

# Oblikovanje programske potpore

## Testiranje programske potpore



**Sveučilište u Zagrebu**  
**Fakultet elektrotehnike i računarstva**  
*Zavod za elektroniku, mikroel., računalne i inteligentne sustave*



# Tema

- **Testiranje i ciljevi testiranja**
- **Klasifikacija testiranja**
- **Upoznavanje procesa testiranja**
- **Principi testiranja sustava i komponenti**
- **Funkcionalno i strukturno testiranje**
- **Generiranje test slučajeva**
- **Svojstva alata za automatizaciju testiranja**
- **Cilj:**
  - Upoznavanje tehnika testiranja kao jedne od mogućih tehnikova verifikacije programske podrške
  - Razumijevanje terminologije, procesa testiranja i različitosti tehnikova testiranja

# Što je testiranje?

- Cilj testiranja je pokazati da program ispravo obavlja željene funkcije. (?)
- Testiranje je proces uspostave povjerenja ispravnog rada. (?)
- Testiranje je proces pokazivanja odsustva pogrešaka. (?)
- ***Testiranje je proces izvođenja programa sa svrhom pronalaženja pogrešaka. (!)***





# Testiranje

- Aktivnost s ciljem **otkrivanja informacija** o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema testiranog produkta.
- Za ostvarenje cilja upotrebljava:
  - eksperimentiranje;
  - logika, matematika;
  - modeli;
  - alati (programi, mjerni instrumenti, analizatori ...)
- Otkrivanje informacija
  - Organizirana i detaljna potraga za relevantnim informacijama
  - Aktivan proces istraživanja
    - postavljanje pitanja i analiza rezultata

# Ciljevi testiranja



- Testiranje uobičajeno obuhvaća više ciljeva, a time zahtijeva primjenu različitih strategija i testove, dokumentaciju i daju različite rezultate
- Najčešći cilj: **Pronaći i ispraviti pogreške**
- Osigurati pouzdanost, ispravnost, otkrivanje pogrešaka
- Minimizirati rizike
  - Provjeriti sukladnost rada (engl. interoperability) s ostalim komponentama
  - Pomoći u donošenju odluke o puštanju u rad/prodaju
  - Zaustaviti prerano puštanje u rad/prodaju (engl. premature product releases)
  - Minimizirati troškove tehničke podrške
  - Procijeniti sukladnost specifikacijama
  - Sukladnost s normama (zakonske, tehničke, ...)
  - Minimizirati rizik tužbi obzirom na sigurnost → vidi slijedeću sliku
- Definirati način sigurne uporabe
- Procjena kvalitete



# Testiranje

- **Problem testiranja:**
  - Bez specifikacije nema testiranja
- Testiranje znači usporedbu stvarnih rezultata s postavljenim standardima
- The Institute of Electrical and Electronics Engineers (IEEE) definira:
  - **test** - kao jedan ili više testnih slučaja (engl. Test case)
  - **testiranje** - kao proces analize programskog koda sa svrhom pronađaska razlike između postojećeg i zahtijevanog stanja (kvar, bug), te vrednovanja svojstava programa
    - ⇒ Odgovornost za verifikaciju i validaciju



# Primjer dokumenata

- IEEE Std. 829-1998 Standard for Software Test Documentation Contents
  - 1. Plan testiranja - Test Plan
    - Glavni plan testiranja i osnovne razine.
  - 2. Oblikovanje testiranja - Test Design Specification
    - Svojstva, pristup i rezultata testiranja.
  - 3. Testni slučajevi - Test Case Specification
    - Opis testnih slučajeva.
  - 4. Testne procedure -Test Procedure Specification
    - Koraci provođenja testova.
  - 5. Bilješke testiranja - Test Log
    - Zapis tijeka provođenja testova.
  - 6. Izvješća odstupanja - Test Incident Report
    - Bilješke anomalija u zahtjevima, oblikovanju, kodu ili testovima.
  - 7. Sažetak izvješća testiranja - Test Summary Report
    - Završno izvješće



# Terminologija

- **Testni podaci (I)**
  - Ulazi odabrani za provođenje određenog testa
- **Očekivani izlaz (O)**
  - zabilježen **prije** provođenje testa
- **Testni slučaj (engl. test case)**
  - uređeni par (I, O)
- **Stvarni izlaz**
  - Rezultat dobiven provođenjem testa
- **Kriterij prolaza testa**
  - Kriterij usporedbe očekivanog i stvarnog izlaza određen prije provođenja testa



# Svojstva testiranja

- Testiranje programske podrške zasniva se na **dinamičkoj** verifikaciji ponašanja programa na **konačnom** broju testnih slučaja, **pogodno odabranih** iz uobičajeno **beskonačne domene** izvođenja, obzirom na **očekivano** ponašanje
- Testiranje podrazumijeva izvođenje programa na ispitnim podacima
- Beskonačno i detaljno testiranje je nemoguće
  - praktična ograničenja resursa i vremena
- Tehnike testiranja se razlikuju u načinu odabira pogodnih kriterija i testnih slučajeva
- Mora omogućiti donošenje odluke o prihvatljivosti i očekivanim rezultatima



# Povijest testiranja

- Evolucija koncepta testiranja:
  - 1956 Prva faza: **ispravljanje pogrešaka** (engl. Debugging)
    - Testiranje = provjera rada programa i ispravljanje
    - engl. check-out & debugging
  - 1957 – 1978 Druga faza: **Orijentacija na demonstraciju** (engl. Demonstration)
    - **Razdvajanje provjere rada programa i ispravljanja grešaka;**
    - Testiranje pokazuje ispravan rad (tipični ulazi), sukladnost specifikaciji
  - 1979 – 1982 Treća faza: **Orijentacija na razaranje** (engl. Destruction)
    - Testiranje = izazivanje zatajenja sa svrhom otkrivanja pogrešaka
    - Uspješan test otkriva zatajenje
  - 1983 – 1987 Četvrta faza: **Orijentacija na evaluaciju** (engl. Evaluation)
    - Testiranje = dio validacije i verifikacije
    - Rano otkrivanje pogrešaka u zahtjevima, oblikovanju i implementaciji
  - 1988 – 2000 Peta faza: **Orijentacija na prevenciju** (engl. Prevention)
    - Testiranje = jedan od načina pokušaja izbjegavanja pogrešaka
    - Prevencija pogrešaka u zahtjevima, oblikovanju i implementaciji
  - 2000 – Šesta faza: **Orijentacija na razvoj prog. potpore** (engl. Discipline)
    - engl. **test-driven software development**
    - Rezultira programskim kodom podobnim za testiranje



# Testiranje danas

- Nije (ne smije biti) aktivnost koja započinje nakon završetka kodiranja s ograničenom svrhom otkrivanja pogrešaka
- Aktivnost objedinjena u procesu razvoja i održavanja i važan dio izgradnje produkta
- Započinje u ranim fazama analize zahtjeva
  - planovi testiranja i procedure moraju biti sistematično i kontinuirano razvijane, te prilagođavane sukladno razvoju



# Uporaba informacija u testiranju

## ■ Specifikacije

- Formalne, neformalne
- za odabir, generiranje, provjeru (testnih slučaja)

## ■ Informacije oblikovanja

- za odabir, generiranje, provjeru

## ■ Programski kod

- za odabir, generiranje, provjeru

## ■ Uporaba

- Povijest, model

## ■ Iskustvo



# Problemi zatajenja programa

- Ugrađena dijagnostika kao pomoć u analizi
  - Umetanje provjera u kod programa (**npr. `#ifdef _DEBUG_`**)
  - Negativi efekti (struktura podataka, resursi, vrijeme,...)
- Zatajenje zbog efekta “nenamjernog sljepila”
  - **Ljudi: Ne vide ono što nije u centru pažnje**
  - **Programi: Uvijek ne vide ono što im nije naglašeno da paze**
    - Precizniji i manje prilagodljivi
- Neponovljiva zatajenja
  - Razmatranje krivih uvjeta
    - nemoguće analizirati sve
- **Testovi praktično ne mogu pokriti sve mogućnosti grešaka** ☺



# Upravljanje pogreškama

## ■ **Prevencija** – engl. Error prevention

- Uporaba pogodnih metoda oblikovanja za smanjenje složenosti
- Sprečavanje nekonzistentnosti - CVS
- Primjena verifikacije za sprečavanje kvarova u algoritmima

## ■ **Detekcija** - engl. Error detection

- Tijekom rada
  - **Testiranje**
  - Ispravljanje pogrešaka – Debugging
  - Nadzor rada – Monitoring

## ■ **Oporavak** - engl. Error recovery

- U radu programa (npr. dijeljenje s nulom)
- Baze podataka (ponavljanje transakcija)
- Modularna zalihost (redundancija)

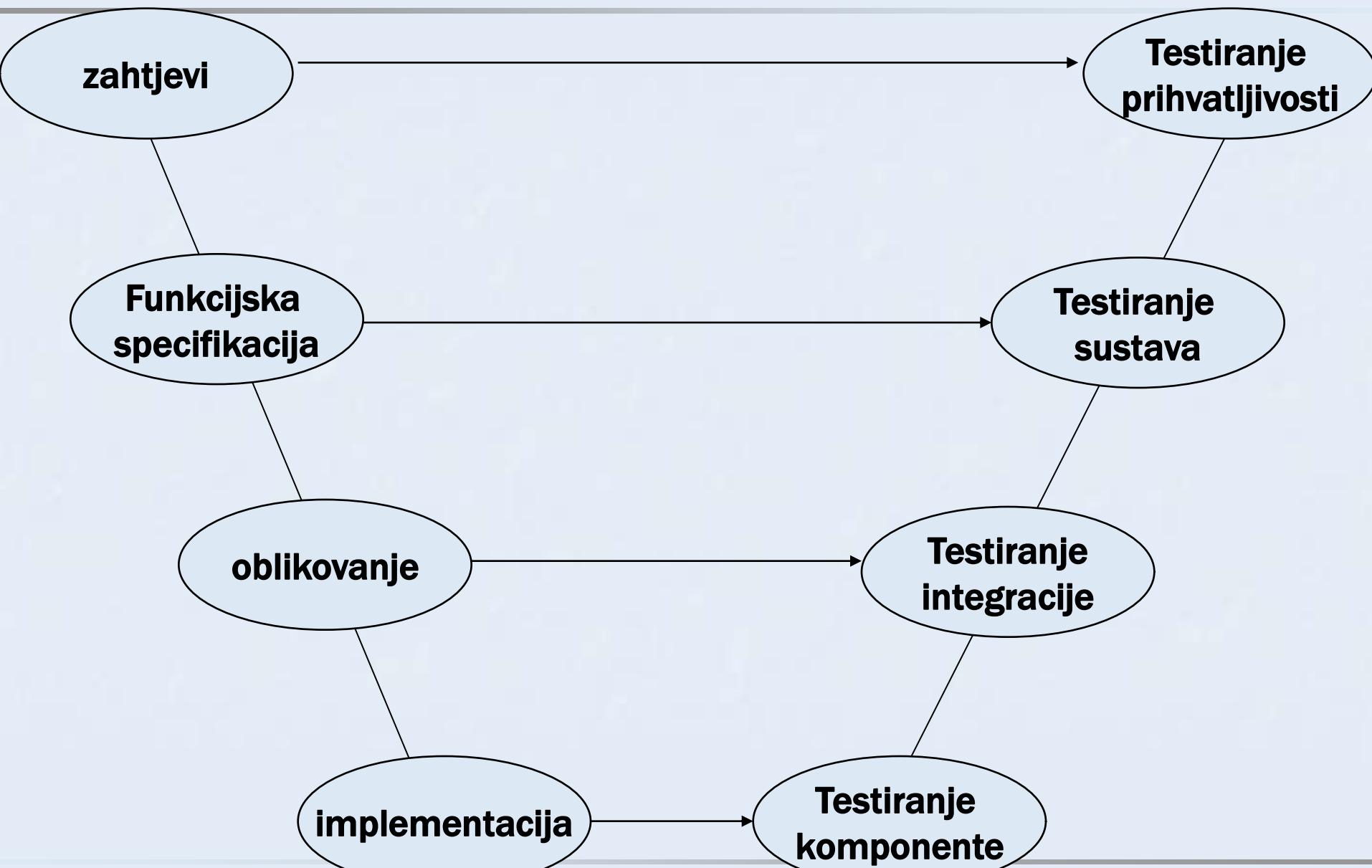


# Kada testirati?

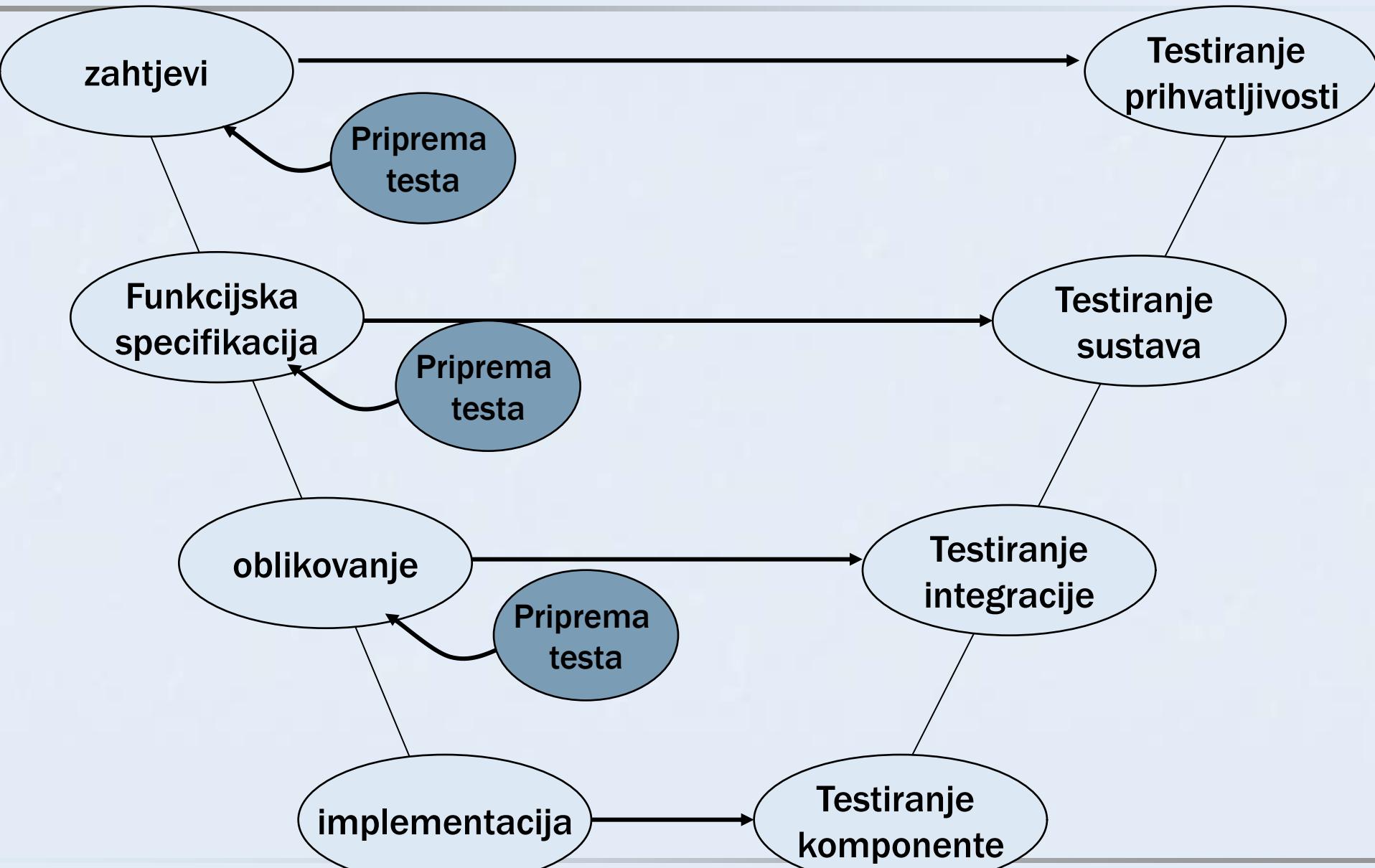
- Problem testiranja je nemogućnost potpunog testiranja bilo kojeg netrivijalnog modula
- ***“Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence”***
  - **1972 Dijkstra**
- Tri koraka:
  - Prije kodiranja
  - Tijekom kodiranja
  - Nakon kodiranja



# V-model testiranja

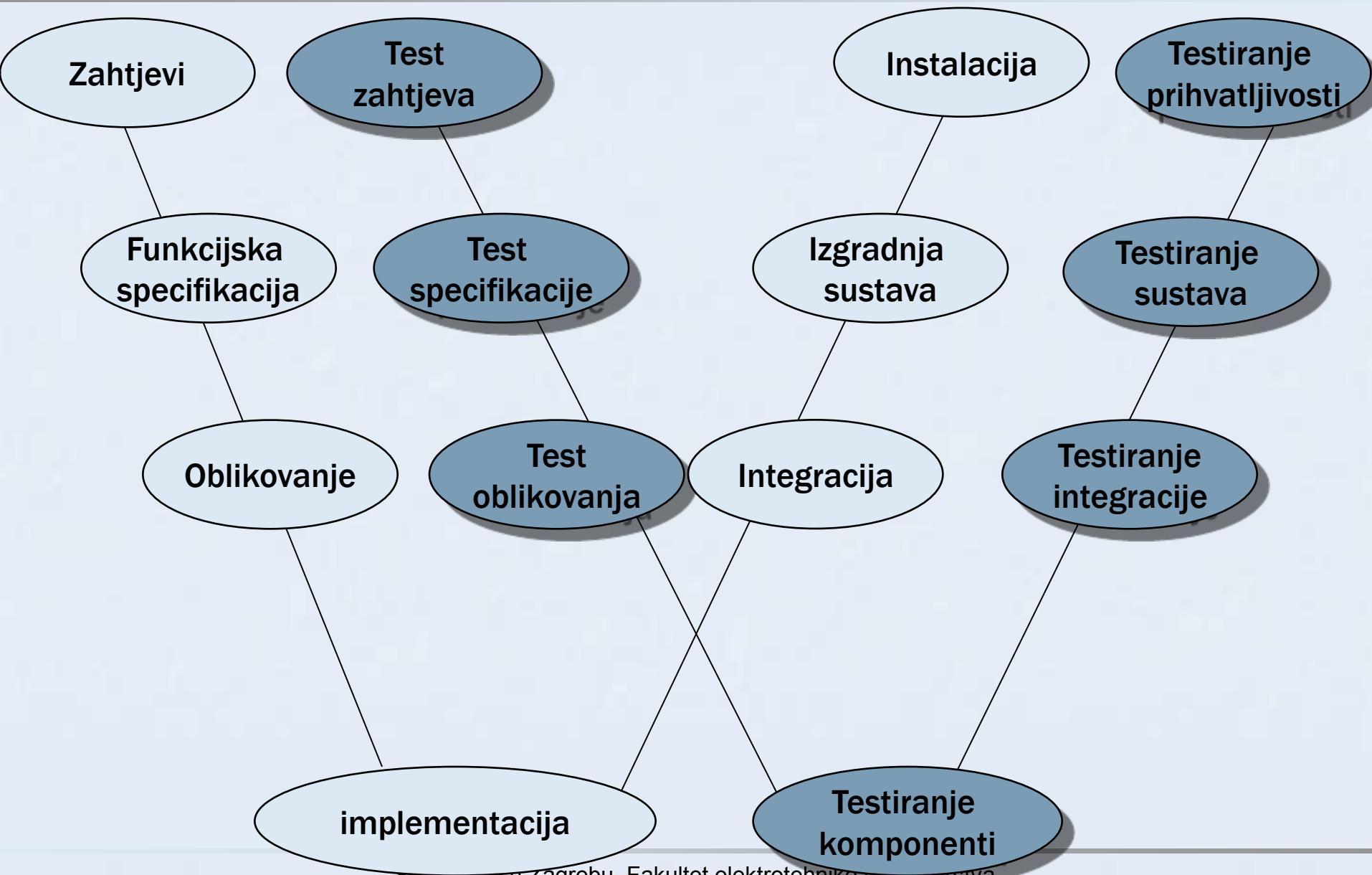


# V-model testiranja s ranom pripremom





# W-model testiranja





# Aktivnosti testiranja

## ■ Četiri osnovne kategorije:

### 1. **Oblikovanje testova** (engl. Test design)

- Oblikovanje testnih vrijednosti sa svrhom zadovoljenja ciljeva testiranja

- Analogno oblikovanju arhitekture programske podrške

### 2. **Automatiziranje testiranja** (engl. Test automation)

- Programiranje

### 3. **Testiranje** (engl. Test execution)

- Provođenje testova i bilježenje rezultata

### 4. **Valorizacija testova** (engl. Test evaluation)

- Poznavanje domene i postupaka testiranja

### ■ **Zahtijevaju različita znanja i vještine**



# Tehnike testiranja

- Klasifikacija obzirom na primjenjivost odabira testnih slučajeva
  - **Testiranje zasnovano na pokrivenosti**
    - engl. Coverage-based testing
    - Zahtjevi testiranja su specificirani obzirom na pokrivenost testiranog programa
  - **Testiranje zasnovano na pogreškama**
    - engl. Fault-based testing
    - Testni slučajevi koji omogućavaju otkrivanje pogrešaka
  - **Testiranje zasnovano na kvarovima**
    - engl. Error-based testing
    - Testni slučajevi zasnovani na poznavanju tipičnih mesta izloženih kvarovima (posebice u krivoj uporabi).



# Tehnike testiranja II

- Klasifikacija obzirom na izvor informacija upotrijebljenih za testne slučajeve
  - **Strukturno testiranje, bijela kutija**
    - White (glass) box testing
    - Structural testing
    - Program-based testing
  - **Funkcijsko testiranje, crna kutija**
    - Black box testing
    - Functional testing
    - Specification-based testing

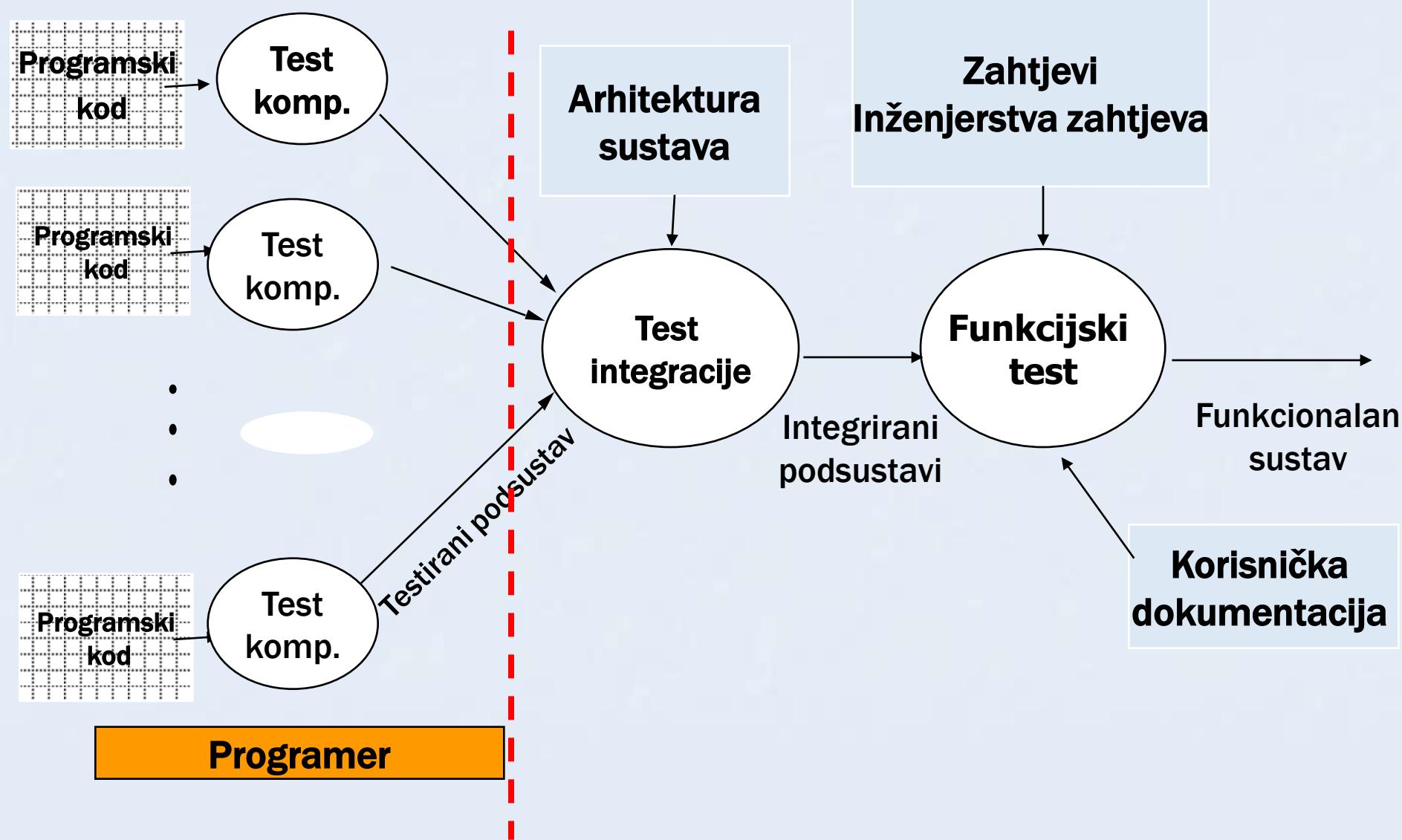


# Proces testiranja II

- **Testiranje komponenti**
  - engl. Component/Unit testing
- **Funkcijsko testiranje**
  - engl. Integration/Function testing
- **Testiranje sustava**
  - engl. System/Release testing
- **Test prihvatljivosti**
  - engl. Acceptance Testing
- **Test instalacije**
  - engl. Installation testing
- **Alpha test**
- **Beta test**

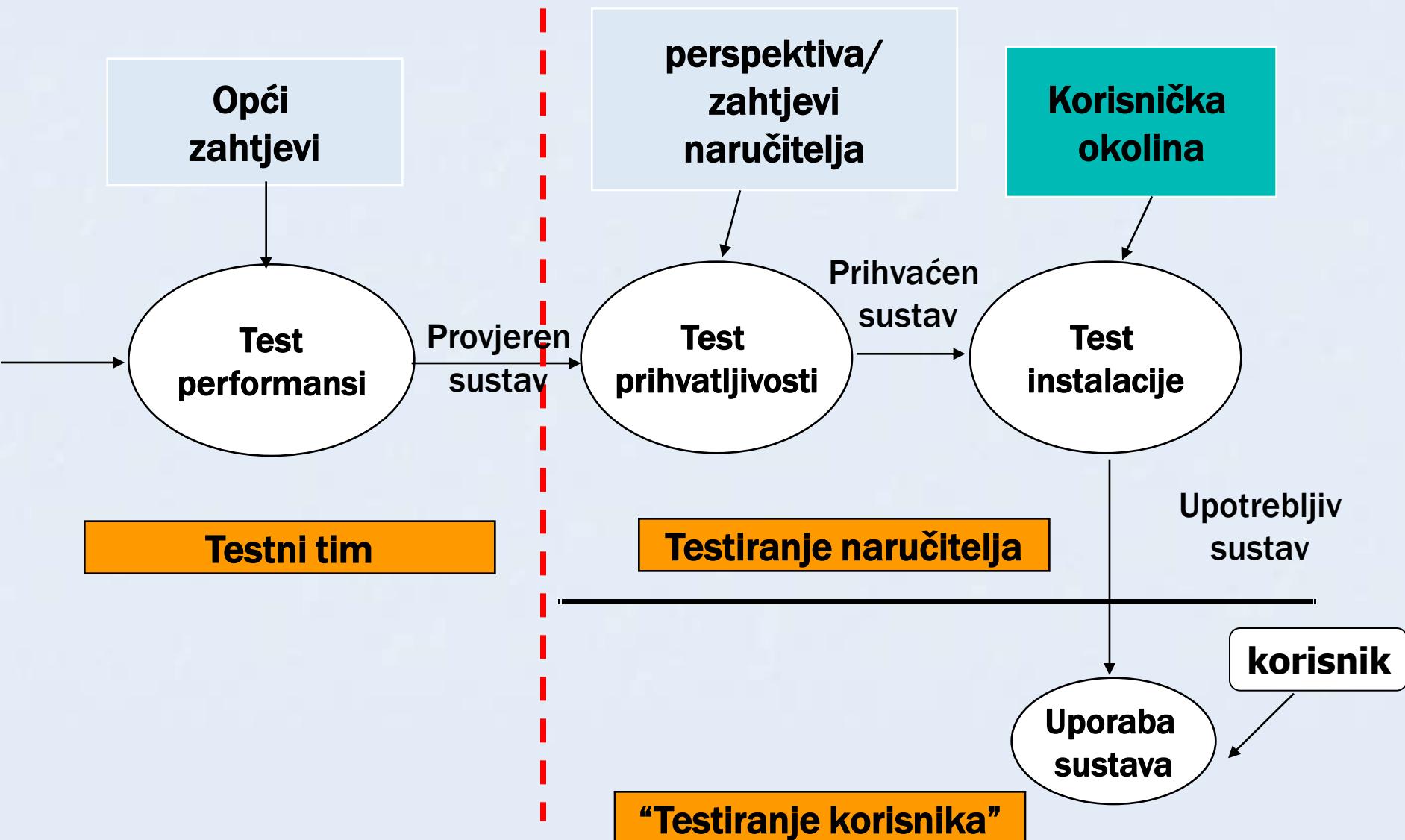


# Proces testiranja III





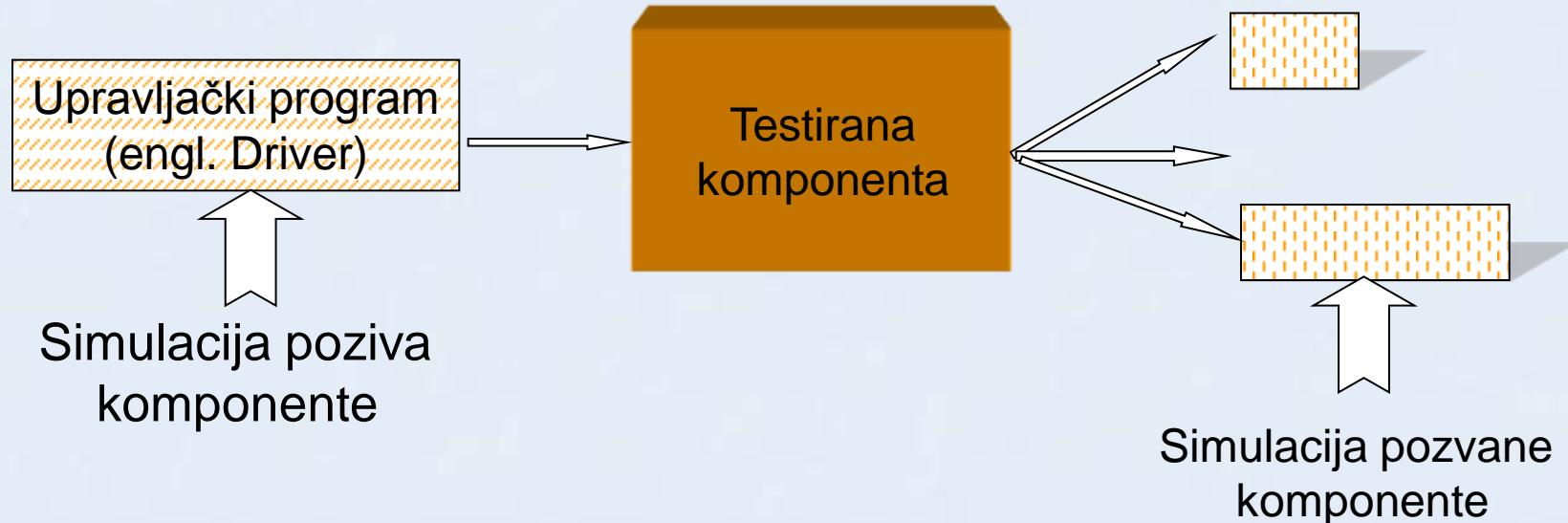
# Proces testiranja Illa





# Okolina testiranja komponenti

## ■ Postupak izolacije komponente u svrhu testiranja





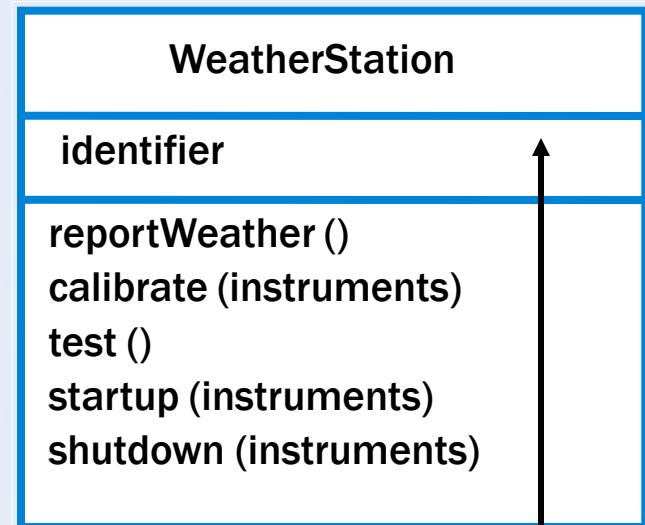
# Testiranje klase

- U OO sustavima klasa predstavlja komponentu
- Testiranje klase
  - Testiranje svih operacija
  - Postavljanje i ispitivanje svih atributa objekta
  - Ispitivanje objekta u svim stanjima (simulacija događaja koji mijenjaju stanje)
- Poteškoće testiranja klasa uzrokuje uporaba nasljeđivanja
  - Informacije koje treba testirati nisu lokalizirane



# Primjer

- **Klasa** *WeatherStation*
- **Definirati test slučajeve za**
  - reportWeather,  
calibrate, test,  
startup, shutdown.
- **Uporabom dijagrama stanja** pronaći sekvence prijelaza za testiranje i odgovarajuće sekvence događaja koje ih uzrokuju (vidi dijagram)
- **Npr.**
  - Waiting → Calibrating → Testing → Transmitting → Waiting
  - Shutdown → Waiting → Shutdown
  - Waiting → Collecting → Waiting → Summarising → Transmitting → Waiting

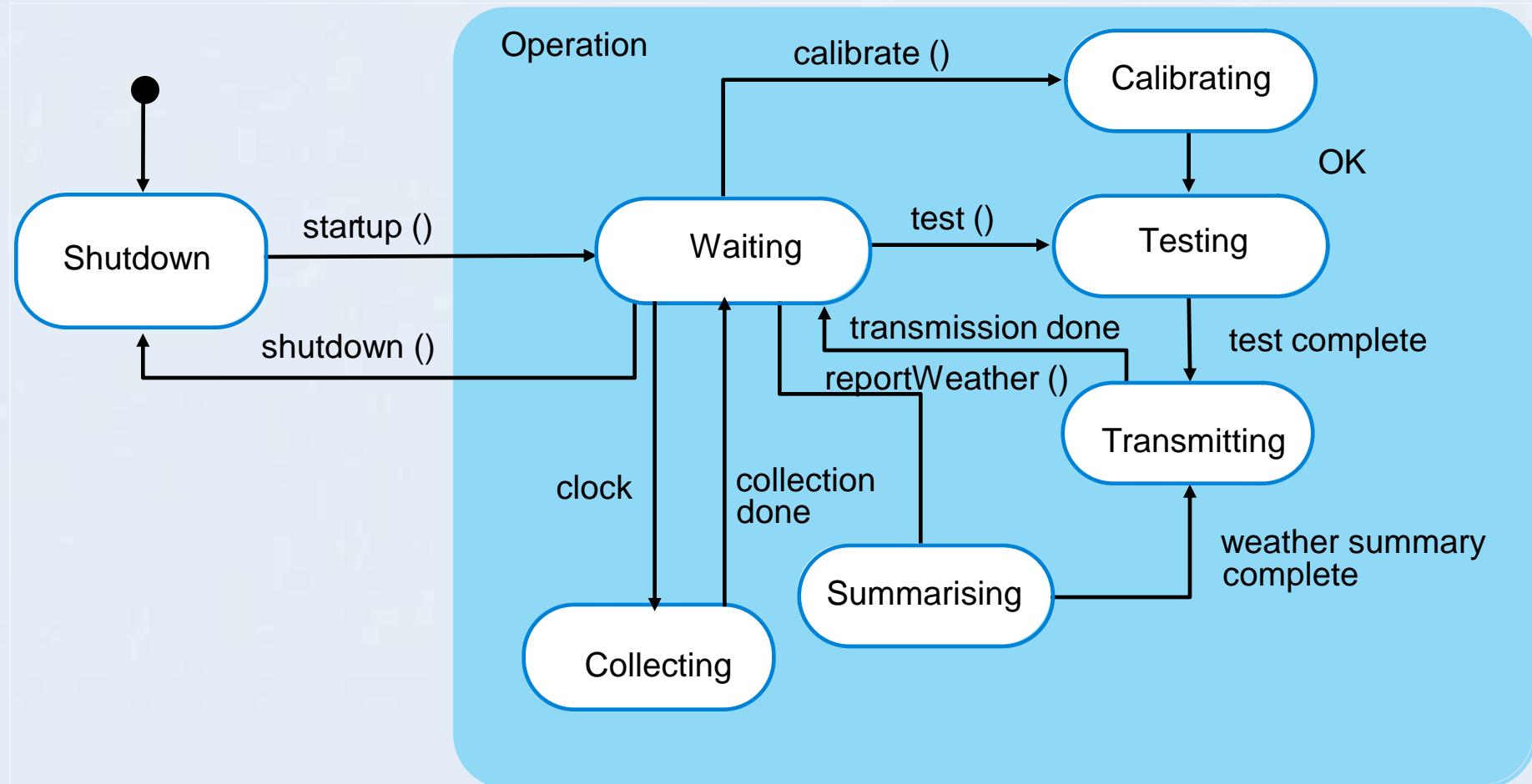


- test atributa **identifier**
- provjeriti da li je postavljen



# Primjer

## Dijagram stanja Weather station



Izvor: Sommerville, I., Software engineering



# Tipovi sučelja programskih komponenata

- **Parametarsko sučelje** (engl. *Parameter interfaces*)
  - Podaci (i funkcije) se prenose pozivima procedure
- **Dijeljena memorija** (engl. *Shared memory interfaces*)
  - Procedure dijele (pišu i čitaju) zajednički memorijski prostor
- **Proceduralno sučelje** (engl. *Procedural interfaces*)
  - Podsustavi (komponente) obuhvaćaju skup procedura koje pozivaju ostali podsustavi (npr. objekti)
- **Sučelje zasnovano na porukama** (engl. *Message passing interfaces*)
  - Podsustavi traže usluge od ostalih podsustava slanjem poruke (npr. klijent-uslužitelj)



# Kvarovi sučelja

- Pogrešna uporaba (*engl. Interface misuse*)
  - Komponenta koja poziva drugu pogrešno upotrebljava njezino sučelje
    - **Npr. pogrešan redoslijed parametara**
- Nerazumijevanje sučelja (*engl. Interface misunderstanding*)
  - Komponenta koja poziva drugu pogrešno prepostavlja specifikaciju njezinog ponašanja (**npr. binarno pretraživanje na neuređenom nizu**)
- Vremenske pogreške (*engl. Timing errors*)
  - Pozvana i pozivajuća komponenta rade **različitom brzinom** (npr. zajednička memorija te različita brzina čitanja i pisanja)
- **Naputak za testiranje sučelja:**
  - **Oblikuj testove tako da parametri poprime ekstremne vrijednosti**
  - **Uvijek testiraj pokazivače (ako se šalju sučeljem) s null vrijednostima**
  - **Oblikuj test proceduralnog sučelja tako da zataji komponenta**
  - **Sustave s razmjenom poruka testiraj na stres (generiraj više pouka nego što je normalno potrebno)**
  - **Sustave s dijeljenom memorijom testiraj s različitom redoslijedom aktiviranja komponenti**



# Pristupi integracijskom testiranju

## ■ **Big bang**

- Integrirati sve komponente bez prethodnog testiranja
- U slučaju pogrešaka problem predstavlja otkrivanje mesta pogreške

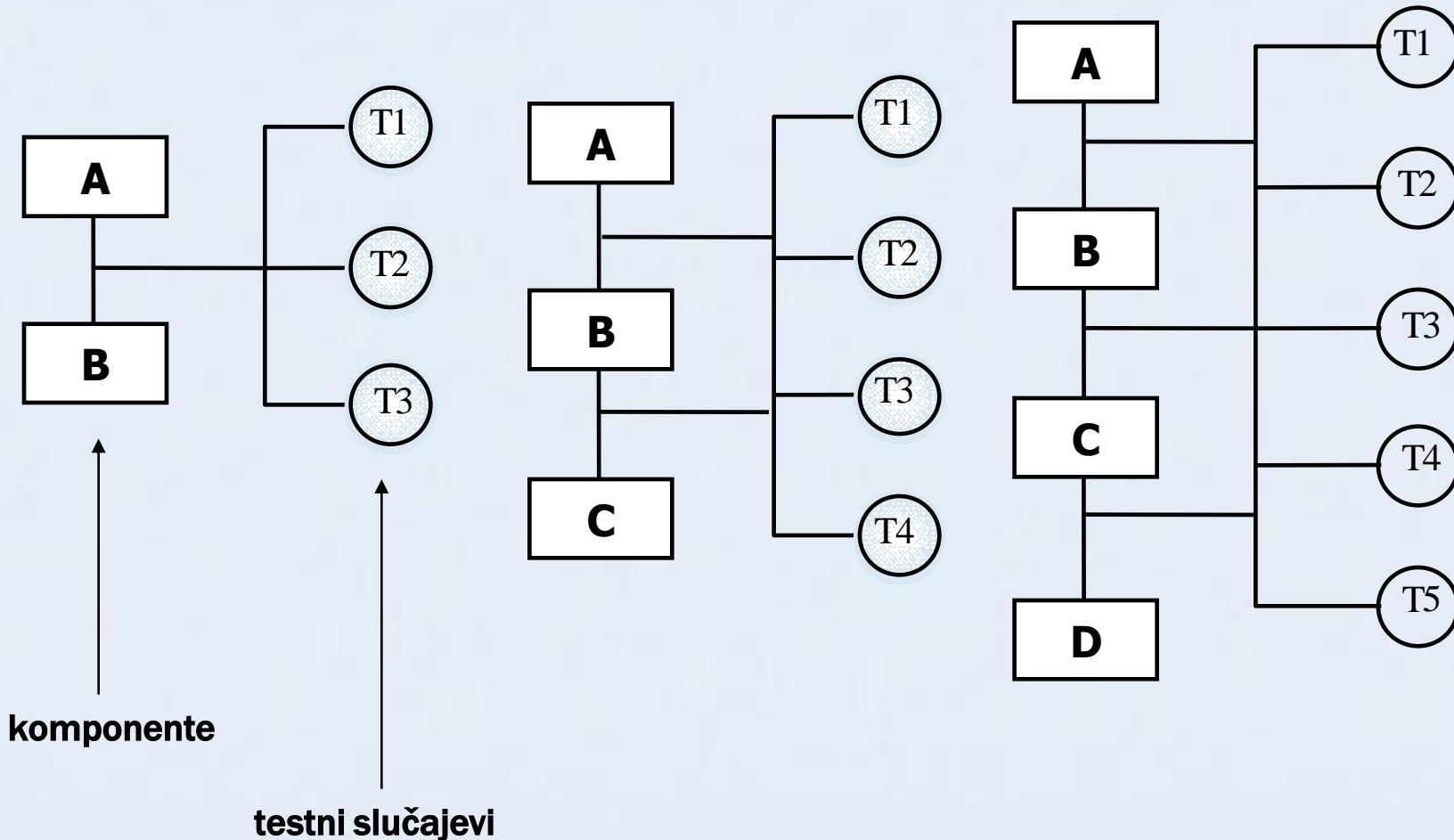
## ■ **Poboljšani big bang**

- Integracija svih komponenti nakon provedenog testiranja
- U slučaju pogrešaka i dalje prisutan problem otkrivanja mesta pogreške

## ■ **Inkrementalni**

- Integracija i testiranje sustava dio po dio
- Uobičajen pristup
- Efikasno u lokalizaciji mesta pogreške

# Inkrementalno integracijsko testiranje



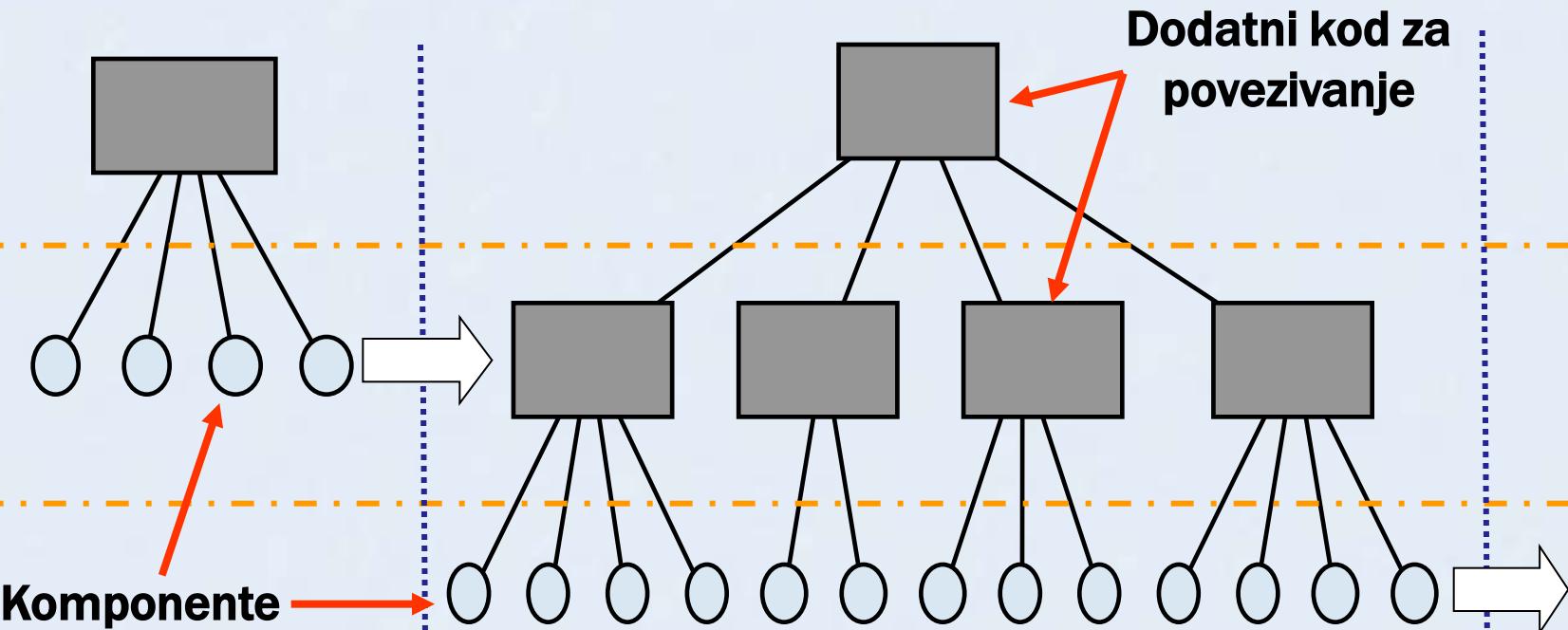
Izvor: Sommerville, I., Software engineering

# Inkrementalno integracijsko testiranje



## ■ Odozgo na dolje

- Razviti kostur sustava i popuniti ga komponentama



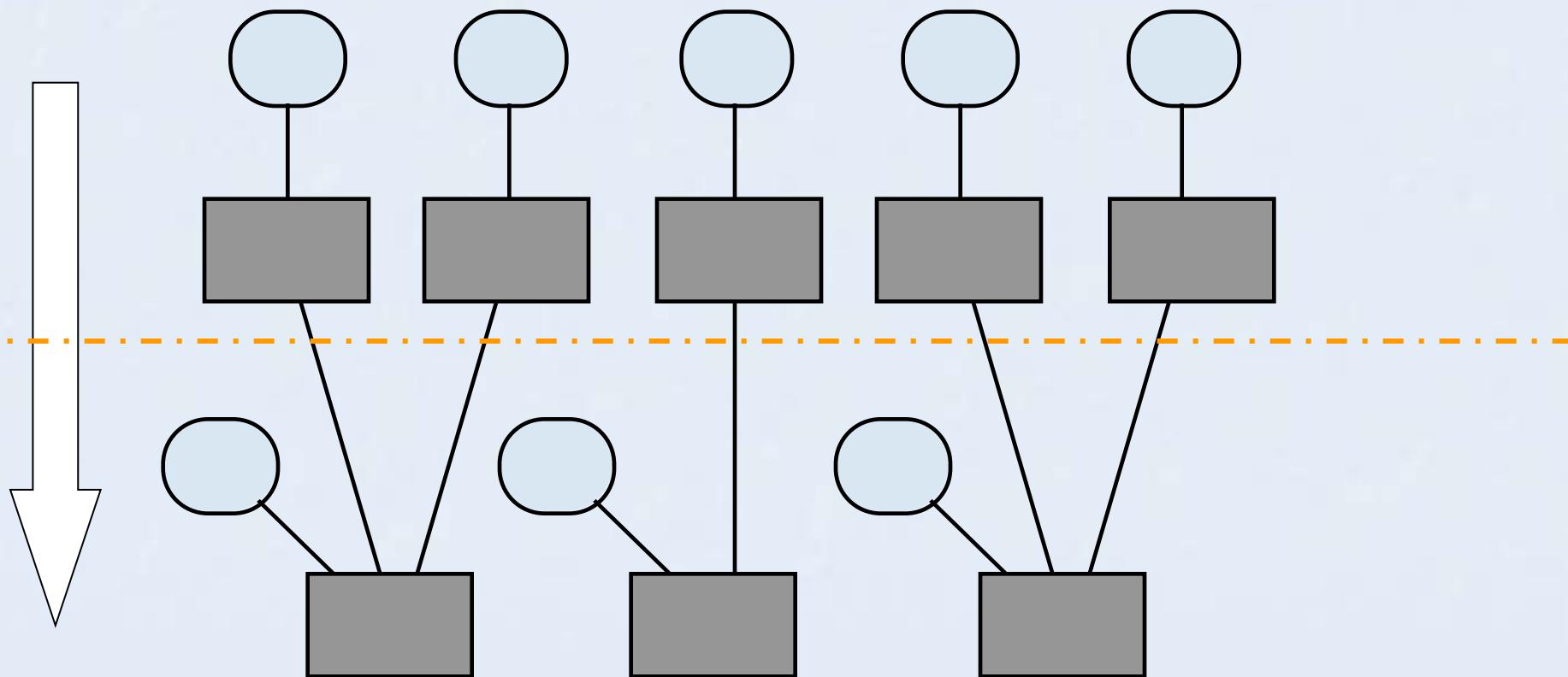
Izvor: Sommerville, I., Software engineering

# Inkrementalno integracijsko testiranje



## ■ Odozdo na gore

- Prvo integrirati neke komponente (najvažnije i najčešće funkcionalnosti), te dodati preostale



Izvor: Sommerville, I., Software engineering



## ■ Funkcijska integracija

- Integriranje komponenti u konzistentne funkcije bez obzira na hijerarhijsku strukturu (kombinacija odozgo-prema-dolje i odozdo-prema-gore)
- Najčešći postupak u praksi



# Ciljevi testiranja

- Objašnjeno ranije, ali zbog značaja ponavljamo:
  - **Otkrivanje (provociranje) pogrešaka u programu**
  - **Uspješan test** uzrokuje **nенормално ponašање** programa !?
  - Test prikazuje postojanje, a ne odsustvo pogreške
  - **Potrebno kontinuirano testiranje tijekom razvoja** programske podrške (V i W modeli testiranja)



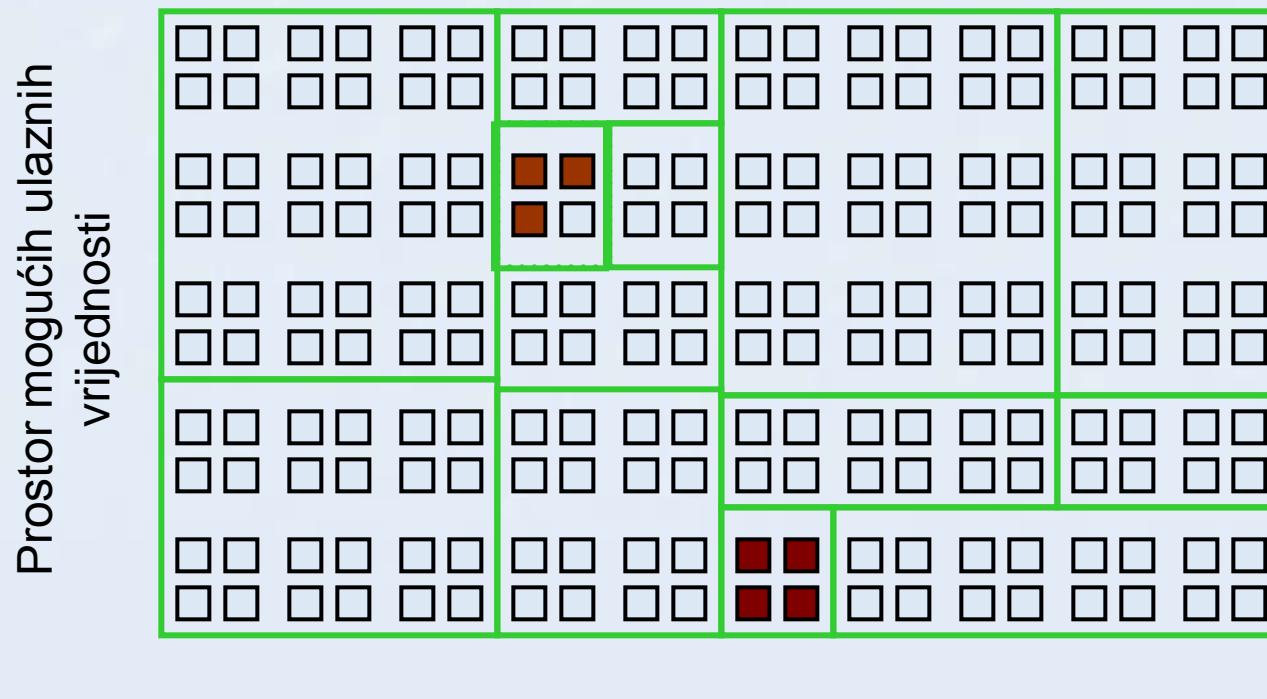
# Strategije testiranja

- **Iscrpno testiranje** (engl. Exhaustive testing)
  - Testiranje svih mogućih slučajeva
  - Moguće samo za ograničene primjere
- **Slučajno testiranje** (engl. Random Testing)
  - Nije isto kao i “ad hock” testiranje
  - Odabir testnih slučajeva temeljem vjerojatnosti
    - Uniformna razdioba
    - Razdioba temeljena na prethodno prikupljenim podacima
- **Sustavno testiranje**
  - Podjela ulaznih podataka u pod-domene i odabir testnih slučajeva temeljem različitih karakteristika
    - svojstva koda, specifikacija, rizik, ...



# Sustavno testiranje particija

- 30-85 pogrešaka u 1000 LOC (*Line Of Code*)
- Dobro testirani programi 0.5 – 3 pogreške
- Pogreške su često grupirane
- Cilj je izolirati područja s vjerojatnom pojavom pogreške





# Funkcionalno testiranje

- engl. **Black box Testing**
- Prepostavlja da **nema znanja programskog koda** ili oblikovanja sustava
  - Koncentracija na U/I ponašanje
  - Testiranje samo prema zahtjevima i specifikacijama
- Ako za ulazne podatke možemo predvidjeti izlaz modula test prolazi
  - Gotovo nemoguće generirati sve moguće ulaze
- Cilj: Smanjiti broj testnih slučajeva **ekvivalentnom podjelom ulaza** i analizom graničnih vrijednosti
- Oblikovanje testnih slučajeva (engl. **test case design**)
  - Zasnovano na specifikaciji sustava
  - Podjela vrijednosti ulaznih podataka
  - Podjela ulaznih podataka u klase
  - Odabir testnih slučaja za svaku klasu
  - Analiza graničnih vrijednosti



# Testiranje particija

- Ulazni podaci i rezultati često se mogu grupirati u različite klase u kojima je ponašanje članova slično (princip: *Podijeli i vladaj*).
- Svaka od tih klasa predstavlja ekvivalentnu particiju (engl. **equivalence partition**) u kojoj se program ponaša na isti način za sve njegove članove
- **Testni slučajevi moraju pokrivati sve ekvivalentne particije**
  - Posebice granične vrijednosti svake particije jer su to najčešći uzroci kvara
- **Odabir ekvivalentnih particija za testiranje**
  - **Podijeliti ulazne ekvivalentne particije na:**
    - Valjana vrijednost
    - Nevaljana vrijednost
  - **Vrijednosti ulaza su valjane u intervalu – odabrati 3 testna slučaja**
    - **Vrijednost ispod intervala**
    - **Vrijednost u intervalu**
    - **Vrijednost iznad intervala**

prepostavka



# Primjer funkcionskog testiranja

- Zadatak: *Testirati program za odlučivanje o pravu na glasovanje. Pravo glasovanja ovisi o starosti osobe, minimalna dob je 18 godina.*

- Jedna varijabla - dob
- Svi ulazni podaci mogu se podijeliti u **dvije ekvivalentne podjele**

- 0-17 - ne smije glasovati
- $\geq 18$  - smije glasovati

- **Test slučaj 1**

- Odabir reprezentativnih podataka iz svake podjela

- **Test slučaj 2**

- Odabir podataka na granicama podjela

Test slučaj 1

Test	Podatak	Rezultat
1	12	NE
2	21	DA

Test slučaj 2

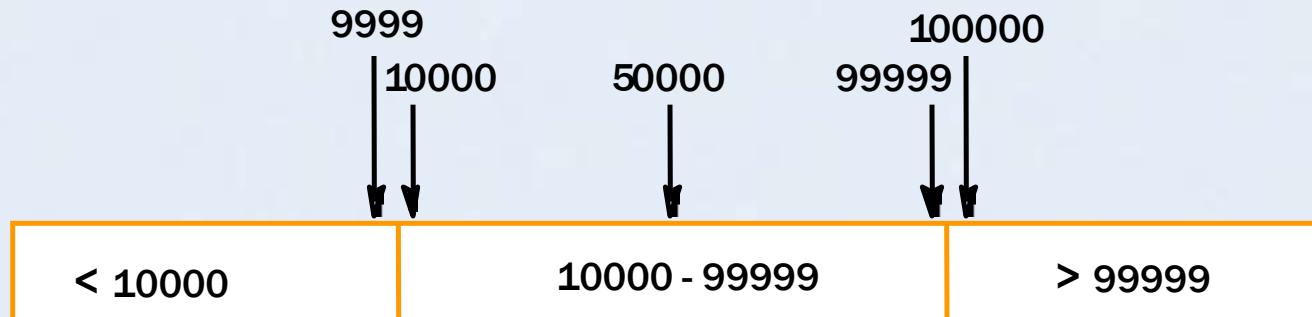
Test	Podatak	Rezultat
3	17	NE
4	18	DA



# Ekvivalentne podjele

- Primjer: Ulaz 5-znamenkasti broj vrijednosti između 10000 i 99999

- <10,000
- 10,000-99,999
- >99,999





# Primjer:

Traži:

Rezultat:

element (ulazna sekvencija)	kljuc	nadjen, L
17	17	T, 1 (1 vrijednost)
17	0	F, ? (1 vrijed., ne)
17, 21, 23, 29	17	T, 1 (više vrij., početak)
9, 16, 18, 30, 31, 41, 45	45	T, 7 (kraj)
17, 18, 21, 23, 29, 38, 41	23	T, 4 (oko sredine)
17, 18, 21, 23, 29, 38, 41	21	T, 3
12, 18, <b>23, 21</b> , 32	23	T, 3 (neuređen niz)
21, 23, 29, 33, 38	25	F, ? (više vrijd., ne)



# Struktурно testiranje

- engl. **Structural Testing, White Box Testing**
- Testiranje očekivanog ponašanja zasnovano na svojstvima programa ili oblikovanju (tj. strukturi programa)



- Cilj testiranja je pokrivanje izvođenja svih mogućih naredbi programa
- Primjer:
  - Testiranje komponenti, struktурно testiranje (**statement coverage, branch coverage etc, unit testing**)
- Oblikovanje testnih slučajeva (engl. test case design)
  - Zasnovano na strukturi programa (poznata struktura)



# Struktурно testiranje

## Analiza puta

### Graf tijeka programa

- engl. **control flow graph**
- Graf. reprezentacije tijeka programa

### Čvorovi – procesi ○, programske odluke ○ ili ◇ Lukovi – kontrola tijeka

## Ispituju se:

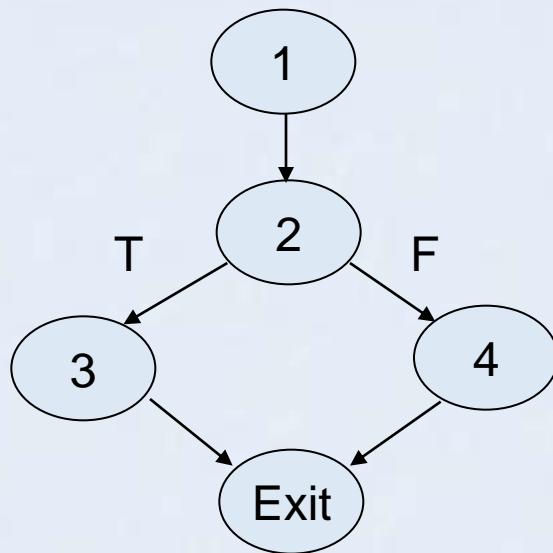
- Svi neovisni putevi
- Logički izrazi (T,F)
- Petlje (rubovi, sredina)
- Interni podaci - struktura

```
public roots (double a, b, c)
{
    double q = b * b - 4 * a * c;
    if (q > 0 && a != 0) {
        ...
    }
    else if (q == 0) {
        x = (-b) / (2 * a);
    }
    else {
        ...
    }
}
```



# Primjer struktturnog testiranja

## Graf tijeka programa



**Test slučaj 1:** osigurati izvođenje oba puta tijekom testiranja

Test	Podatak	Rezultat
1	12	NE
2	21	DA

## Analiza puteva

Mogući putevi:

1.  $1 \rightarrow 2 \rightarrow 3 \rightarrow \text{kraj}$
2.  $1 \rightarrow 2 \rightarrow 4 \rightarrow \text{kraj}$

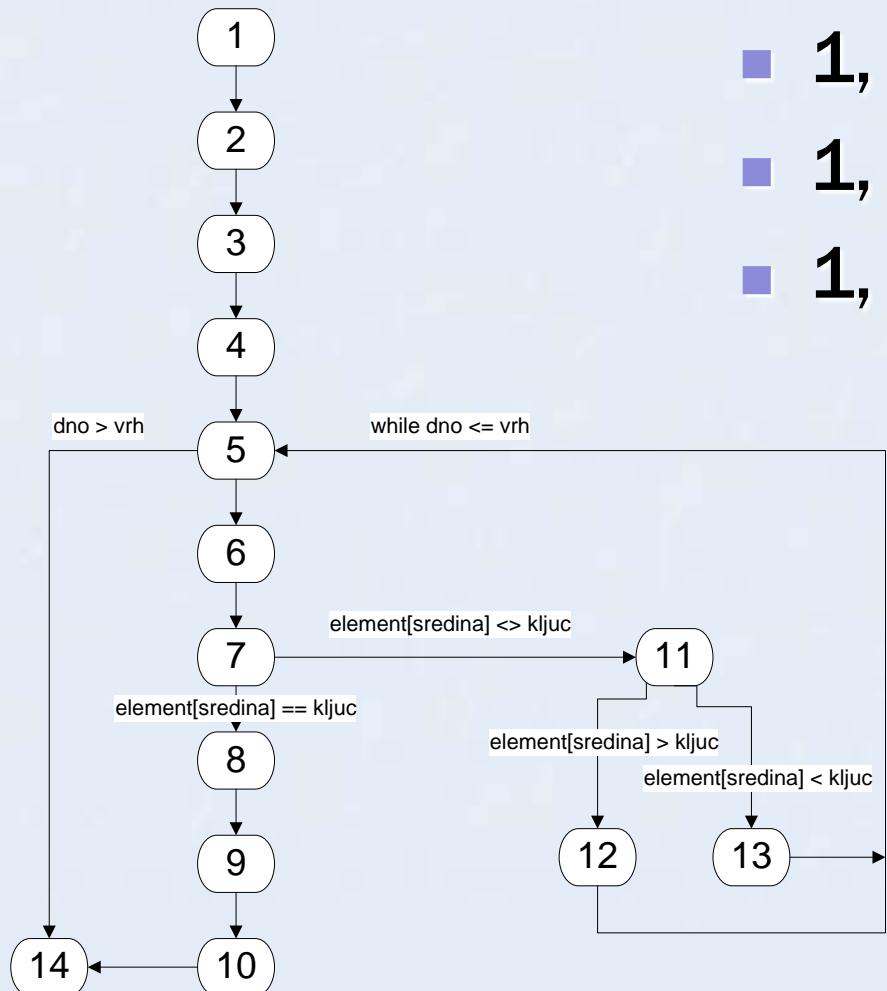
**Test slučaj 2:** osigurati ispravan rad *if* naredbe na graničnim vrijednostima

Test	Podatak	Rezultat
3	17	NE
4	18	DA



# Primjer: Binarno pretraživanje

Čvor je linija u programu



- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14
- 1, 2, 3, 4, 5, 14
- 1, 2, 3, 4, 5, 6, 7, 11, 12, 5, ...
- 1, 2, 3, 4, 6, 7, 2, 11, 13, 5, ...

Ako se izvedu svi navedeni putovi:

- svaka naredba je testirana najmanje jednom
- testirano je svako grananje



# Kombinacijsko testiranje

- engl. **Combination testing**
- međutjecaj varijabli
- Za N varijabli ( $V_1, V_2 \dots V_N$ ) broj mogućih kombinacija
  - $V_1 \times V_2 \times \dots \times V_N$
- Primjer:
  - Broj kombinacija dvije varijable definirane kao jednoznamenkasti dekadski broj
    - $10 \times 10 = 100$
  - Koliki je broj mogućih kombinacija prva četiri poteza u šahu?
    - 318.979.564.000



# Selektivno testiranje

- testiranje svih kombinacija je najčešće nemoguće
- suzujemo prostor testiranja na:
  - Testiranje temeljnih putova
    - engl. **Basis path testing**
  - Testiranje uvjeta
    - engl. **Condition testing**
  - Testiranje petlji
    - engl. **Loop testing**
  - Testiranje protoka podataka
    - engl. **Dataflow testing**



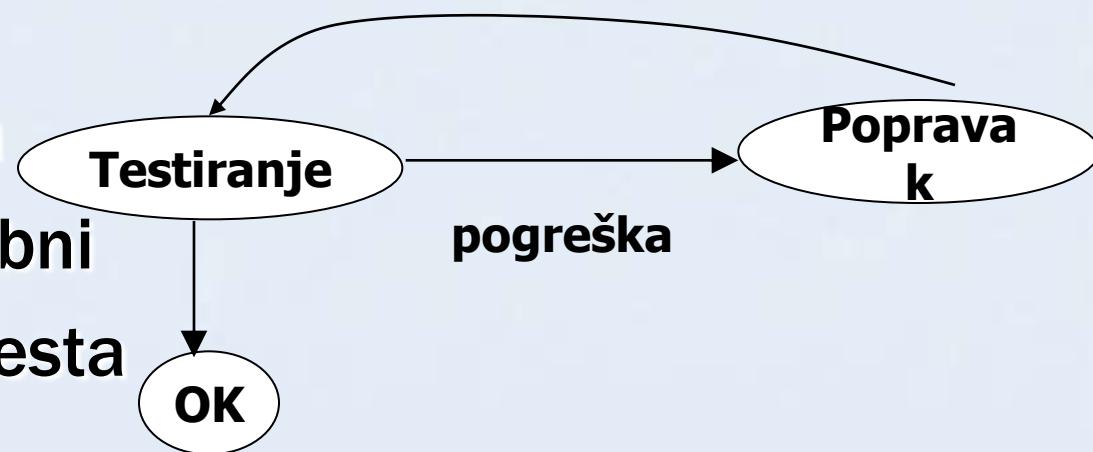
# Testiranje uvjeta

- Ispitivanje svih logičkih uvjeta u modulu
  - Jednostavni uvjeti
    - Booleve varijable True, False
    - Jednostavni relacijski izrazi  $a < b$ ,  $a >= b$  .....
  - Složeni izrazi
    - $((a = b) \& (c > d))$
- Metode testiranja
  - Testiranje grana (svaka grana svakog uvjeta ispituje se bar jednom)
  - Testiranje domene
    - Npr. za relaciju  $a < b$ , treba 3 testa:  $a < b$ ,  $a = b$ ,  $a > b$
    - Booleov izraz sa  $n$  varijabli  $\Rightarrow 2^n$  testova



# Regresijsko testiranje

- engl. **Regression Testing**
- Ponovno testiranje nakon promjene (popravka)
- Testiranje ispravljenih programa s ciljem potvrđivanja ispravnosti promjena i ne postojanja negativnog utjecaja na nepromijenjene dijelove programa
- Promjene programa
  - Neki testovi nepotrebni
  - Mijenjaju rezultate testa
  - Izrada novih testova





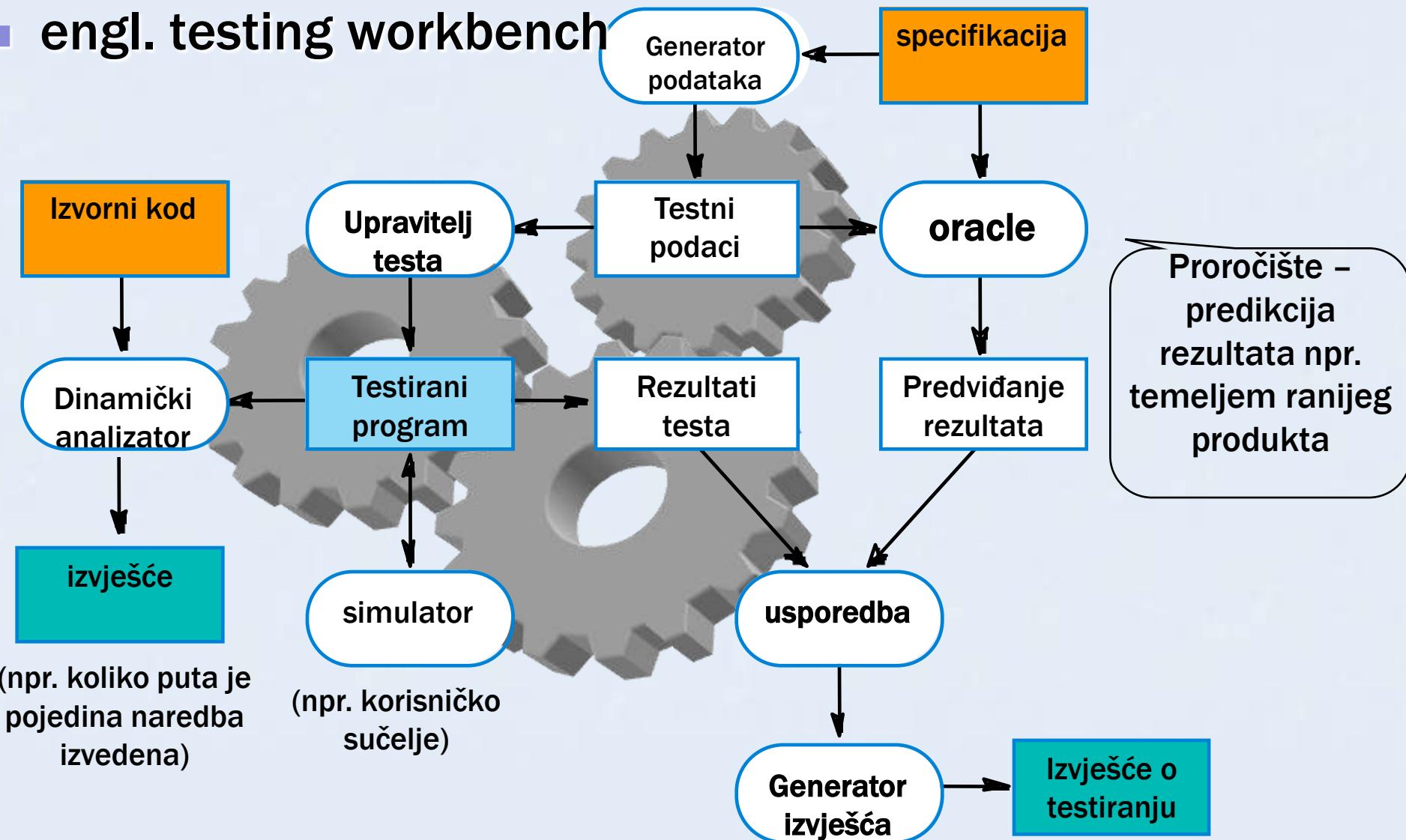
# Automatizacija testiranja

- Cijena ručnog testiranja visoka
  - Ponavlja se svakim testiranjem
- Automatizacija ručnog testiranja
  1. Generiranje ulaznih podataka i očekivanih rezultata
  2. Izvođenje testa
  3. Evaluacija
  - Jedinično 3-30 puta cijene ručnog testiranja
  - Zahtjeva formalizirani ručni proces testiranja
  - Cijena ovisi o postavljenom dosegu testiranja - Postupak kodiranja testova
  - Cijena ponavljanja  $\cong 0$
- Prednosti
  - Povećana pouzdanost
  - Povećana kvaliteta testiranja (automatizacija procesa)
  - Kraće vrijeme izvođenja testova
  - Automatska analiza rezultata testova
- Nedostatak
  - Cijena, Vrijeme

# Testna radna klupa



## ■ engl. testing workbench





# Zaključak

- Testiranje je složena i zahtjevna aktivnost oblikovanja programske podrške
  - Kritični dio razvoja sustava
  - Provodi se na svim razinama i planira kao sastavni dio razvojnog procesa
  - **Problem strategije**
    - veliki broj ciljeva  $\Rightarrow$  oblikovanje strategije za postizanje ciljeva dionika
  - **Problem odluke - *The Oracle Problem***
    - Da li je program prošao test ili ne
    - Nemogućnost potpunog testiranja
- Testiranje može pokazati postojanje pogreške, a ne može dokazati njihovo nepostojanje!
- Alternativa: uporaba formalnih metoda