
PROGRAMMING
SOFTWARE
BUGS
TESTING
METHODS
DATABASE
USABILITY

KVALITET SOFTVERA

Mnogi autori u knjigama i naučnim časopisima tvrde sledeće: "U svetu se danas (godišnje) u softverske projekte uloži jako puno novčanih sredstava. (+ VREME)"

Iako se tehnologije svakim danom sve više i više razvijaju, odnosno alati za razvoj softvera se razvijaju svaki dan, i dalje je značajan broj grešaka i otkaza koji se javljaju kod gotovih softverskih proizvoda.

CILJ:

Otkriti što više grešaka u što kraćem roku.

Problem vremensko ograničenje i vremenski rokovi

POZITIVNI I NEGATIVNI TEST

POZITIVNI TEST

NEGATIVNI TEST

Pozitivni test

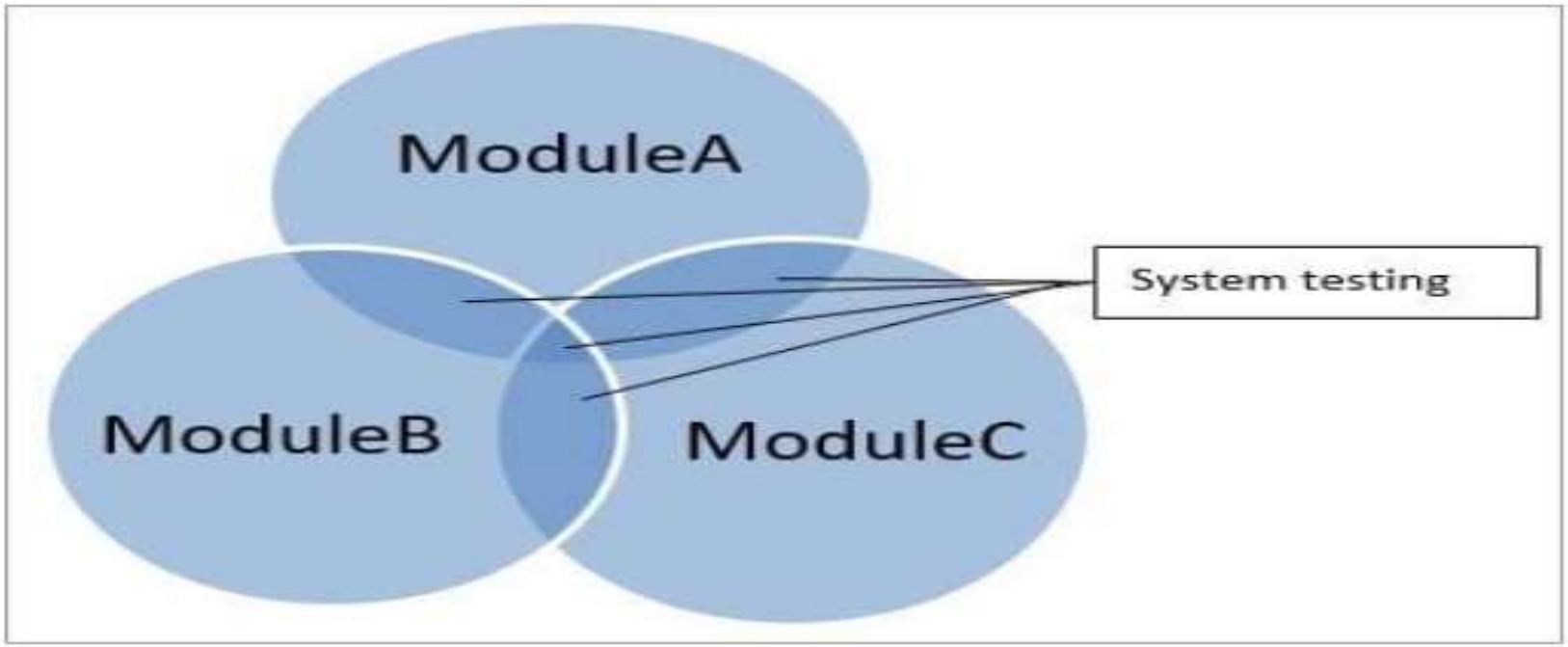
Kada korisnik pažljivo i tačno unosi podatke odnosno ispoštuje sva pravila i unese sve podatke na predviđeni način ovaj test proverava da li je rezultat u skladu sa očekivanim.

Jedan scenario može da ima desetine poslovnih pravila koja moraju da budu zadovoljena. Dobra praksa je testiranje što većeg broja test slučajeva, kako bi se na vreme identifikovali svi bagovi.

Negativni test

predstavlja alternativna scenarija u kojima je osnovna pretpostavka da neće sve da prođe kao što je očekivano.

SISTEMSKO TESTIRANJE SOFTVERA



TESTIRANJE SOFTVERA

Jedan od ideja kvaliteta proizvoda je težnja da taj proizvod bude izrađen u skladu sa specifikacijom (Crosby još 1979).

Međutim postoji nekoliko specifičnosti koje kvalitet softvera odvajaju od kvaliteta ostalih "klasičnih" proizvoda

- Specifikacija svakog proizvoda ide ka ispunjavanju želja i potreba korisnika.
- Međutim, kod razvoja softvera često postoje zahtevi koji nisu izričito specificirani (na primer zahtev za stabilnošću softvera).
- Nepostoji pouzdan način za kvantifikaciju pojedinih karakteristika softvera (na primer: stabilnost, pouzdanost).
- Softverska specifikacija je često nekompletna.

KVALITET SOFTVERA

Kvalitet softvera je više dimenzionalni koncept koji se ne može jednostavno definisati.

Potrebno je pratiti različite parametre i obezbediti kvalitet softvera, kreirati planove vezane za kvalitet i ostvariti potrebnu kontrolu.

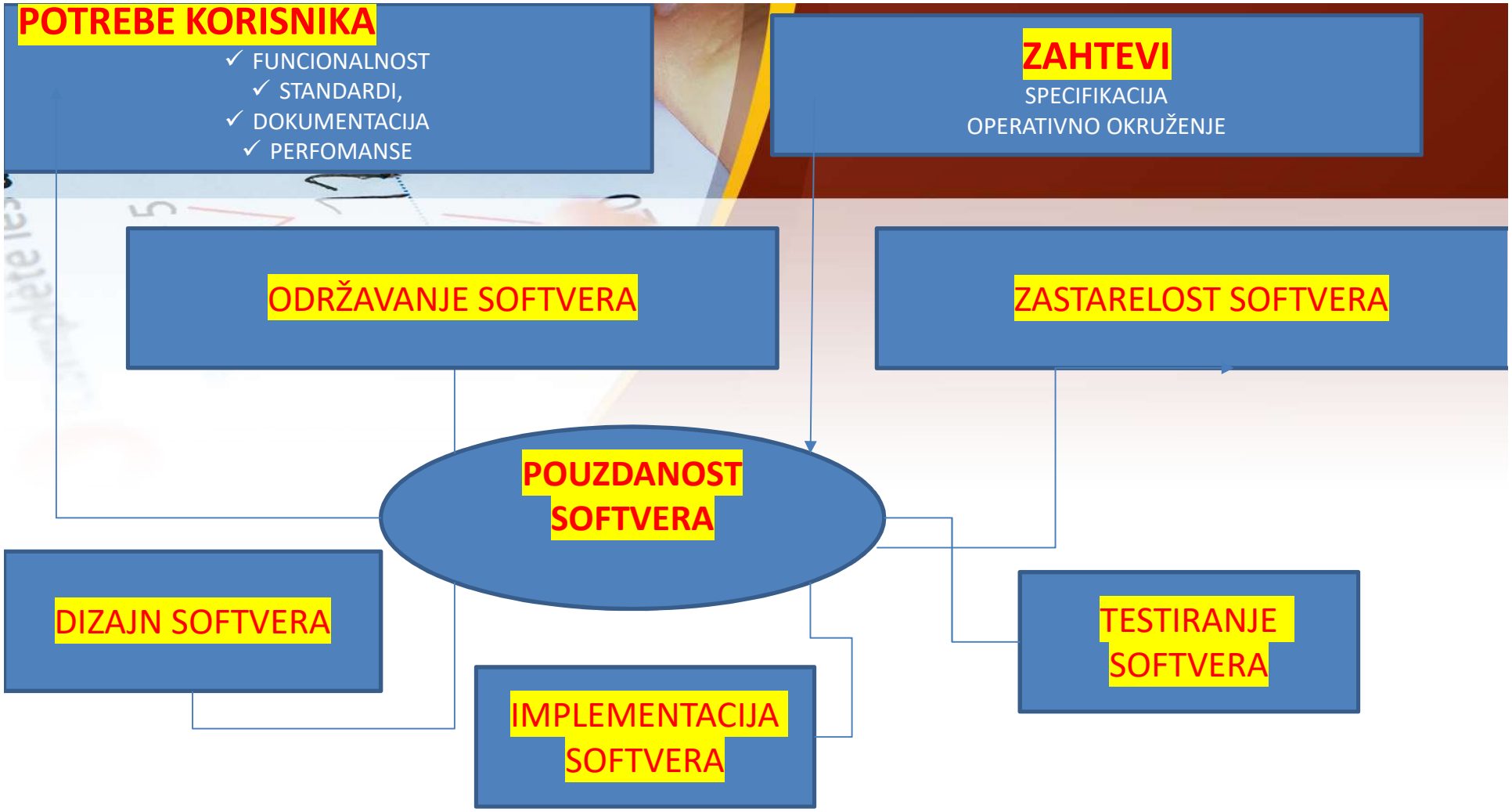
Potrebno je razviti i implementirati standarde sistema i dokumentaciju sistema kvaliteta kvaliteta koja će biti primenjena na softverske proizvode.

KVALITET SOFTVERA

U principu jedna od ideja kvalitetnog proizvoda je težnja da taj proizvod bude izrađen u skladu sa specifikacijom (postavljen od Crosby 1979).

Pod kvalitetom u širem smislu podrazumevamo skup svih osobina jednog proizvoda koje zadovoljavaju propisane uslove u pogledu:

- namene proizvoda
- njegove funkcije i pouzdanosti
- jednostavnosti i ekonomičnosti u upotrebi
- održavanja


POTREBE KORISNIKA
✓ FUNCIONALNOST
✓ STANDARDI,
✓ DOKUMENTACIJA
✓ PERFOMANSE
ZAHTEVI
SPECIFIKACIJA
OPERATIVNO OKRUŽENJE
ODRŽAVANJE SOFTVERA
ZASTARELOST SOFTVERA
POUZDANOST SOFTVERA
DIZAJN SOFTVERA
TESTIRANJE SOFTVERA
IMPLEMENTACIJA SOFTVERA

POUZDANOST

- ✓ Mere konačnog proizvoda softvera govori o kompleksnosti softvera.
- ✓ Mere upravljanja projektom
- ✓ Mere procesa izrade softvera
- ✓ Mere potencijalnih grešaka i otkaza.
- ✓ Mere zahteva za pouzdanost

KVALITET SOFTVERA

Među važne osobine kvaliteta softvera su:

- funkcionalnost – u kojoj meri softver zadovoljava specifikacije i potrebe korisnika
- pouzdanost – frekvencija i težina otkaza, sposobnost oporavka, predvidivost, tačnost, srednje vreme između otkaza,...
- upotrebljivost – meri se naporom koji zahteva učenje programa, njegovo korišćenje, priprema ulaza i interpretacija izlaznih rezultata
- efikasnost – brzina odziva, raspoloživost, brzina oporavka, iskorišćenje resursa
- stepen podrške – mogućnost testiranja, proširivanja, prilagođavanja, konfigurisanja, instaliranja.

KVALITET SOFTVERA

Provera kvaliteta softvera pruža brojna rešenja ključnim uzrocima problema razvoja softvera:

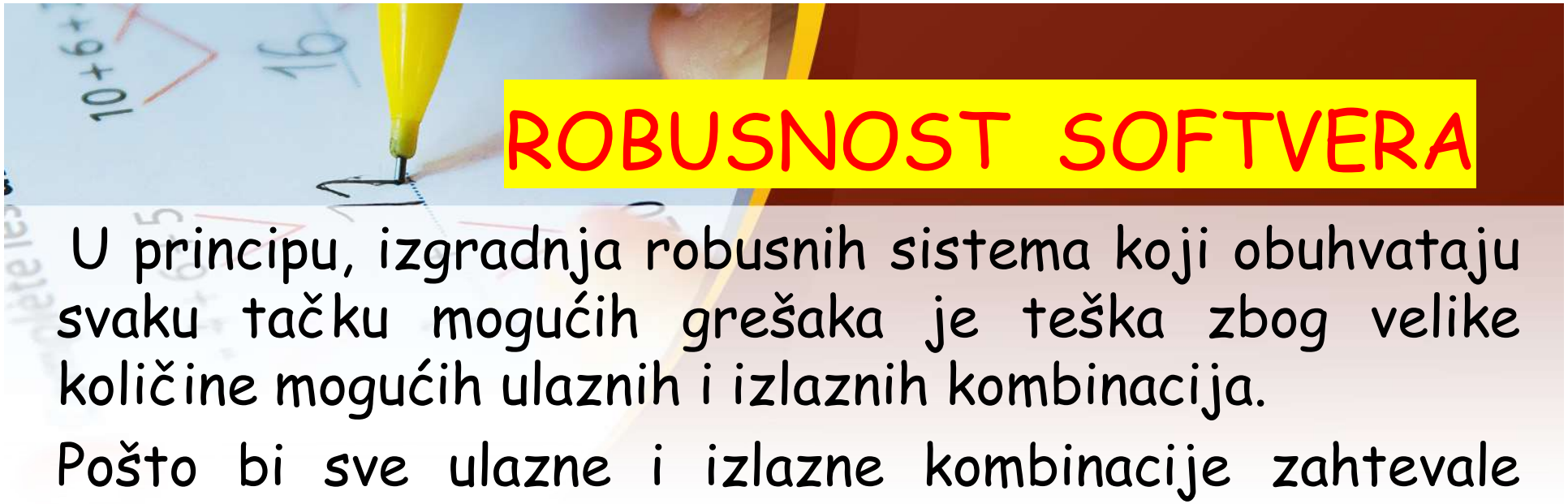
- Procena statusa razvojnog projekta se izrađuje objektivno, ne subjektivno (vrednuju rezultati testa)
- Objektivno procenjivanje otkriva nekonzistentnosti u zahtevima, dizajnu i implementaciji
- Testiranje i verifikacija su usmereni na oblasti najvišeg rizika
- Greške se otkrivaju rano, (smanjuju se troškovi njihovog ispravljanja)

ROBUSNOST SOFTVERA

Pojam robusnost (u računarstvu) je sposobnost računarskog sistema da se nosi sa greškama tokom izvršavanja i da se nosi sa pogrešnim ulazima.

Formalne tehnike, kao sto je faz-testiranje, su od suštinskog značaja za pokazivanje robusnosti jer ova vrsta testiranja uključuje nevažeće ili neočekivane ulaze.

Razni komercijalni proizvodi vrše testiranje softverskih sistema koristeći robusnost i obično je sastavni deo analize procene neuspeha.



ROBUSNOST SOFTVERA

U principu, izgradnja robusnih sistema koji obuhvataju svaku tačku mogućih grešaka je teška zbog velike količine mogućih ulaznih i izlaznih kombinacija.

Pošto bi sve ulazne i izlazne kombinacije zahtevale previše vremena za testiranje, programeri ne mogu da prolaze kroz sve slučajeve detaljno.

Umesto toga, programer će pokušati da uopšti takve slučajeve.


Request for Praposal
Start
Requirements
Analysis
Coding
Design
Deploy
End
Testing
Web Client
Database Server
Web Server

Ocena kvaliteta realizacije softvera

Softver se podvrgava rigoroznim testovima pre izlaska na tržište. Treba da se vodi računa i :

- Da se istakne razlike između prvobitnog koncepta i konačnog izlaza
- Da se verifikujte da softver funkcioniše onako kako su projektanti planirali, problem specifikacije
- Da se validira krajnji proizvod - proizvod mora zadovoljiti zahteve korisnika
- Da se procene karakteristike i kvalitet samog sofrvera

Ocena kvaliteta realizacije softvera

Svi nedostaci se ispravljaju, a softver ide kroz testiranje regresije - sistem za proveru da li program i dalje funkcioniše nakon modifikacija.

Izveštaj i dobra dokumentacija su jako bitni!

Kvalitet je nepostojanje nedostataka. (Juran 1988)

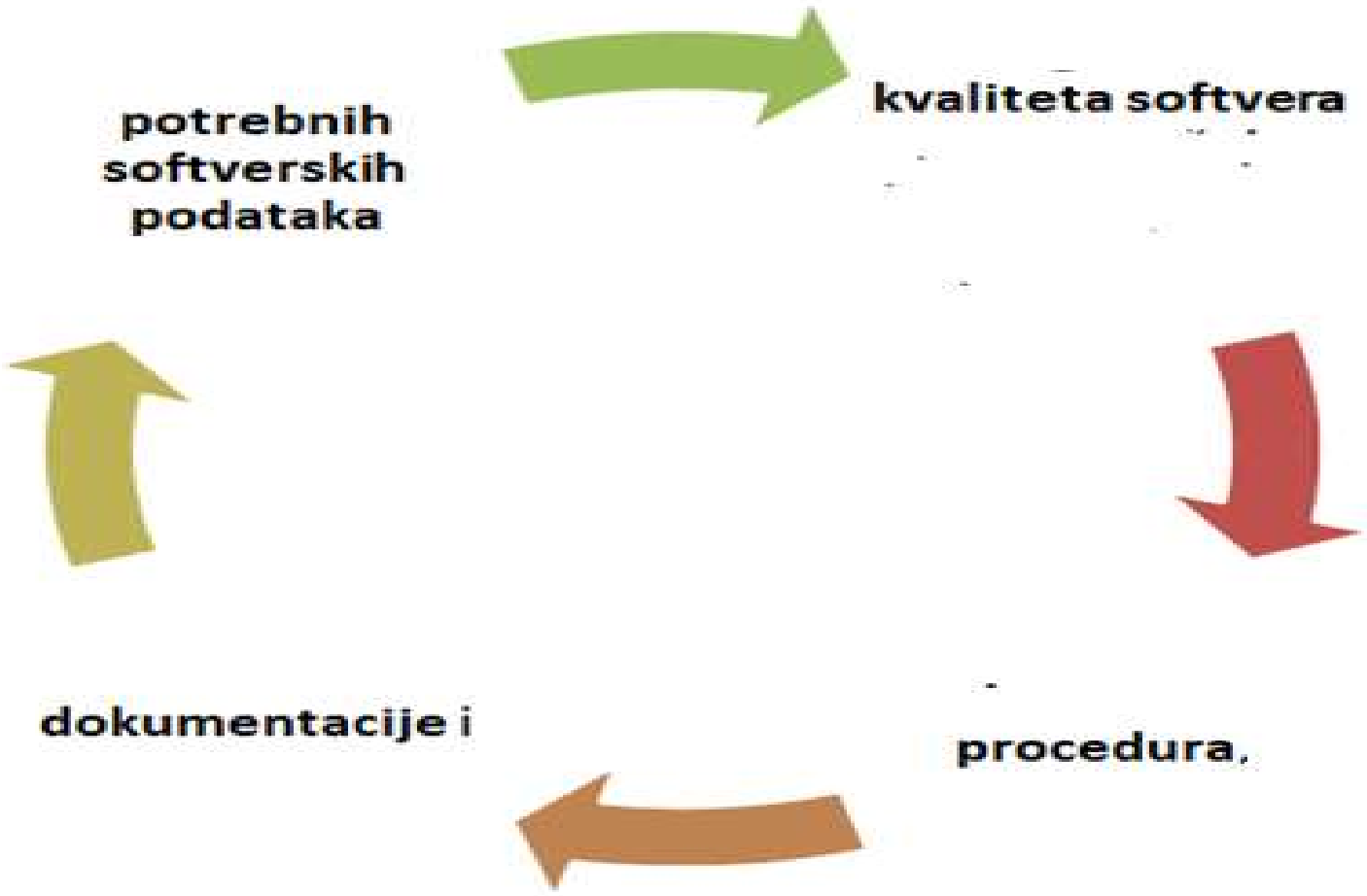
Kvalitet softvera

Cilj – kvalitetan softver.

Definicija preporučena od strane IEEE 610.12 (IEEE Standard Glossary of Software Engineering Terminology) je:

1. **Stepen u kojem sistem**, komponente, ili procesi zadovoljavaju **specifične zahteve**.
2. **Stepen u kojem sistem**, komponente ili procesi zadovoljavaju **klijentove ili korisnikove potrebe ili očekivanja.**

!


potrebnih
softverskih
podataka
kvaliteta softvera
procedura,
dokumentacije i

TESTIRANJE SOFTVERA

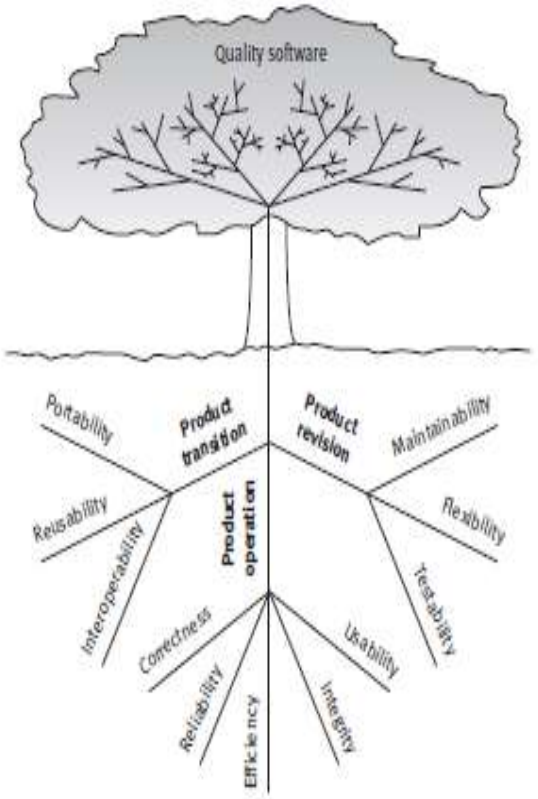
McConnell u knjizi CodeComplete

McConnell u knjizi CodeComplete, istakao **11 faktora** za kvalitetan softver o on ih je **grupisao u 3 kategorije**:

-**Faktori proizvoda (softvera)**: korektnost (correctness), pouzdanost (reliability), efikasnost (efficiency), celovitost (Integrity), iskoristivost (usability).

-**Faktori revizije proizvoda (softvera)**: održavanje (maintainability), fleksibilnost (flexibility), pogodan za testiranje (testability).

-**Faktori tranzicije proizvoda (softvera)**: : portabilnost (portability), ponovna iskorišnjenost (reusability), interoperabilnost (interoperability).



Principi testiranja softvera

Neki važniji principi testiranja softvera su:

- ✓ Svi testovi se moraju pratiti do projektnih zahteva
- ✓ Testovi se moraju planirati znatno pre samog testiranja
- ✓ Važi Pareto princip (80:20 - 80% efekata potiče od 20% uzroka)

80% svih grešaka koje se ne otkriju prilikom testiranja verovatno se odnosi na svega 20% programskih komponenti

- ✓ Testiranje počinje od malih delova i napreduje ka celini
- ✓ Iscrpno testiranje nije moguće (složenost)
- ✓ Testiranje je efikasno ako ga vrši neko drugi (a ne testerski tim i programeri koji su radili ma projektu)
- ✓ Resursi za testiranje su ograničeni (npr. vreme)
- ✓ U toku testiranja softver ne treba menjati

Testiranje upotrebljivosti (Usability testing)

- Specifični korisnici - nisu bilo koji korisnici već korisnici za koje je softver dizajniran.
- Specifični ciljevi - specifični korisnici imaju svoje specifične ciljeve.
- Specifični kontekst korištenja – softver je dizajniran da radi u okruženju u kojem će ga korisnici koristiti.

GLAVNE PREDNOSTI TESTIRANJA UPOTREBLJIVOSTI

- Povećanje zadovoljstva korisnika
- Povećanje broja korisnika
- Poboljšanje profitabilnosti – upotrebljiviji proizvodi stvaraju zadovoljne korisnike
- Poboljšanje dizajna interfejsa

TESTIRANJE SOFTVERA

Testiranje softvera može se posmatrati kao jedan od metoda koji se koriste za kontrolu kvaliteta softvera.

- BITAN JE : KVALITET SOFTVERA

Testiranje omogućava da se tokom razvoja i pisanja testova veoma rano identifikuju problemi u arhitekturi softverskog rešenja, funkcionalnoj specifikaciji,….

Mnogi autori u knjigama i naučnim časopisima tvrde sledeće:

"Svrha testiranja softvera može biti verifikacija ili validacija".


Verifikacija i Validacija
VERIFICATION
Am I building the product right?
Unit Test
Integration Test
Automated Testing
Regression Testing
System Test
Beta Test
Customer Acceptance Test
Usability Test
VALIDATION
Am I building the right product?

Verifikacija i Validacija

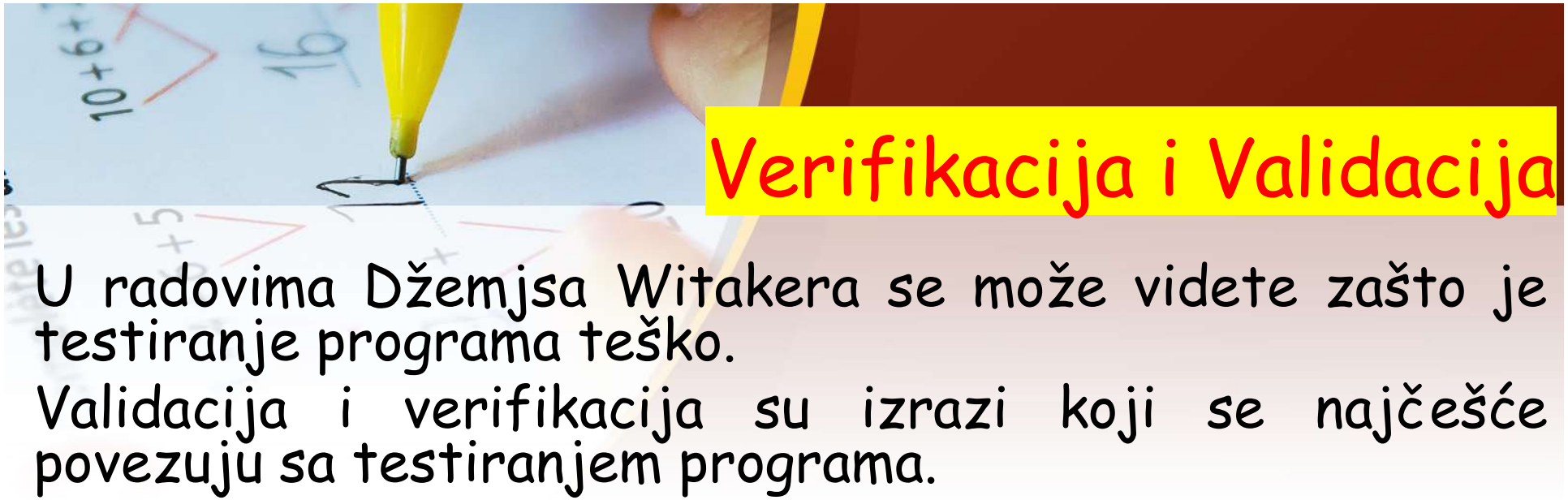
- Verifikacija softvera – Da li dobro razvijamo softver?
- Validacija softvera – Da li razvijamo dobar softver?
- Verifikacija daje odgovor na pitanje ("Are we building the product right?").

– Barry Boehm, 1981

Verifikacija i Validacija

U literaturi se može videti i ovakva oznaka (V&V).

- Testiranjem se vrši
  - **validacija** softverskog proizvoda
    - ❑utvrđuje se da li je realizovani proizvod pogodan za namenjenu svrhu
    - ❑vodi računa o korisnikovom pogledu na aplikaciju
  - **verifikacija** softverskog proizvoda
    - ✓na nivou pojedinačne faze utvrđuje da li je rezultat realizacije određene faze u skladu sa specifikacijom za tu fazu
    - ✓više je fokusirana na tehničku realizaciju



Verifikacija i Validacija

U radovima Džemjsa Witakera se može videte zašto je testiranje programa teško.

Validacija i verifikacija su izrazi koji se najčešće povezuju sa testiranjem programa.

Ove tehnike otelotvoruju principe deduktivnog zaključivanja, iste one koje koriste i programeri prilikom samog konstruisanja programa.

Zašto ne bi koristili iste principe u sistemu za automatsku sintezu, koji može da konstruiše program umesto da samo dokazuje njegovu ispravnost?

Verifikacija i Validacija

Ova dva slična pojma se zapravo dosta razlikuju Verifikacija softvera je izuzetno važna oblast računarstva.

Neispravan softver može imati katastrofalne posledice, koje uključuju i izgubljene ljudske živote

- Kako bismo izbegli ovakve situacije, neophodno je precizno utvrditi ispravnost razvijenog softvera

Verifikacija i Validacija

- **Validacija**: proces evaluacije sistema ili komponente za vreme ili na kraju procesa razvoja da bi se utvrdilo da li zadovoljava zahteve definisane od korisnika

Da li kreiramo **pravi proizvod**?

- **Verifikacija**: proces evaluacije sistema ili komponente kako bi se utvrdilo da li proizvod korektno implementira određenu funkciju
- Da li kreiramo proizvod na **pravi način**?

Verifikacija i Validacija

Šta se podrazumeva pod ispravnim softverom?

Razlikuju se pojmovi potpuno ispravnog i delimično ispravnog softvera.

Ne postoji algoritamski način da se proveri da li se neki program zaustavlja (halting problem), pa se često ni ne ispituje potpuna, već samo delimična ispravnost softvera

Verifikacija i Validacija

- Ukoliko je softver validan i verifikovan, to ne znači da on ne može imati grešku, već da je pouzdan i dobar za ono za šta je namenjen.
- Neretko softver testiramo za ograničenim skupom podataka
- Niko ne može garantovati da ne možemo pronaći test podatke na kojima se može desiti da naš program ne radi.
- SUGESTIJA:

Sprovesti testiranje sistema koristeći samo reprezentativni skup ulaznih podataka.

Verifikacija i Validacija

Verifikacija i validacija se provode kroz celi životni ciklus softvera, bez obzira na njegovu kompleksnost, veličini i sl.

Proces verifikacije i validacije se sastoji od tehnika kontrole softvera i testiranja softvera.

Kontrola i testiranje softvera su komplementarne tehnike.

Verifikacija i Validacija

Dinamička verifikacija softvera obuhvata tehnike ispitivanja ispravnosti koda u toku njegovog izvršavanja.

Najčešći vid verifikacije softvera je

TESTIRANJE.

Testiranje se često koristi kao sinonim za verifikaciju softvera

Verifikacija i Validacija

Verifikacija i validacija se izvode u svakoj od faza životnog ciklusa razvoja.

Treba da se koriste u procesu testiranja softvera.

Verifikacija obuhvata sistematične procedure pregleda, analize i testiranja ugrađene u životni ciklus, počevši od faze softverskih zahteva, do faze kodiranja.

Verifikacija obezbeduje kvalitet i održavanje softvera.

Koncept verifikacije uključuje dva kriterijuma: softver mora adekvatno i korektno izvršavati sve funkcije i ne sme izvršavati one funkcije koje sam po sebi ili u kombinaciji sa ostalim funkcijama mogu loše da utiču na performanse čitavog sistema.

Verifikacija i Validacija

Verifikacija	Validacija
Da li je softver napravljen prema navedenim zahtevima i specifikacijama dizajna.	Konačni softver kako bi proverio da li on zadovoljava krajnje korisnike
Postoje specifične zahtevi u određenim fazama razvoja softvera	Implementiran je softver.
Statičko testiranje	Dinamičkog testiranje.
Da li softver pravimo ispravno	Da li pravimo pravi softver

Osnovni cilj Verifikacija i Validacija

je da je sistem u potpunosti ispunjava svoju namenu.

Ova dva termina se takođe nazivaju kontrolom kvaliteta softvera koju koriste testeri softvera u životnom ciklusu razvoja softvera. Iako i izgledaju i zvuče slično, razlikuju se u analizi.

Očekivanja korisnika – korisnici mogu imati očekivanja za određenu vrstu softvera.

I na kraju postoji i:

Plasiranje proizvoda na tržište ranije može biti važnije od pronalaženja defekata u softveru. (Procena raznih faktora)

Verifikacija i Validacija

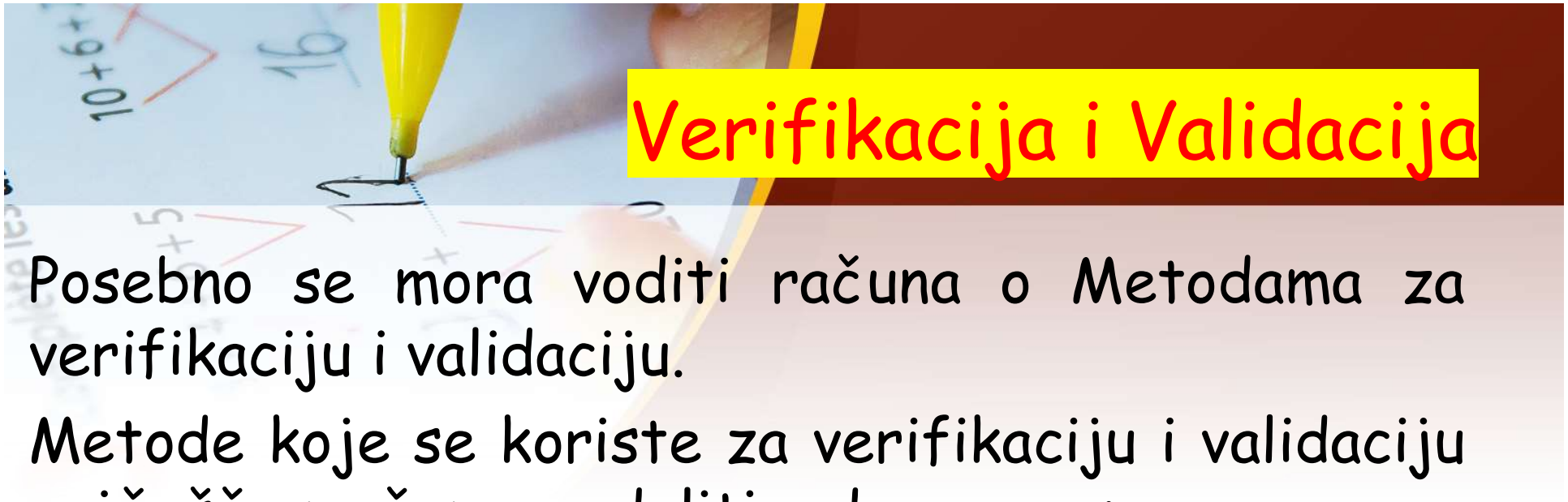
- Ciljevi testiranja:

Direktni ciljevi

- Otkrivanje i smanjenje broja grešaka
- *Povećanje kvaliteta testiranog softvera*
- *Izvršenje planiranih testova uz efikasno raspolaganje resursima – (vreme,ljudi)*

- **Indirektni ciljevi**

- Statistika o vrstama grešaka u cilju preventivnih aktivnosti za otkrivanje grešaka

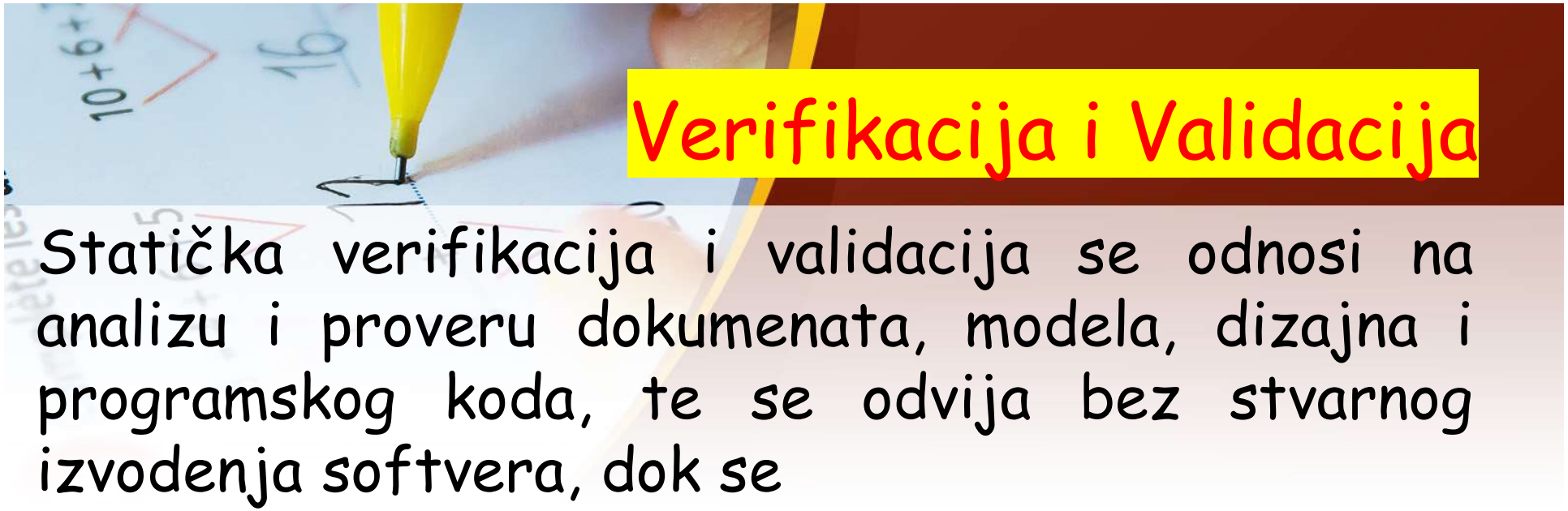


Verifikacija i Validacija

Posebno se mora voditi računa o Metodama za verifikaciju i validaciju.

Metode koje se koriste za verifikaciju i validaciju najčešće možemo podeliti u dve grupe:

- Statička verifikacija i validacija
- Testiranje softvera



Verifikacija i Validacija

Statička verifikacija i validacija se odnosi na analizu i proveru dokumenata, modela, dizajna i programskog koda, te se odvija bez stvarnog izvodenja softvera, dok se

Testiranje svodi na eksperimentalno izvođenje softvera ili delova softvera, sa test skupom podataka uz detaljno analiziranje rezultata.

Statička verifikacija i validacija

- ✓ Jedna statička provera može otkriti puno grešaka, dok testiranje može da otkrije samo jednu grešku i onda se može se desiti da nakon samo te jedne greške dolazi do pada softvera (aplikacije).
- ✓ Kod statičke metode dolazi do izražaja znanje o programiranju budući da se osobe koje obavljalju proveru oslanjaju se na iskustvo i greške koje se najčešće se pojavljuju i na njih se oni fokusiraju.

Statička verifikacija i validacija

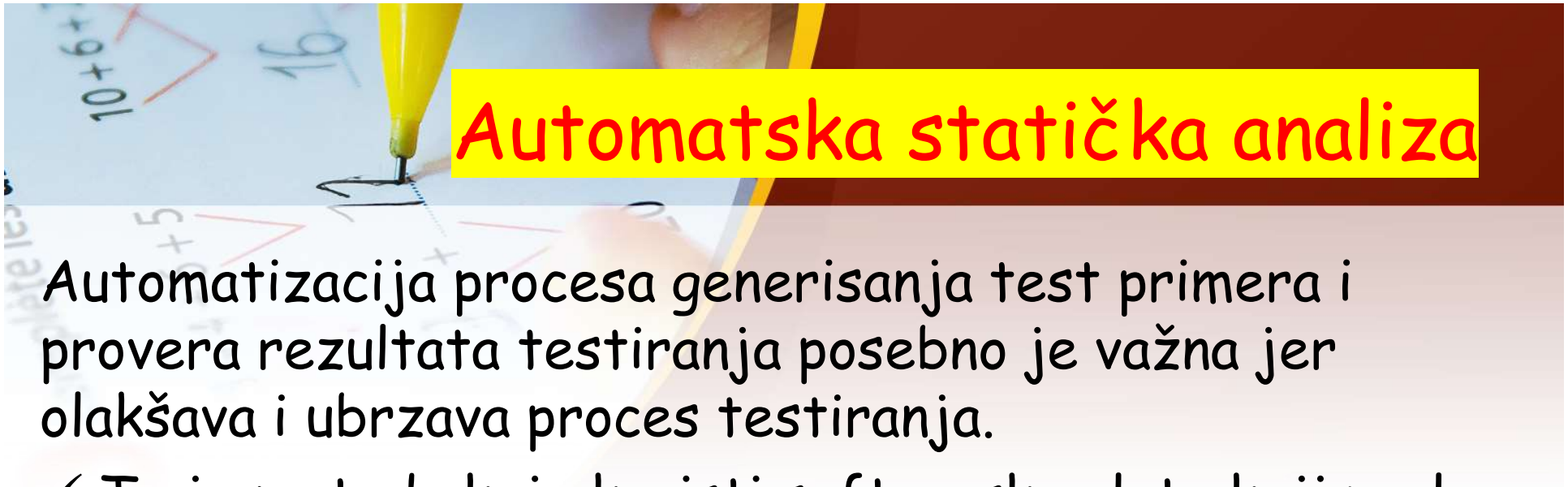
Neke od bitnijih tehnika statičke verifikacije su:

Analiza toka podataka(eng.data flow analysis)

Apstraktna interpretacija (abstract interpretation)

Simbolička analiza(Symbolic analysis)

Proveravanje ograničenih modela primer (Bounded model checking), koja se najviše koristi za za verifikaciju logičkih kola.



Automatska statička analiza

Automatizacija procesa generisanja test primera i provera rezultata testiranja posebno je važna jer olakšava i ubrzava proces testiranja.

- ✓ To je metoda koja koristi softverske alate koji prolaze kroz tekst koda i otkrivaju moguće greške i anomalije
- ✓ U okviru statičke automatizovane provere ispravnosti softvera formiraju se uslovi ispravnosti iskazani formulama u terminima matematičkih teorija

Formalna verifikacija

Idejom formalne verifikacije bavila su se mnoga velika imena naučnika 70-tih godina 20. veka (Hoare, Dijkstra, Wirth).

Formalna verifikacija - danas je :

- ✓ Predmet intenzivnog istraživanja.
- ✓ Polako postiže sve bolje rezultate.

Postoji varijanta formalne verifikacije koja se zove provera modela (model checking, koja se dosta koristi).

Formalna verifikacija

Formalna verifikacija svodi se na matematičko dokazivanje konzistentnosti programa sa svojom specifikacijom.

Ovu metodu moguće je koristiti samo kada je :

- ❑ semantika programskog jezika je formalno definisana
- ❑ postoji specifikacija za dati program/projekat

Oni koji je koriste tvrde da ona vodi do pouzdanijeg i sigurnijeg softvera.

Dokaz korektnosti programa je velik i složen postupak , pa u njemu mogu da postoje i greške.

Dinamička verifikacija softvera

zasnovana je na tome da se provera ispravnosti softvera vrši tokom njegovog izvršavanja.

- Najčešće se dinamička verifikacija softvera vrši testiranjem i ti pojmovi se poistovećuju, što nije sasvim ispravno.

Testiranje je složen proces koji obuhvata pronalaženje što raznovrsnijeg skupa ulaza, definisanje očekivanih izlaza za svaki od tih ulaza, a zatim izvršavanje programa i provera da li je program za date ulaze vratio odgovarajuće izlaze

Metodologije razvoja softvera

U okviru razvoja softvera, cilj je da se maksimizuje profit pravljenjem prozvoda visokog kvaliteta ali u vremenskim i budžetskim granicama.

Najzastupljenije i trenutno najpopularnije metodologije razvoja softvera promovišu paralelnu implementaciju i pisanje testova za svaku od celina koja se razvija u okviru softverskog sistema

Metodologije razvoja softvera

Ciljevi testiranja softvera su :

- Verifikacija i validacija, kojima se proverava da li je softver saglasan sa specifikacijom zahteva
- Što ne bi značilo uvek da je softver tehnički ispravan, pouzdan i bezbedan.
- Poboljšanje kvaliteta softvera.
- Procena pouzdanosti.

Aktivnosti testiranja SOFTVERA

- Planiranje
- ✓ Vremenski raspored procesa testiranja i testnih aktivnosti
- ✓ Objekat testiranja i zahtevi
- ✓ Strategija testiranja i prateći alati
- ✓ Dizajn i specifikacija okruženja
- ✓ Testiranje okruženja
- ✓ primeniti metode za dizajn testiranja na bazi dizajna softvera i zahteva
- ✓ Specifikacija testnih slučajeva sa ciljem postizanja pokrivenosti testiranja.
- ✓ Instalacija testnog okruženja
- ✓ Instalacija alata i testnog okruženja

TESTIRANJE SOFTVERA

Faza analize zahteva

Cena greške = vreme potrebno da se utvrde i zapiišu novi zahtevi

Faza kodiranja

Cena greške = dodatno vreme programera.

Vreme može da varira u zavisnosti od kompleksnosti greške, ali je značajno manje nego kada se ispravlja greška koju pronađe neko drugi.

Kada programer pronade sam svoju grešku, on obično razume problem i zna kako da ga reši.

Faze testiranja sofvera

- Razvojno testiranje – sistem se testira u toku razvoja da bi se otkrili bagovi ili drugi defekti.
- Testiranje proizvoda (release testing) – poseban tim testira kompletnu verziju aplikacije pre nego što se ona isporuči korisnicima.
- Korisničko testiranje (user testing) – korisnici ili potencijalni korisnici sistema testiraju sistem u zadatom (sopstvenom) okruženju.

Razvojno testiranje

Obuhvata sve aktivnosti testiranja koje izvršava tim za razvoj softvera.

- ✓ Testiranje jedinica (Unit testing)
- ✓ Testiranje komponenti (Component testing) – nekoliko posebnih jedinica se integrišu kako bi kreirale složene komponentejedinica Testiranje komponenti se fokusira na testiranje interfejsa komponenti.
- ✓ Testiranje sistema

Testiranje komponenti -Component testing

Prvo se izvodi jedan test. Kao što i samo ime govori, ovo je metoda testiranja na nivou objekta.
Pojedinačne softverske komponente testiraju se na greške.

Ovaj test zahteva tačno poznavanje programa i svakog instaliranog modula.

Na ovaj način ovu verifikaciju vrše programeri, a ne testeri.

U tu svrhu se kreiraju test kodovi koji proveravaju da li se softver ponaša kako je predviđeno

OSNOVNE faze testiranja softvera

Planiranje definiše:

- Vrste testova koje će biti sprovedene
- Opseg testiranja
- Izbor strategije i metode testiranja
- Koji je kriterijum završetka testiranja
- Koji su potrebni resursi i
- Kakav je način komunikacije izmedu članova tima

Strategije testiranja

Testiranje po delovima, gde se identifikuju grupe ulaza koje imaju zajedničke karakteristike i koje treba uraditi na isti način.

- Treba izabrati testove iz svake grupe
- Testiranje po zadatim procedurama i uputstvima
- Izbor testova.
- Uputstva predstavljaju predhodna iskustva o vrstama grešaka koje programeri prave prilikom razvoja komponenti.

Evaluacija testova

Testiranje se obično završava kada je softver isporučen korisniku, mada se može se desiti i u nekim drugim situacijama, na primer, kada je projekat otkazan ili je neki cilj postignut.

TESTIRANJE SOFTVERA SE NIKAD NE ZAVRŠAVA

Tokom ove faze, test skriptovi i dokumentacija se arhiviraju, dok se primenjeni proces testiranja analizira i diskutuje o tome šta je bilo dobro, a šta ne.

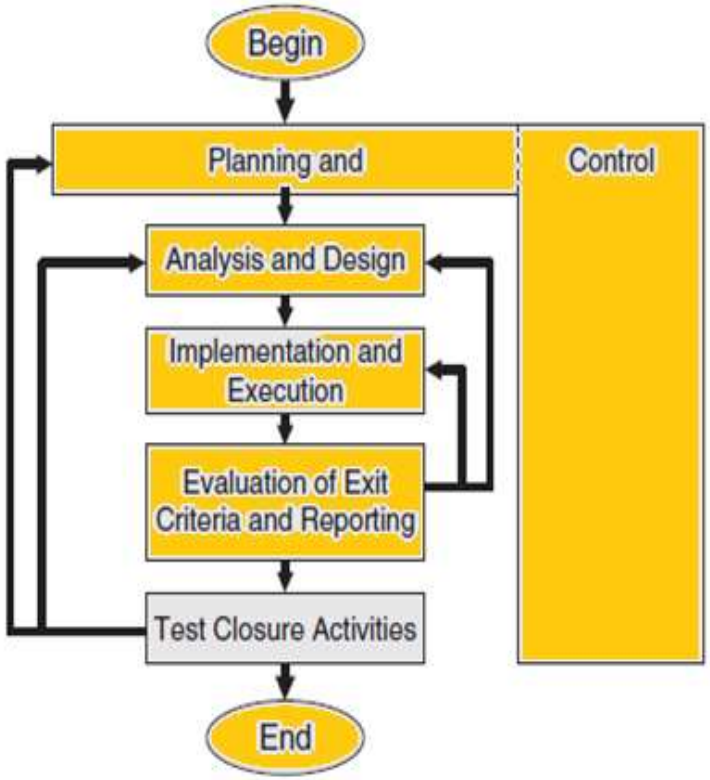
Ciklus testiranja

Za svaki nivo testiranja prolazi se kroz kompletan ciklus testiranja softvera.

Nakon svakog ciklusa potrebna je izmena u kodu. (ako postoje greške)

Nakon izmena potrebno je ponoviti testove.

Upravljanje testiranjem vodi se računa o inicijalizaciji, nadzoru i upravljanju ovim ciklusom



Kategorije rizika

- ✓ Loša procena vremena
- Problem sa specifikacijom
  - Nemogućnost pravilne procene funkcionalnosti i vremena potrebnog za razvoj funkcionalnosti.
  - Korišćenje resursa se ne nadgleda pravilno.
  - Neočekivana proširenja obima projekta sa dodatnim zahtevima i izmenom specifikacije uz nemogućnost dobijanja dodatnog vremena

Operativni rizici:

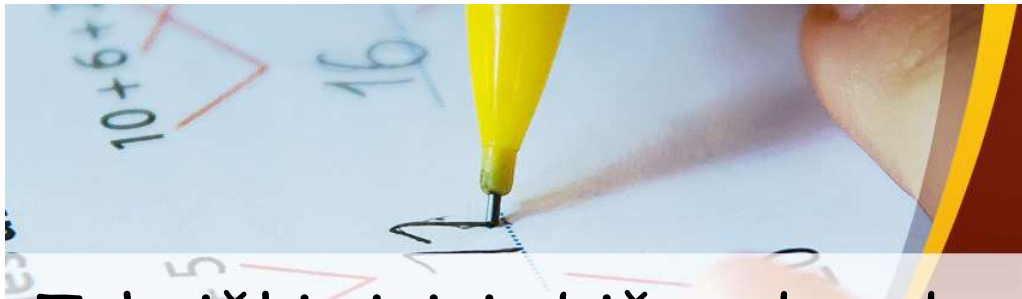
- Rizici uzrokovani nedostatkom pravilne implementacije, sistemskim greškama ili nekim spoljnim događajima rizika

NOVE VERZIJE SOFTVERA

- Postavljanje novih verzija softvera je bio veliki, složen i rizičan zadatak. Nakon što bi nova verzija softvera bila testirana, razvojni tim bi dobio zadatak da je rasporedi u proizvodnji.
- U zavisnosti od veličine softvera, taj proces je mogao da traje satima, danima ili nedeljama, i zahtevao je detaljan prolazak kroz kontrolne liste, što je činilo mnogo manuelnih koraka, kao i posebnu stručnost.

NOVE VERZIJE SOFTVERA

- Postavljanje novih verzija softvera je
  - ✓ veliki,
  - ✓ složen i
  - ✓ rizičan zadatak.
- Nakon što bi nova verzija softvera bila testirana, razvojni tim bi dobio zadatak da je pusti u prdukciju.
- U zavisnosti od veličine softvera, taj proces je mogao da traje satima, danima ili nedeljama, i zahtevao je detaljan prolazak kroz kontrolne liste, što je činilo mnogo manuelnih koraka, kao i posebnu stručnost.



Tehnički rizici

Tehnički rizici obično dovode do bagova u funkcionalnosti i performansama sistema.

Uzroci tehničkih rizika su:

- Stalna promena specifikacije
- Problem u vremenu
  - Korišćenje zastarelih tehnologija ili tehnologija u ranoj fazi.
  - Složenost implementacije projekta
  - Kompleksna modularna integracija projekata