P9 Testiranje

1 Uvod

1.1 Testiranje programske opreme

Namen testiranja je pokazati, da program deluje tako, kot je bilo predvideno in da se napake programa odkrijejo, še preden se začne uporabljati.

1.2 Cilji testiranja programske opreme

- razvijalcu oz. stranki želimo pokazati, da programska oprema ustreza njihovim zahtevam
- odkrivanje situacij, kjer je delovanje programske opreme nepravilno, nezaželeno ali ni v skladu s specifikacijo

1.3 Vrednotenje in testiranje napak

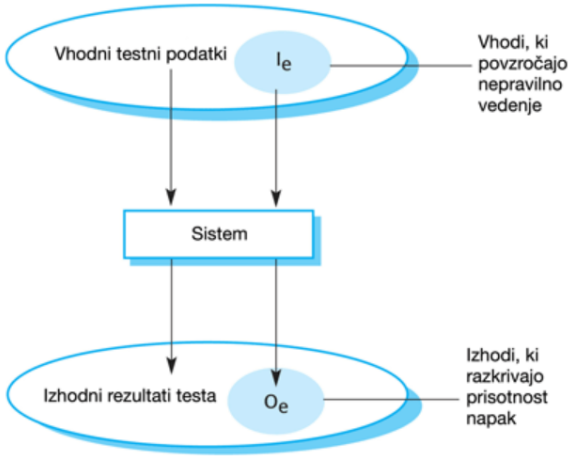
Prvi cilj je povezan z **vrednotenjem**, kjer se pričakuje, da bo sistem pravilno izvajal določeno množico testnih primerov.

Drugi cilj je povezan z **testiranjem napak**, kjer so testni sistemi zasnovani tako, da izpostavljajo napake.

1.4 Cilji procesa testiranja

Vrednotenje - uspešno izveden test potrjuje, da sistem deluje skladno z načrtom.

Testiranje napak - uspešno izveden test je takšen, ki pripravi sistem do nepravilnega delovanja in tako izpostavi napako v sistemu.



Slika 17.1: Vhodno-izhodni model testiranja programa

Figure 1: Vhodno-izhodni model testiranja programa

1.5 Preverjanje in vrednotenje

Pri **preverjanju** oz. **verifikaciji** se vprašamo, ali **pravilno gradimo sistem**.

Pri **vrednotenju** oz **validaciji** se vprašamo, ali **gradimo pravilni sistem**.

1.6 Zaupanje v preveranje in vrednotenje

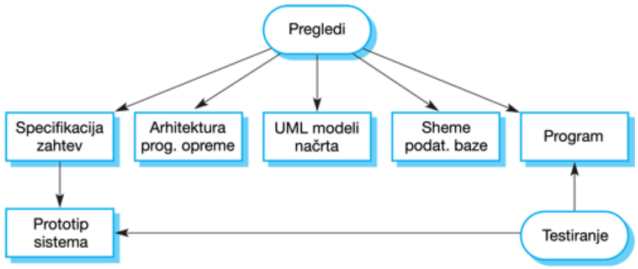
Cilj preverjanja in vrednotenja **(V & V)** je vzpostaviti zaupanje, da je sistem primeren za njegov namen, kjer je zaupanje odvisno od:

- **namena programske opreme**
- **pričakovanja uporabnikov**
- **tržnega okolja**

1.7 Pregledi in testiranje

Pregledi programske opreme se pri odkrivanju težav ukvarjajo z analizo statične predstavitve sistema (**statično preverjanje**).

Testiranje programske opreme se ukvarja z izvajanjem in opazovanjem vedenja sistema (**dinamično preverjanje**).



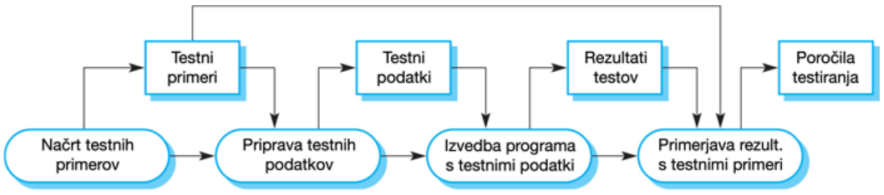
Slika 17.2: Preverjanje in testiranje

Figure 2: Preverjanje in testiranje

1.7.1 Pregledi programske opreme

Ne zahtevajo izvajanja sistema, zato se lahko opravijo še **pred** implementacijo in se lahko uporabijo za poljubno predstavitev sistema *(zahteve, načrt, konfiguracijski podatki, testni podatki, . . .)*.

1.7.2 Testiranje programske opreme



Slika 17.3: Model **procesa testiranja** programske opreme

Figure 3: Model procesa testiranja programske opreme

Pri testiranju programske opreme poznamo več **stopenj**:

- pri **razvojnem testiranju** se sistem preizkuša med razvojem z namenom odkrivanja napak
- pri **testiranju izdaje** ločena skupina za testiranje preizkusi celotno različico sistema pred izdajo končnim uporabnikom

- pri **uporabniškem testiranju** uporabniki ali potencialni uporabniki sistema preizkušajo sistem v svojem okolju

2 Razvojno testiranje

Razvojno testiranje vključuje vse aktivnosti testiranja, ki jih izvaja razvojna skupina:

- **testiranje enot** se osredotoči na **testiranje funkcionalnosti objektov** ali **metod**
- **testiranje komponent** se osredotoči na **testiranje komponentnih vmesnikov**
- **testiranje sistema** se osredotoči na **testiranje interakcij med komponentami**

2.1 Testiranje enot

Testiranje enot je proces izolacije posameznih komponent in testiranja napak, kjer **enote** lahko klasificiramo kot:

- **posamezne funkcije** ali **metode** v okviru objekta
- **razrede objektov**
- **sestavljene komponente** z določenimi vmesniki, ki se uporabljajo za dostop do njihove funkcionalnosti

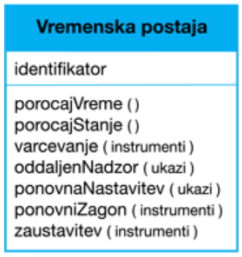
2.1.1 Testiranje razredov objektov

Testiranje razredov objektov vključuje popolno pokritost razreda s testi:

- testiranje vseh operacij, povezanih z objektom
- nastavitev in pregledovanje vseh atributov objekta
- izvajanje objekta v vseh možnih stanjih

2.1.2 Testiranje vremenske postaje

Na sliki 17.4 je prikazan vmesnik objekta vremenske postaje. Za testiranje je treba opredeliti testne primere za **poročanje o vremenu**, **nastavitvi**, **preverjanju stanja**, **zagonu** in **zaustavitvi**.



Slika 17.4: Vmesnik objekta vremenske postaje

Z diagramom stanj določimo zaporedje prehodov stanja, ki jih je treba testirati, in zaporedje dogodkov, ki te prehode povzročijo, npr.:

- zaustavitev → izvajanje → zaustavitev,
- nastavitev → delovanje → preverjanje stanja → prenos podatkov → delovanje,
- delovanje → zbiranje podatkov → delovanje → priprava povzetka → prenos podatkov → delovanje.

2.1.3 Samodejno testiranje

Kjer je le mogoče, uporabimo **samodejno testiranje**, ki je sestavljeno iz treh delov:

- **namestitev**, kjer inicializiramo sistem s testnim primerom, in sicer določimo vhode ter pričakovane izhode
- **izvajanje**, kjer pokličemo objekt ali metodo, ki jo želimo testirati

- **potrjevanje**, kjer primerjamo rezultat klica s pričakovanim rezultatom. Če je prisotno ujemanje, je test uspešen, sicer je neuspešen.

2.2 Izbira testnih primerov

Imamo dve vrsti testnih primerov:

- prva vrsta testnih primerov mora odražati **normalno delovanje sistema** in pokazati, da komponenta deluje po pričakovanjih
- druga vrsta testnih primerov pa mora temeljiti na izkušnjah pri testiranju o tem, kje se pojavijo **pogoste težave**

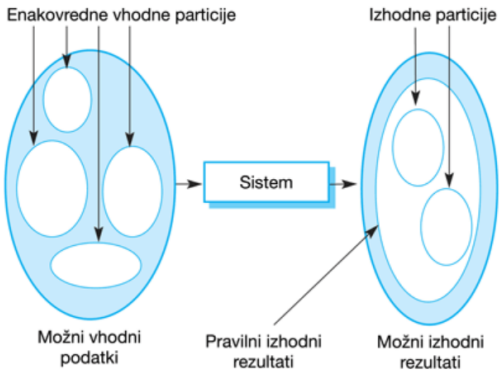
2.2.1 Strategije testiranja

Poznamo dve strategiji testiranja:

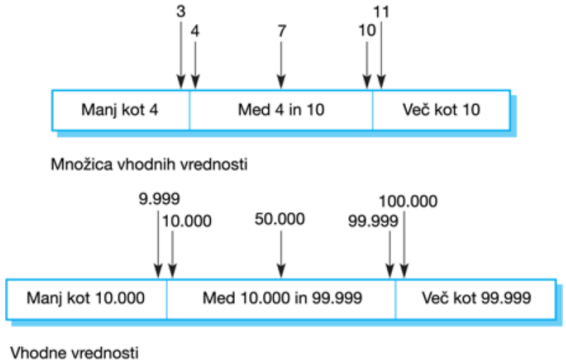
- **testiranje particij**, kjer določimo skupine vhodnih podatkov, ki imajo skupne značilnosti in jih je treba obdelati na enak način
- **testiranje na podlagi smernic**, kjer se pri izbiri testnih primerov za testiranje uporabijo smernice

2.2.2.1 Testiranje particij

Pri **testiranju particij** so vhodni podatki in izhodni rezultati pogosto iz različnih razredov, kjer pa so člani razreda medsebojno povezani.



Slika 17.5: Porazdelitev vhodnih podatkov v enakovredne particije



Slika 17.6: Enakovredne particije

2.2.2.1 Testiranje na podlagi smernic

Splošne smernice pri testiranju so:

- izbira vhodov, ki prisilijo sistem, da zgenerira vsa možna sporočila o napakah
- uporaba vhodnih podatkov, ki povzročijo prekoračitev pomnilnika
- ponovitev enakih vhodnih podatkov ali nizov vhodnih podatkov
- generiranje neveljavnih izhodnih rezultatov

Pogosto uporabljena smernica je **testiranje zaporedij**.

2.3 Testiranje komponent

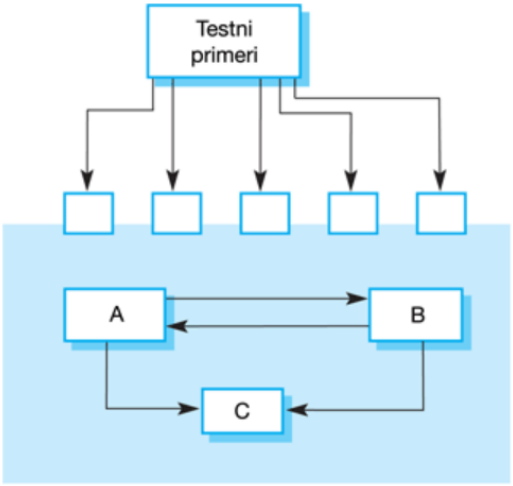
Komponente programske opreme so pogosto sestavljene iz več medsebojno povezanih objektov.

2.3.1 Testiranje vmesnika

Pri **testiranju vmesnika** je cilj odkrivanje napak vmesnika ali napak zaradi neveljavnih predpostavk o vmesniku.

Poznamo več vrst vmesnikov:

- **vmesnik parametrov** - *podatki se prenašajo od ene metode ali procedure do druge*
- **skupni pomnilniški vmesnik** - *pomnilniški blok se deli med procedurami ali funkcijami*
- **postopkovni vmesnik** - *podsistem združuje nabor procedur, ki kliče drug podsistem*
- **vmesnik za posredovanje sporočil** - *podsistemi zahtevajo storitve drugih podsistemov*



Slika 17.7: Testiranje vmesnika

2.3.2 Napake vmesnika

- **napačna uporaba vmesnika** - *komponenta pokliče drugo komponento in naredi napako pri uporabi vmesnika (npr. parametri v napačnem vrstnem redu)*
- **nerazumevanje vmesnika** - *komponenta pri klicu upošteva določene predpostavke o delovanju druge komponente, ki so napačne*
- **časovne napake** - *komponenti delujeta z različno hitrostjo in tako se dostopa do neažurnih informacij*

2.4 Testiranje sistema

Testiranje sistema med razvojem vključuje integracijo komponent pri izdelavi različice sistema in nato testiranje integriranega sistema.

2.4.1 Testiranje integracije komponent

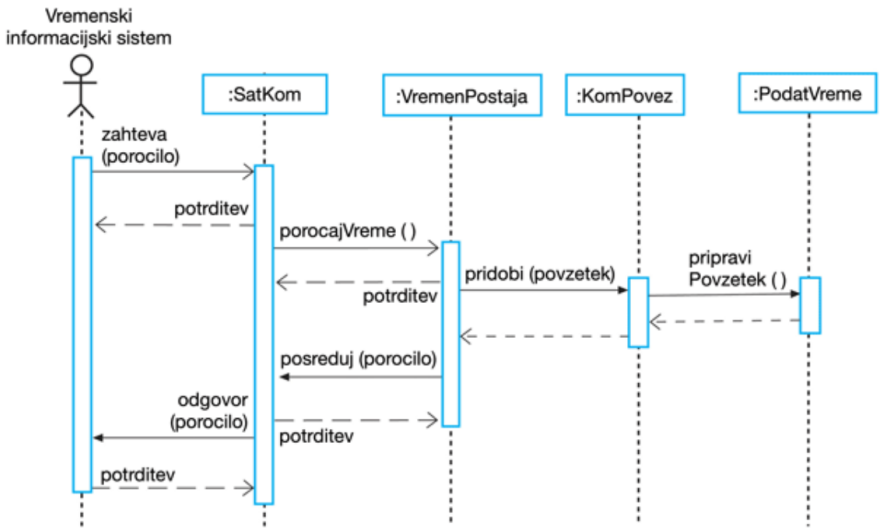
Med testiranjem sistema se lahko z novo razvitimi komponentami integrirajo komponente za večkratno uporabo in gotovi sistemi; nato pa se testira celoten sistem.

2.4.2 Testiranje primerov uporabe

Primeri uporabe, ki so nastali za identifikacijo interakcije s sistemom, se lahko uporabijo za osnovo testiranja sistema.

2.4.2.1 Testni primeri, izpeljani iz diagrama zaporedja

Kot prikazuje primer diagrama zaporedja za zbiranje podatkov vremenske postaje na sliki 17.8, je vnos zahteve po pridobitvi poročila povezan s posredovanjem potrditve. Na podlagi te zahteve se kot odgovor pričakuje poročilo. Pripraviti je treba povzetek podatkov, s katerimi preverimo, ali je poročilo ustrezno sestavljeno.



Slika 17.8: **Diagram zaporedja** za zbiranje podatkov vremenske postaje

Vhodna zahteva po poročanju vremena v okviru `VremenPostaja` pričakuje kot rezultat generiran povzetek poročila. To lahko testiramo s pripravo neobdelanih podatkov, ki ustrezajo povzetku, pripravljenih za testiranje `SatKom` in preverjanje, ali `VremenPostaja` pravilno izdela povzetek. Ti neobdelani podatki se uporabljajo tudi za testiranje objekta `PodatVreme` .

2.4.3 Politike testiranja

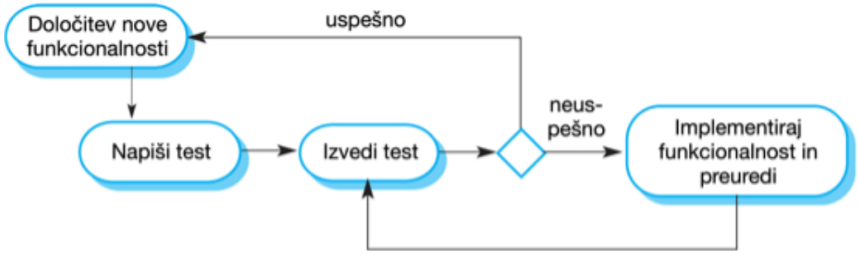
Primeri politik testiranja:

- testirati je treba **vse sistemske funkcije**, do katerih je mogoče **dostopati preko menijev**
- tesitrati je treba **kombinacije funkcij**, do katerih se lahko dostopa preko istega menija
- ko se zahteva **uporabniški vnos podatkov**, je treba testirati vse funkcije s **pravilnim** in **nepravilnim** vnosom

3 Testno usmerjen razvoj (TDD)

Testno usmerjen razvoj (TDD) je pristop k izdelavi programske opreme, kjer se testiranje in razvoj izvorne kode prepletata. Testi so napisani pred samo kodo in potrditev testov je ključno gonilo razvoja.

3.1 Aktivnosti proces TDD



Slika 17.9: Testno usmerjen razvoj (TDD)

Figure 4: Testno usmerjen razvoj (TDD)

3.2 Prednosti TDD

- **Pokritost s testi** - *vsak segment kode, ki ga napišemo, ima vsaj en povezan test, tako da ima vsa napisana koda vsaj en test.*
- **Regresijsko testiranje** - *zbirka regresijskih testov se razvija postopoma, skupaj z razvojem programa.*
- **Poenostavljeno iskanje napak** - *ob neuspešnem testu je jasno, kje je težava.*
- **Sistemska dokumentacija** - *sami testi predstavljajo določeno obliko dokumentacije, ker opisujejo, kako naj bi program deloval.*

3.2.1 Regresijsko testiranje

Regresijsko testiranje je testiranje sistema, kjer se preveri, da vpeljane spremembe niso *pokvarile* delovanje prej delujoče kode.

4 Testiranje izdaje

Testiranje izdaje je proces testiranja določene izdaje sistema, ki je namenjen za uprabo zunaj razvojne skupine.

4.1 Testiranje izdaje in sistema

Testiranje izdaje je določena oblika testiranja sistema, vendar obstajajo pomembne razlike:

- za **testiranje izdaje** mora biti odgovorno **ločena skupina**, ki ni bila vključena v razvoj sistema
- **testiranje sistema** s strani razvojne skupine se mora osredotočiti na **odkrivanje napak v sistemu**. Cilj **testiranja izdaje** pa je preveriti, ali **sistem izpolnjuje zahteve** in je dovolj dober za zunanjo uporabo (validacija)

4.2 Testiranje na podlagi zahtev

Testiranje, ki temelji na zahtevah, vključuje preučevanje vsake zahteve in pripravo ustreznih testov.

17.4.2.1 Primer zahtev v sistemu Mentcare

- Če ima bolnik alergijo na določeno zdravilo, bo sistem ob predpisovanju tega zdravila zdravnika opozoril s sporočilom.
- Če se zdravnik, ki je predpisal zdravilo, odloči, da ne bo upošteval opozorila o alergijah, mora navesti razlog, zakaj je opozorilo prezrl.

17.4.2.2 Primer testov v sistemu Mentcare

- Izberi bolnika brez znanih alergij. Predpiši zdravilo, za katero obstajajo alergije. Preveri, da sistem ne izda opozorilnega sporočila.
- Izberi bolnika z znanimi alergijami. Predpiši zdravilo, na katero je bolnik alergičen, in preveri, ali je sistem izdal opozorilo.
- Kreiraj kartoteko bolnika, kjer so dodane alergije na dve ali več zdravil. Predpiši obe zdravili vsako posebej in preveri, ali je za vsako zdravilo izdano pravilno opozorilo.
- Predpiši dve zdravili, na kateri je bolnik alergičen. Preverite, ali sta pravilno izdani obe opozorili.
- Predpiši zdravilo, ki bo povzročilo opozorilo, in ga nato zavrnite. Preverite, ali sistem od uporabnika zahteva pojasnilo, zakaj se opozorilo ni upoštevalo.

4.3 Testiranje na podalgi scenarijev

Testiranje, ki temelji na scenarijih, je pristop testiranja izdaj, kjer se izdela tipišne scenarije uporabe, na podlagi katerih se razvije tesne primere.

17.4.3.1 Scenarij uporabe v sistemu Mentcare

Jasna je medicinska sestra, ki je specializirana na področju duševnega zdravja. Ena od njenih nalog je, da obišče bolnike na domu, kjer preveri, ali je njihovo zdravljenje učinkovito in da ne trpijo zaradi stranskih učinkov zdravil.

Na dan opravljanja obiskov na domu se Jasna prijavi v sistem Mentcare in ga uporabi za tiskanje urnika obiskov za ta dan skupaj s povzetkom informacij o bolnikih, ki jih je treba obiskati. Zahteva tudi, da se zdravniške kartoteke teh bolnikov prenesejo na njen prenosni računalnik. Sistem od nje zahteva geslo, ki se uporabi za šifriranje podatkov na njenem prenosnem računalniku.

Eden od bolnikov, ki jih obišče na domu, je Bogomil, ki se zdravi z zdravili za depresijo. Bogomil verjame, da mu zdravilo pomaga, vendar je prepričan, da ima neželeni stranski učinek, da ponoči ne more spati. Jasna poišče Bogomilovo zdravniško datoteko, jo odpre in sistem jo vpraša za geslo za dešifriranje podatkov. Jasna preveri predpisano zdravilo in pogleda njegove stranske učinke. Izkaže se, da je nespečnost že znani neželeni stranski učinek, zato opozori na problem v obliki zaznamka v Bogomilovo zdravniško kartoteko in mu predlaga, da obišče ordinacijo s prošnjo, da se zdravilo zamenja. Bogomil se s predlogom strinja in se z Jasno dogovori, da ga pokliče, ko se bo Jasna v ordinaciji dogovorila z zdravnikom za Bogomilov obisk. Jasna zaključi posvet in sistem ponovno šifrira posodobljeno Bogomilovo zdravniško kartoteko.

Po končanem posvetu se Jasna vrne v ordinacijo in v podatkovno bazo naloži posodobljene zapise o obiskanih bolnikih. Sistem pripravi seznam telefonskih klicev, ki jih mora opraviti Jasna, da obvesti bolnike o nadaljnjem spremljanju na domu in obiskih v ordinaciji.

17.4.3.2 Funkcije, ki jih scenarij testira

- Preverjanje pristnosti uporabnika s prijavo v sistem,
- prenos določenih zdravniških kartotek bolnikov v prenosnik in nazaj v sistem,
- načrtovanje obiska na domu,
- šifriranje in dešifriranje zdravniških kartotek bolnikov na mobilni napravi,
- iskanje in spreminjanje zdravniške kartoteke,
- dostop do informacij o stranskih učinkih zdravil in medsebojnih vplivih,
- sistem za upravljanje pozivov bolnikov.

4.4 Testiranje zmogljivosti

Testi zmogljivosti običajno vključujejo množico testov, kjer se obremenitev počasi povečuje, dokler delovanje sistema ne postane nesprejemljivo.

Testiranje izjemnih situacij je oblika testiranja zmogljivosti, kjer je sistem namerno preobremenjen in se poskuša doseči neustrezno delovanje.

5 Uporabniško testiranje

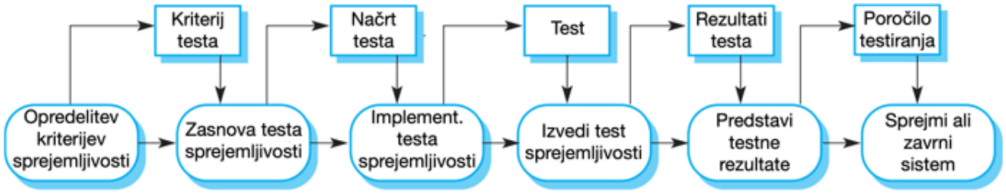
Uporabniško testiranje je stopnja v procesu testiranja, kjer uporabniki ali naročniki zagotovijo vhodne podatke in nasvete o testiranju sistema.

5.1 Vrste uporabniškega testiranja

Pri **alfa testu** uporabniki sodelujejo z razvojno ekipo tako, da pri razvijalcih testirajo programsko opremo

Pri **beta testu** je uporabnikom na voljo izdaja programske opreme, ki jim omogoča eksperimentiranje in povzročanje potencialnih težav, ki jih odkrivajo skupaj z razvijalci.

Pri **testu sprejemljivosti** stranka testira sistem z namenom odločanja, ali je le-ta sprejemljiv za uvedbo v okolje stranke.



Slika 17.10: Proces testa sprejemljivosti

5.1.1 Agilne metode in test sprejemljivosti

Pri agilnih metodah je uporabnik oz. stranka del razvojne skupine in je odgovorna za sprejemanje odločitev o sprejemljivosti sistema.

6 Zaključne ugotovitve

- Testiranje lahko samo pomaga razkriti prisotnost napak v programu, ne more pa potrditi, da ni prisotnih napak.
- Za **razvojno testiranje** je odgovorna ekipa za razvoj programske opreme. Ločena testna skupina pa je odgovorna za **testiranje sistema**, preden se ta izda stranki.
- **Razvojno testiranje** vključuje: testiranje enot, kjer testiramo posamezne objekte in metode; testiranja komponent, kjer testiramo sorodne skupine objektov, in sistemsko testiranje, kjer testiramo delne ali popolne sisteme.
- Pri testiranju programske opreme morate poskusiti “razbiti” programsko opremo z uporabo izkušenj in smernic, tako da izberete testne primere, ki so v preteklosti bili učinkoviti pri odkrivanju napak v drugih sistemih.
- Samodejne teste pišemo kadarkoli je mogoče. Takšne teste lahko samodejno izvaja program, ki se pokliče ob vsaki spremembi sistema.
- **TDD** je pristop k razvoju, kjer so testi napisani pred kodo.
- Pri testiranju na podlagi scenarijev se pripravijo tipični scenariji uporabe, ki so osnova pri pripravi in izvedbi testnih primerov.

- **Test sprejemljivosti** je vrsta uporabniškega testiranja, kjer se odloča, ali je programska oprema primerna za uporabo v produkcijskem okolju.