**Testiranje programa**: izvođenje programskog koda skupom ulaznih podataka

**Cilj testiranja**: pronaći neispravnosti u programskom kodu; test je dobar ako njegovo izvođenje prouzroči ispad programa

**Odabir test uzoraka**: program ne možemo testirati svim mogućim ulaznim podacima (iscrpno testiranje); reprezentativni uzorak test podataka

**Osnovni zadatak testiranja**: pronalaženje svih elemenata ulazne domene koji izazivaju ispad programa

**Ispad programa**: slučajni događaj (odstupanje rezultata izvođenja programa od programske specifikacije); postoji razlika između izlaznog stanja definiranog specifikacijom i stvarnog rezultata izvođenja programa; signalizira postojanje neispravnosti u programskom kodu

**Neispravnost programa *bug***: dio programskog koda koji uzrokuje ispad programa; nastaje uslijed pogreške programera; otkriva se inspekcijom koda ili detekcijom ispada programa

**Testna jedinica  *implementation under test***: programska cjelina koja se testira

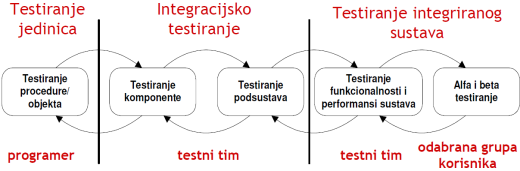
**Testni uzorak *test case***: element ulazne domene I koji omogućuje jedno samostalno izvođenje testne jedinice

**Zbirka testnih uzoraka *test suite***: skup testnih uzoraka koji su povezani u grupu na temelju odabranog cilja testiranja ili implementacije povezanosti

***Debugging***: analiza koda radi lokaliziranja i otklanjanja neispravnosti

**Definicija testiranja IEEE/ANSI**: testiranje programske opreme je proces izvođenja sustava ili komponente pod strogo definiranim uvjetima, pri čemu se promatraju i bilježe rezultati izvođenja te ocjenjuju svojstva sustava ili komponente; proces analize programa s ciljem otkrivanja razlika između zahtijevanog i stvarnog ponašanja programa

**Faze životnog ciklusa**: **definiranje zahtjeva** (definiranje i opois funkcionalnosti budućeg programskog proizvoda); **dizajn sustava**(definiranje arhitekture sustava, komponente i njihovi odnosi; opis komponenti na različitim nivoima apstrakcije); **implementacij**a (kodiranje i testiranje komponenti sustava; testiranje jedinica, testiranje komponenti); **integracija sustava i testiranje** (povezivanje komponenti i testiranje cijelog sustava; testiranje podsustava, testiranje integriranog sustava, alfa test); **održavanje** (korektivno testiranje)

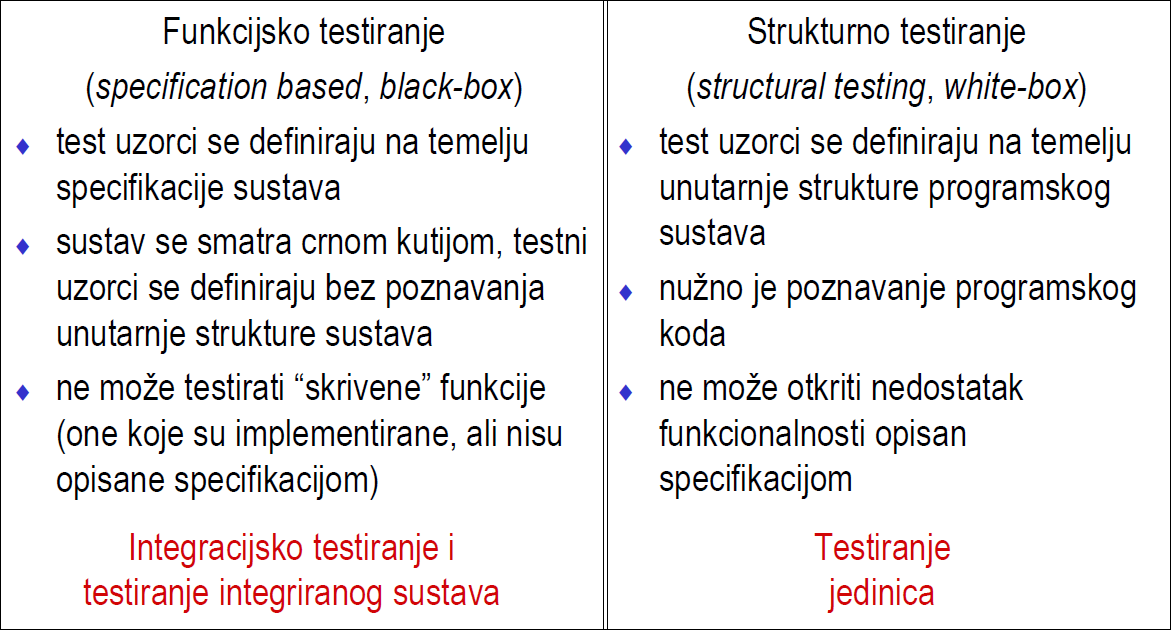
****

**Osnovni koraci testiranja**: **1.** definiranje skupa testnih uzoraka za testiranje programskog sustava; **2.** izvođenje sustava odabranim uzorcima i detekcija eventualnih ispada; **3.** ako je opažen ispad, slijedi analiza koda i *debugging*; **4.** uklanjanje neispravnosti; **5.** ponovno izvođenje testova prethodno definiranim test uzorcima (regresijsko testiranje); **6.** u slučaju ispada ponoviti 3.4.5.

**Aksiomi testiranja**: testiranje se koristi radi pronalaženja neispravnosti, a ne radi dokazivanja njihovog nepostojanja. dobar test uzorak uzrokuje ispad sustava i time razotkriva nesipravnost; osnovni problem testiranja je određivanje trajanja procesa testiranja; proces testiranja valja pažljivo planirati i pratiti. testni uzorci se odabiru prema definiranom kriteriju. testni uzorci i rezultati testiranja trebaju se bilježiti u bazu podataka radi ponovnog korištenja

**Trajanje procesa testiranja**: metrika 1 (postotak programskih elemenata koji su izvedeni testiranjem; pokrivenost iskaza; odluka, putova); metrika 2 (pouzdanost)

**Metode testiranja**: metoda određuje princip testnih uzoraka iz ulazne domene koji će učinkovito testirati programski sustav; **odabiranje na temelju** (slučajno; specifikacija – funkcijsko testiranje; programski kod – strukturno testiranje)



**Funkcijsko testiranje**: testiranje domene *domain testing*

**Strukturno testiranje**: testiranje putova *path testing*; toka podataka *data flow testing*

**Domena**: ulazna domena *I*, skup svih mogućih test uzoraka

**Poddomena**: particija domene definirana graničnim uvjetima; program se „ponaša“ jednako za sve elemente iz poddomene

**Pretpostavka metode testiranja**: test uzorci s granice poddomene su najčešći uzorci ispada programa

**Koraci testiranja**: **1.** odredi poddomene za sve ulazne varijable koje čine testne uzorke testne jedinice; **2.** za svaku varijablu odaberi vrijednost na samo granici poddomene, zatim vrijednost izvan granice poddomene, te tipičnu vrijednost iz poddomene; **3.** definiraj testne uzorke koristeći odabrane vrijednosti; **4.** odredi očekivane izlaze za odabrane testne uzorke i testiraj testnu jedinicu

**Testiranje putova**: strukturno testiranje na temelju programskog koda; kod modeliramo usmjerenim grafom i na temelju grafa definiramo putove; izvođene testa – prolazak kroz određeni put grafa; problem – programske petlje (velik broj putova; putovi koji prolaze kroz neku petlju 1..n puta su ekvivalentni)

**Modeliranje koda grafom**: čvor grafa (modelira blok naredbi); grana grafa (modelira prijenos toka upravljanja)

**Određivanje broja testnih putova**: **Ciklomatička kompleksnost** (C = e – n + 2; e – broj grana; n – broj čvorova)

**Kriterije za pokrivenost koda**: **Pokrivenost iskaza** *statement coverage* (generirati test uzorke tako da se svaki iskaz izvornog koda izvede barem jednom); **Pokrivenost grananja** *branch coverage* (izvesti T i F grane svakog grananja); **Pokrivenost uvjeta**  *Condition coverag* (svaki uvijet poprima T i F ishod barem jednom); **Pokrivenost svih putova**  *All-paths coverage* (izvesti sve nezavisne putove)

**Testiranje toka podataka**: strukturna tehnika (izvođenjem programskog koda izvode se akcije nad varijablama (slijedovi akcija: ispravno – DD, DU, UU, UD, UK; neispravno – DK, KD, KU, KK)

**Objekt**: može pohraniti stanje *informacija*; može izvoditi određene operacija *ponašanje*; posjeduje identitet *object identifier OID*)

**Karakteristike OS**: enkapsulacija (objekt skriva unutarnju strukturu od okolice

**Klasa**: opisuje unutarnju strukturu objekta; instanca klase – objekt

**Sustav**: skup objekata (dinamičko ponašanje – komunikacija među objektima)

**Nasljeđivanje**: ako klasa B nasljeđuje klasu A, podaci i metode klase A postaju dio klase B

**Polimorfizam**: „ista“ metoda može imati različite implementacije u dvije različite klase; statičko i dinamičko povezivanje; klasa B nasljeđuje klasu A i redefinira, nadjačava metodu klase A *overriding*)

**Prednosti objektnog pristupa**: smanjena vjerojatnost pogrešaka vezanih uz naredbe kontrole toka; integracijsko testiranje je manje dramatično (najprije se testira grupa klasa s dobro definiranim sučeljima, potom podsustavi i na kraju cijeli integrirani sustav)

**Problemi testiranja OS**: dinamičko povezivanje i kompleksna struktura nasljeđivanja; apstraktne klase; povećana vjerojatnost grešaka vezanih uz sučelja; složena kontrola promjene stanja objekta

**Plimorfizam i dinamičko povezivanje**: povezivanje jedne reference s više objekata; statički i dinamički polimorfizam; smanjuje veličinu i leksičku kompleksnost koda; moguće neispravnosti (povezivanje pogrešnog objekta s referencom; metoda objekta može biti neispravna)

**Asocijacija**: modelira vezu među klasama (asocijacija, agregacija, kompozicija); implementacija asosijacije podložna sljedećim tipovima grešaka (pogrešna implementacija višestrukosti asocijacije; problem brisanja objekta; problem mijenjanja objekta)

**Programsko inženjerstvo**: definira sve praktične aspekte razvoja i podrške programu, od inicijacije programskog projekta, razvoja, isporuke, održavanja i evolucije programa (PO - kvaliteta, proces, metode, pomagala)

**Svojstva programske opreme**: **Osnovna svojstva** (nevidljivost, složenost, promjenjivost, interoperabilnost; ne mjenjaju se); **Ostala svojstva** (programski jezik, brzina sklopovlja, memorija; mogu se mijenjati); **Arhitektura programa** (funkcijska; objektno orjentirana)

**Parametri kvalitete**: korektnost; pouzdanost; prilagođenost korisniku; točnost; robusnost; pogodnost za održavanje; pogodnost za testiranje;vitalnost; cijena; dijagnostičnost; ispitljivost

**Problem pouzdanosti programske opreme**: **Razvoj novih sustava** (nema prethodnog iskustva; nemogućnost dobre predikcije; rješavanje problema koji su bez primjene računala nerješivi); **Implementacija zahtjevnih funkcionalnosti** (složene manipulacije informacijama); **Složenost programske opreme** (ne postoji univerzalno prihvaćena mjera složenosti; veličina LOC najčešće primjenjivana, evidentni i dramatični porast; SW sustavi koje u potpunosti nitko ne razumije; upravljanje neočekivanom složenošću – sustav ne smije biti složeniji nego što je potrebno za omogućavanje zadane funkcionalnosti); **Dokazivanje raspoloživosti** (inheretna diskretnost ponašanja dig. sustava; nemoguće testirati pod svim mogućim uvijetima)

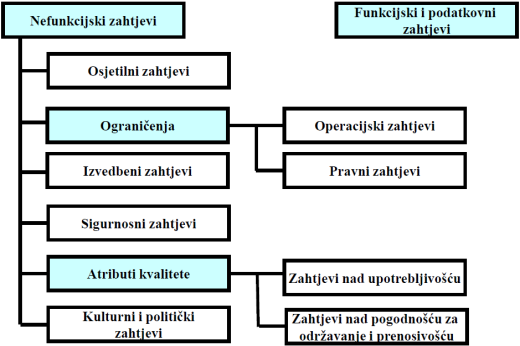


**Podjela programskih zahtjeva**: **funkcijski** (opisuju funkcije prog. proizvoda; ovise o tipu prog. proizvoda i korisnicima; opis željenih usluga u detalje; trebaju definirati: na koji način detektirati neispravnost; kako se zaštiti od ispada; oporavak sustava); **ne-funkcijski** (opisuju ograničenja vezana uz prog. proizvod ili proces njegova razvoja; trebaju specificirati: željenu pouzdanost i raspoloživost)

**Proizvod**: efikasnost; pouzdanost; portabilnost

**Organizacija**: implementacija; standardi; isporuka

**Vanjski**: interoperabilnost; etika; zakoni; privatnost; sigurnost



**Ne-funkcijska specifikacija pouzdanosti**: treba se izraziti kvantitativno; dinamičko svojstvo sustava (spec. pouzdanosti u odnosu na kod programa nije korisna; ova je mjera korisna jedino za analizu nakon isporuke programa kako bi se procijenila korištena tehnika razvoja programa); adekvatna mjera pouzdanosti se treba odnositi na ukupnu pouzdanost sustava

**Uzroci neispravnosti u SW sustavima**: zahtjevi; dizajn; implementacija; ljudski postupci; interakcije s okružjem

**Ispad**: *software failure*; događaj u nekom trenutku vremena kada prog. sustav nije u mogućnosti osigurati uslugu ili funkcionalnost prema očekivanjima korisnika ili prema programskoj specifikaciji

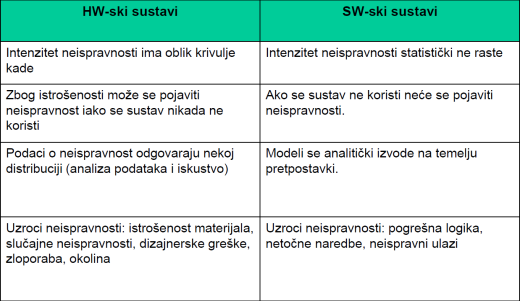
**Neispravnost**: *software fault*; neispravno stanje programskog sustava, stanje koje nije u skladu s programskim dizajnom

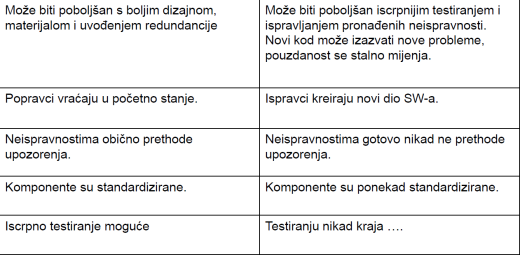
**Pogreška**: *Error, mistake*; programerska pogreška koja rezultira uvođenjem neispravnosti u programski sustav (dizajn, kod)

**Neispravnost i ispadi**: programski ispadi su posljedica programskih neispravnosti; neispravnosti ne uzrokuju uvijek ispad programa (tranzijentne neispravnosti); greške ne uzrokuju uvijek neispravnosti (ugrađeni mehanizmi detekcije i ispravljanja grešaka; ugrađeni mehanizmi zaštite od neispravnosti)

**Primjeri uzroka programskih ispada**: podatkovne greške; dvosmislenost podataka; nerazumijevanje ili pogrešno interpretiranje programske spec; pogrešan kod; neadekvatno testiranje; netočno ili neočekivano korištenje programa od strane korisnika;.....

**Usporedba pouzdanosti HW i SW sustava**





**Korisničko poimanje pouzdanosti**: pogrešne pretpostavke o okolini; utječu posljedici ispada sustava

**Pouzdanje**: *dependability*; svojstvo kritičnih programskih sustava koje opisuje u kojoj mjeri korisnici mogu imati povjerenje u taj sustav; najčešće je najvažnije svojstvo kritičnih sustava; predstavlja mjeru korisnikovog pouzdanja da će programski sustav raditi prema korisničkim očekivanjima i da neće doći do ispada u normalnim uvjetima rada; korisnost =! pouzdanost

**Troškovi pouzdanja**: eksponencijalan rast cijene zbog primjene skupljih tehnika i sklopovlja za povećano pouzdanje, te duže i iscrpnije testiranje i validacija programskih sustava kako bi se korisnici uvjerili da je stvarno postignuta željena razina pouzdanja

**Postizanje pouzdanosti**: **Izbjegavanje neispravnosti** (primjena tehnike razvoje prog. opreme koje sprečavaju ili minimiziraju vjerojatnosti programerskih grešaka); **Detekcija i uklanjanje neispravnosti** (primjena tehnika verifikacije i validacije koje povećavaju vjerojatnost pronalaženja i uklanjanja neispravnosti prije nego se programska oprema isporuči i stavi u pogon); **tolerancija neispravnosti** (primjena tehnika u realnom vremenu koje osiguravaju da eventualne programske neispravnosti ne rezultiraju ispadom sustava)

**Izbjegavanje neispravnosti**: *fault free*; precizna formalna specifikacija; organizacija strogo drži do normi kvalitete; skrivanje informacija i enkapsulacija u SW dizajnu; primjena programskih jezika sa sintaksnom provjerom i provjerom za vrijeme izvođenja; pouzdan i ponovljiv proces razvoja

**Tolerancija neispravnosti**: **detekcija neispravnosti** (sustav mora detektirati da se pojavila neispravnosti); **procjena štete** (detekcija dijelova sustava zahvaćenih neispravnošću); **oporavak** (sustav se treba oporaviti u poznato sigurno stanje); **ispravak neispravnosti** (modifikacija sustava kako bi se spriječilo ponavljanje neispravnosti)

**SW arhitekture s tol. neispravnosti**: implementacija u različitim programskim jezicima; primjena različitih alata i razvojnih okružja; primjena različitih algoritama tijekom implementacije

***N-version programming***: ista specifikacija je implementirana u različitim verzijama od strane različitih timova – sve verzije se izvršavaju istovremeno i izlaz se određuje glasanjem, temeljem pojedinih izlaza svake verzije; najčešće u zrakoplovima; empirijski dokazi: timovi na isti način krivo interpretiraju spec. i odabiru isti algoritam

**Sigurnost**: sposobnost programskog sustava da se izvodi bez opasnosti za ljude i okolinu odnosno da u slučaju zlonamjernog napada ili neočekivanog slučajnog događaja nastavi ispravnofunkcionirati bez ugroze raspoloživosti, vjerodostojnosti, cjelovitosti i povjerljivosti informacije i komunikacije; sve vaćnije svojstvo

**Sigurnosni zahtjevi**: isključivi zahtjevi; opisuju sve nepoželjne situacije (više nego opis funkcionalnosti programskog sustava

**Vanjski**: povjerljivost; cjelovitost; autentičnost; raspoloživost (zaštita komunikacije među komponentama sustava, te sustava s okolinom)

**Unutarnji**: neporicanje; kontrola pristupa; raspoloživost (zaštita komponenata sustava)

**Karakteristične neispravnosti**: različiti timovi često rade slične greške; specifikacijske neispravnosti; višestruke specifikacije

**Primarni kritično-sigurni sustavi**: ugrađeni programski sustavi čiji ispad uzrokuje ispad priopadnog sklopovlja i direktno ugrožava ljudske živote

**Sekundarni kritično-sigurni sustavi**: sustavi čiji ispadi uzrokuju ispade drugih sustava što može ugroziti ljudske živote

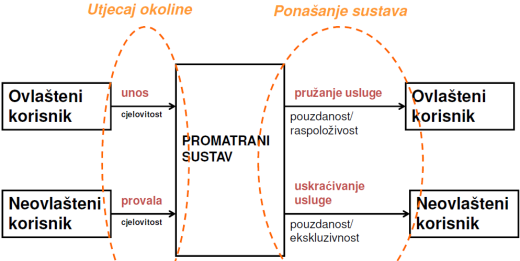
**Pouzdanost** se odnosi na zadane specifikacije i okolinu; **Sigurnost** se odnosi na osiguravanje da sustav ne uzrokuje štetu bez obzira odgovara ili ne odgovara zadanoj specifikaciji

**Privatnost**: pravo osobe da kontrolira jasnost osobnih podataka (korisnik mora – svjestan da se njegovi osobni podaci prikupljaju; informiran o kojim podacima je riječ te u koju se svrhu koriste

**Identitet**: skup atriputa koji jednoznačno opisuju korisnika

**Vjerodostojnost**: autentičnost; potvrda identiteta korisnika (jest; ima; zna)

**Ovlaštenost**: autorizacija; utvrđivanje kojim resursima korisnik ima pravo pristupa (lista za kontrolu pristupa; autorizacija bez identifikacije)



**Sigurnosne prijetnje i napadi**: uništavanje *destruction*; promjena *corruption* (*modification, fabrication, denial*); uklanjanje *removal* (*interception*); razotkrivanje *disclosure* (*interception*); obustava *interruption* (*denial*)

**Rizik**: mogućnost da se ostvari prijetnja zbog postojanja ranjivosti u sustavu; kvantitativno se može izraziti umnoškom utjecaja ostvarene prijetnje na rad sustava i vjerojatnosti da se ista ostvari

**Akcije nad rizicima**: **izbjegavanje** (potpuno izbjegavanje ostvarivanja određenog rizika); **umanjivanje** (smanjivanje utjecaja rizika); **prebacivanje** (prebacivanje utjecaja rizika na drugu domenu); **prihvaćanje**

**SecuRUP artefakti**: vizija/analiza sigurnosti; specifikacija sigurnosnih zahtjeva; plan implementacije sigurnosti; plan ispitivanja sigurnosti; plan isporuke sigurnosti

**10 principa postizanja sigurnosti**: smanjenje područja napada; uspostava referentne sigurnosti; princip najmanjih ovlasti; princip obrane u dubinu; otkazivanje usluge na siguran način; nepovjerenje u bilo kakve usluge; razdvajanje dužnosti; sigurnost nije samo čuvanje tajni; princip jednostavne sigurnosti; sveobuhvatno ispravljanje ranjivosti

**Nesigurni pouzdani sustavi**

**Greške u specifikaciji**: sustav se ponaša prema specifikaciji ali njegovim izvođenjem može nastati neželjeni incident

**Sklopovske greške koje generiraju lažne ulazne parametre**: teško predvidjeti specifikacijom

**Naredbe ovisne o kontekstu**: najčešće rezultat korisničke pogreške; ispravna naredba u krivom trenutku

**Incidenti**: u složenim sustavima rijetko imaju samo jedan uzrok; rezultat kombinacije neispravnosti

**Mjerenje pouzdanosti**: pouzdanost sustava se određuje mjerenjem broja ispada i njihovim povezivanjem s okolnostima u kojima je do ispada došlo, te mjerenjem vremena tijekom kojeg je sustav bio u stanju normalnog rada; pouzdanost kritičnih sustava određuje se implementacijom posebnih programa za mjerenje svih parametara potrebnih za određivanje pouzdanosti

**POFOD**: *probability of failure on demand*; vjerojatnost da će doći do ispada programa pri zahtijevanju određene usluge/funkcionalnosti; korisna mjera kad su zahtjevi za uslugama neredoviti i rijetki, a neisporučivanje istih ima teške posljedice; relevantno za mnoge sigurnosno kritične sustave

**ROCOF**: *Rate of failure occurence*; vjerojatnost pojave neočekivanog ponašanja sustava; relevantno za operativne sustave, sustave za obradu različitih transakcija – obrada velikog broja sličnih zahtjeva koji se redovito pojavljuju

**MTTF**: vrijeme koje je proteklo između 2 ispada sustava; za stabilne sustave MTTF ≈ ROCOF; relevantan za sustave s dugim transakcijama

**Posljedice ispada**: mjere pouzdanosti NE uzimaju u obzir posljedice ispada; tranzijentne neispravnosti ne moraju imati nikakve posljedice, druge neispravnosti mogu imati teške posljedice; potrebno identificirati različite klase neispravnosti i koristiti različitu mjeru za svaku klasu pojedinačno; **strukturirana specifikacija pouzdanosti** – uzima se u obzir broj ispada i njihove posljedice

**Tranzijentni ispad**: pojavljuju se samo za određeni skup ulaznih parametara

**Permanentni ispad**: pojavljuju se za bilo koji ulazni parametar

**Obnavljajući ispad**: sustav se može oporaviti bez intervencije korisnika

**Korumpirajući ispad**: ispad dovodi do pogrešnog stanja sustava ili podataka

**Koraci za specifikaciju pouzdanosti**: **1.** za svaki podsustav analizirati posljedice mogućih ispada; **2.** napraviti podjelu ispada po klasama; **3.** za svaku identificiranu klasu postaviti pouzdanost koristeći prikladnu metriku; **4.** identificirati funkcijske zahtjeve za pouzdanost kako bi se smanjila vjerojatnost kritičnih ispada

**Određivanje OP**: za postojeće sustave (na temelju prošlih podataka); za inovativne sustave nemoguće razviti pouzdan OP; u određivanju sudjeluju (programski inženjeri sustava; dizajneri visoke razine; izvođači testiranja; ključni korisnici); progresivno sužavanje perspektive – od korisnika prema stvarnom radu; treba izračunati koliko često će se neki ulazi koristiti i pripadne vjerojatnosti; OP generirati automatski gdje je to moguće

**Statističko testiranje**: odnosi se na testiranje za određivanje pouzdanosti (nije isto kao testiranje za isključivo pronalaženje grešaka); test podaci trebaju biti u skladu s predviđenim profilom korištenja programske opreme; jedna od mjera je brojanje pronađenih neispravnosti; specificira se prihvatljiva razina pouzdanosti – program se testira i mijenja dok se to ne dostigne

**Koraci statističkog testiranja**: **1.** odrediti OP; **2.** generirati skup test podataka u skladu s OP; **3.** testirati, mjeriti vrijeme izvođenja između pojedinih ispada programa; **4.** nakon određenog broja testiranja (određen statistički) može se odrediti pouzdanost

**Problemi vezani uz statističko testiranje**: **Nesigurnost OP** (problem izražen kod razvijanja do tad nepostojećih prog. proizvoda); **Veliki troškovi određivanja OP** (troškovi ovise o tipu informacije koja je potrebna); **Statistička nesigurnost kod zahtjeva za visokom pouzdanošću** (teško odrediti konfidentnu razinu u OP; uzorak korištenja programa može se mijenjati tijekom njegove eksploatacije)

**Osiguranje minimalnog broja neispravnosti SW**: dobro definirati ponovljiv proces; ne oslanja se u potpunosti na vještine sudionika nego na adekvatan način vrši podjelu poslova među korisnicima; uključuje značajnu verifikaciju i validaciju

**Procesne aktivnosti validacije**: inspekcija zahtjeva; upravljanje zahtjevima; provjera modela; inspekcije koda i dizajna; statička analiza; planiranje i upravljanje testiranjem; upravljanje konfiguracijom

**Modeli pouzdanosti programske opreme**: poznato oko 200 modela pouzdanosti; još nerješeno kako jedinstveno kvantificirati SW-R; ne postoji jedinstven model; ppojedini modeli za pojedine tipove programske opreme; većina modela (pretpostavke, faktori, matematičke relacije)



**Najčešći modeli pouzdanosti**: osnovni i log Poissonov model

**Pretpostavke**: ispadi programa modeliraju se kao nehomogeni Poissonov proces; vrijeme – procesorsko vrijeme izvođenja; ispadi međusobno nezavisni; neispravnosti koje su prouzročile ispade uklanjaju se tijekom testiranja

**Osnovni model**: konačan broj ispada u konačnom vremenu; promjena intenziteta ispada po svakom opaženom ispadu je konstantna; bolji za veći skup podataka

**Log Poissonov model**: konačan broj ispada u beskonačnom vremenu; promjena intenziteta ispada po svakom opaženom ispadu je eksponencijalna; predviđanja dobivena bolja za ranu fazu testiranja; više pesimističan

**Modeli povećanja pouzdanosti**: matematički model za opis promjene pouzdanosti prouzročene testiranjem programa i ispravljanjem pronađenih neispravnosti; koriste se za predviđanje pouzdanosti ekstrapolacijom postojećih podataka; ovise o statističkom testiranju

**Model jednolikog povećanja**: jednostavan model; često ne odgovara realnosti; pouzdanost nužno ne raste s uklanjanjem neispravnosti; brzina porasta pouzdanosti tijekom vremena korištenja programa pada