za testiranje
    - Ranije nema smisla gubiti vrijeme - testiranje više puta
      - Unutar iteracije - moguće izmjene na kodu nakon što je on provjeren
    - Isplanirana iteracija – nema “prostora” za ispravljanje novih grešaka
  - Posao raste sa svakom iteracijom
    - Sve je veći postotak gotov

Usklađivanje razvoja i testiranja



Dodatni izazovi kod dužih projekata

- Više komunikacije
- Komunikacija unutar pojedinog tima i između dva tima
  - Razvojni tim – želi raditi na izradi nove funkcionalnosti
  - Tim testera – želi odgovore na pitanja o greškama
  - Mogućnost djelomičnog rješavanja
    - Predstavnik testera na sastancima razvojnog tima
      - Kao promatrač
      - Saznaje tekuće probleme
      - Kako postaviti okolinu za testiranje
      - Ne gubi puno vremena, jer su sastanci kratki

Testiranje u okviru jedne iteracije

- Testiranje se treba odvijati u unaprijed ograničenom vremenskom okviru – jedne iteracije
  - Ne mora nužno biti idealno (prekratko, predugo)
- Dobra praksa:
  - Tim testera daje procjenu koliko im treba za provjeru određenog korisničkog zahtjeva
- Dodatna poteškoća:
  - Tim testera radi na više od jednog projekta

Kombinacija razvoja i rješavanja grešaka

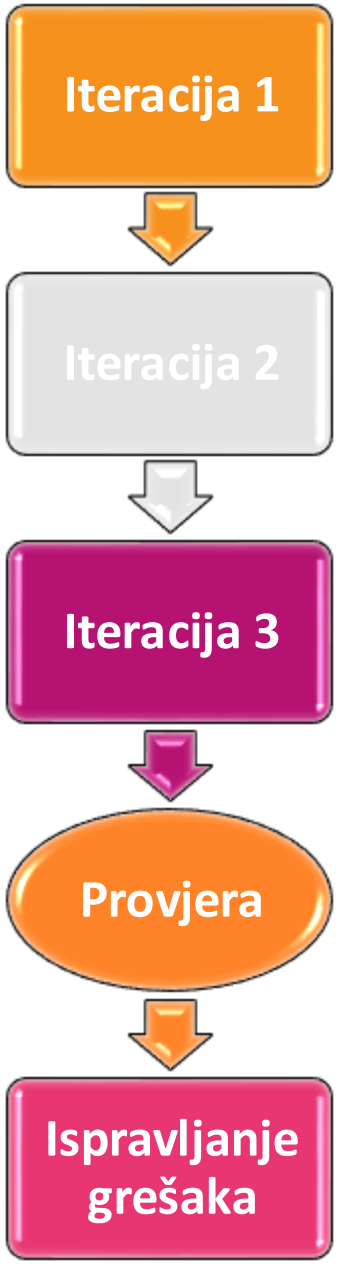
- Razvojni tim će dobiti prijavu grešaka iz 1. iteracije kada budu u 3. iteraciji
- Kako se ponašati prema greškama?
  - (1) Isto kao i prema novim funkcionalnostima
    - Procjena, prioritet
  - (2) Rezerviranje dijela vremena za rješavanje grešaka
    - Npr. 1 dan u tjednu
    - Ne zaboraviti - smanjiti raspoloživo vrijeme za razvoj npr. 4/5
- Ponekad greška velike važnosti, onemogućen daljnji tijek testiranja – rješavanje odmah

Promjene u zahtjevima

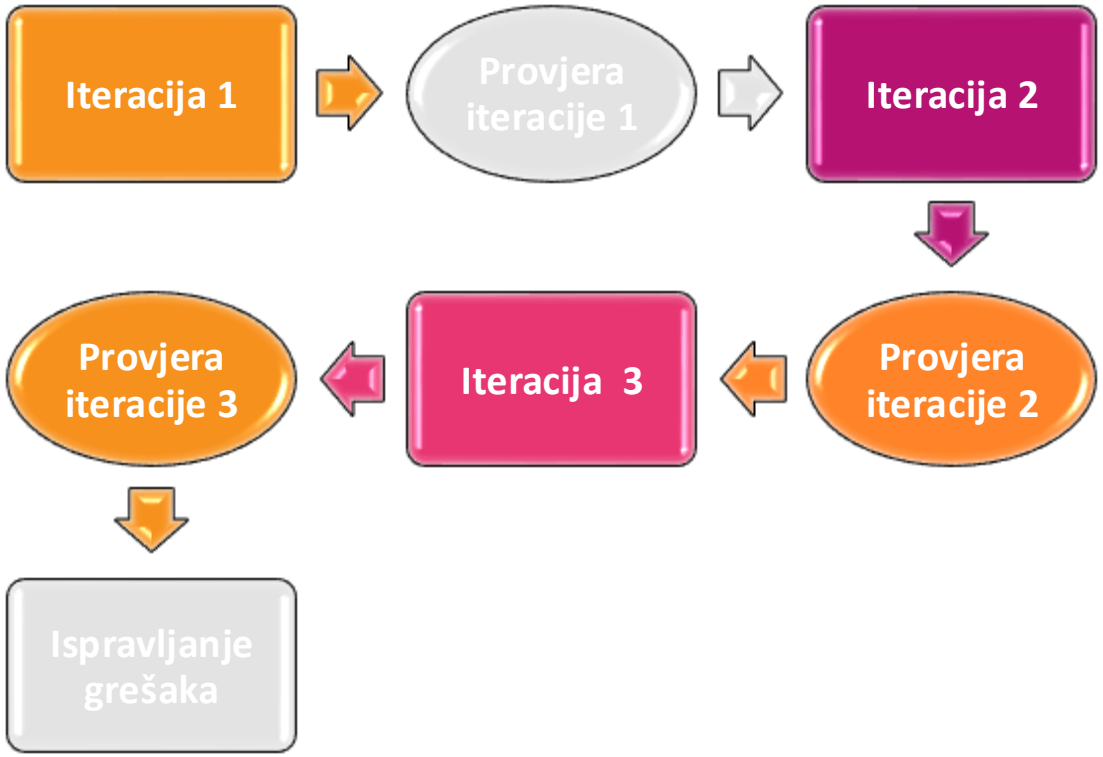
- Funkcionalnost opisana u dokumentu s korisničkim zahtjevima može se promijeniti
  - Ulaganje u pisanje testova može biti uzaludno, jer se za mjesec dana ta funkcionalnost može promijeniti
- Ne postoji neki *pametni recept* što učiniti u takvim situacijama osim komunicirati – obavijestiti tim testera o promjenama – što je ranije moguće
  - Formalni sastanci
- Svi trebaju biti svjesni da će do promjena sigurno doći
- Lakše je raditi ako znamo da su promjene dio posla

Što kada ...
Razvojni tim = Tim testera ?

- Pozitivnost
  - Dobar raspored s obzirom na raspoložive resurse
- Negativnost
  - Testiranje (i otkrivanje potencijalno ozbiljnih problema) tek na kraju
- Preporuka:
  - Dati iteraciju na testiranje užem krugu korisnika



Što kada ...
Razvojni tim = Tim testera ?



- Pozitivnost:
  - Povratna informacija stiže brzo
- Negativnost
  - Jako puno vremena odlazi na testiranje
  - Da li je to korisniku prihvatljivo?
    - Udaljeni rok
- Ponekad korisnik inzistira na ovakvom načinu rada