KI206 - TESTOVI

Ispit

1. Šta je softversko inženjerstvo? ***

   – **Softversko inženjerstvo** je inženjerska disciplina koja se bavi razvojem i proizvodnjom softvera (tj. svim aspektima proizvodnje softvera).

2. Šta čini softverski proces? Koje su četiri osnovne aktivnosti koje su prisutne u svim modelima softverskih procesa? ***

   – **Softverski proces** je redosled aktivnosti koji vodi ka proizvodnji softverskog proizvoda. Čine ga 4 osnovne aktivnosti. **Osnovne aktivnosti** su:
   - Specifikacija softvera,
   - Razvoj softvera,
   - Validacija softvera,
   - Evaluacija softvera

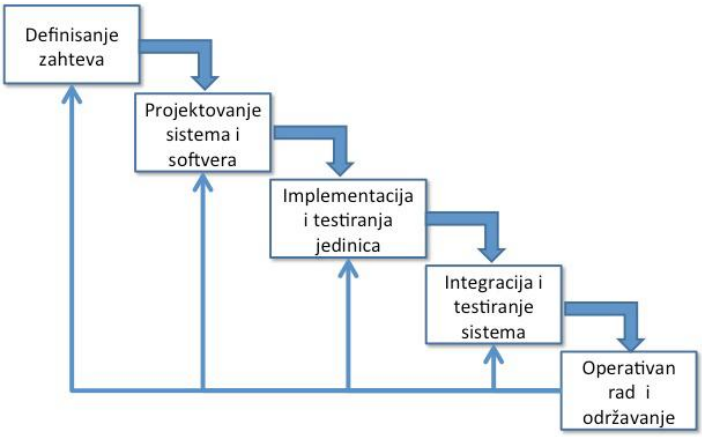
3. Koja su četiri važna atributa (svojstva) koja moraju da poseduju svi profesionalni softveri? Objasnite svaki od navedena četiri atributa? ***

   – **Važni atributi** su:
   - Sposobnost održavanja – softver mora biti tako napisan da može da se lako menja da bi se mogli zadovoljiti zahtevi korisnika za njegovom promenom kada za to dođe vreme.
   - Pouzdanost i sigurnost – znači da sistem u slučaju pada ne bi trebalo da prouzrokuje nikakvu fizičku ili ekonomsku štetu. Takođe i neovlašćenim licima treba onemogućiti da pristupe sistemu i u njemu naprave neku štetu.
   - Efikasnost – znači da sistem ne sme beskonačno trošiti resurse. I sistem mora biti takav da ima dobar odziv, vreme obrade, upotrebu memorije, … i slično.
   - Prihvatljivost – softver mora biti prihvatljiv za one kategorije korisnika za koje je i projektovan. Mora biti razumljiv, koristan i kompatibilan sa sistemima koje korisnici upotrebljavaju.

4. Navedite četiri pravila struke koje inženjeri softvera treba da poštuju? ***
   – **Četiri pravila struke** su:
   - Poverljivost (poverljive informacije se moraju čuvati),
   - Kompetencija (ljudi koji nisu kompetentni se ne smeju prihvatiti posla),
   - Prava intelektualne svojine (moraju se poštovati),
   - Zloupotreba kompjutera (ne sme da se učini od strane softverskog inženjera).

5. Nacrtajte model vodopada koji se koristi za razvoj softvera. Kada se najčešće koristi metod vodopada? ***

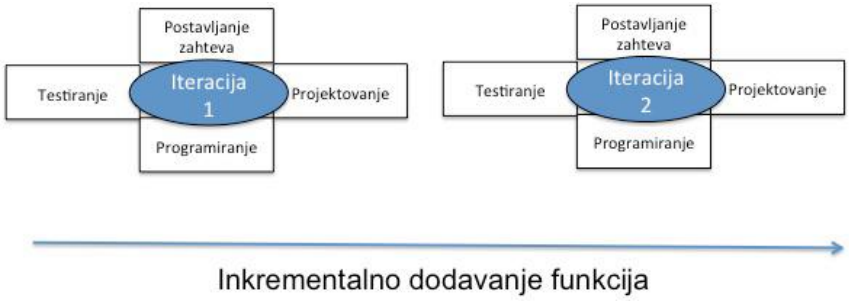
   – **Model vodopada**:



   - Model vodopada se najčešće koristi samo u slučajevima kada su svi zahtevi dobro definisani, jasni i stabilni.

6. Nacrtajte model procesa agilnog razvoja softvera i navedite četiri osnovne vrednosti na koje se oslanja agilni proces razvoja softvera? ***

   – **Model procesa agilnog razvoja**:



   - **Vrednosti** na koje se oslanja agilni proces razvoja softvera su:
     - Programeri i njihove interakcije su važniji od procesa i alata,
     - Softver koji radi je važniji od kompletnosti dokumentacije,
     - Saradnja sa kupcem je važnija nego ugovaranje,
     - Reagovanje na promene je važnije od postavljenog plana.

TEST 1

1. Zašto je softversko inženjerstvo važno? (Označiti dva najvažnija razloga). ***

   – Neophodnost ekonomičnog razvoja pouzdanog softvera za što kraće vreme,
   – Smanjuju se troškovi promene softvera, koji su najčešće i najveći troškovi.

2. Šta od navedenog NE spada u standard prihvatljivog ponašanja softverskih inženjera? ***

   – Odgovornost – Morate se maksimalno posvetiti razvoju kvalitetnog softvera i preuzeti odgovornost za sve moguće greške prilikom razvoja.

3. Koji su ključni izazovi softverskog inženjerstva? (Označiti tri najvažnija). ***

   – **Izazovi** su:
     - Rad u heterogenom okruženju,
     - Bezbednost informacija,
     - Brzo prilagođavanje novim zahtevima.

4. Koji su troškovi izraženi u softverskom inženjerstvu? ***

   – 60% su troškovi razvoja, a 40% troškovi testiranja. Za softver razvijan po narudžbini, troškovi evolucije često nadmašuju troškove razvoja.

5. Koje dve vrste softverskih proizvoda postoje? ***

   – Vrste su:
     - Generički proizvodi,
     - Proizvodi razvijeni po narudžbini.

TEST 2

1. Šta je softverski proces?

   - **Softverski proces** je skup povezanih aktivnosti koji vodi proizvodnji softverskog proizvoda. Ove aktivnosti mogu dovesti do razvoja potpuno novog softvera ili do usavršavanja nekog postojećeg.

2. Kako možemo podeliti softverske procese?

   - Softverski procesi se mogu **podeliti na dva tipa**:
     - Procese vođene planom
     - Agilne procese

3. Kako možemo definisati procese vođene planom a kako agilne procese?

   - **Procesi vođeni planom** su procesi kod kojih su sve aktivnosti unapred planirane i napredak u razvoju softvera se određuje stepenom ostvarivanja tog plana.
   - **Agilni procesi** su procesi kod kojih se planiranje radi postupno (inkrementalno) kako bi se procesi lakše prilagodili promenljivim zahtevima korisnika.

4. Koja tri opšta modela softverskih procesa se najčešće navode?

   - Najčešće se navode:
     - Model vodopada,
     - Inkrementalni (postepeni) razvoj,
     - Softversko inženjerstvo zasnovano na višestrukoj upotrebljivosti.

5. Navesti prednosti i nedostatke razvoja softvera upotrebom komponenata?

   - **Prednosti** su da ovaj model razvoja smanjuje količinu softvera koji treba da bude razvijen i smanjuje troškove i rizike, što vodi ka bržoj isporuci softvera.
   - **Nedostatak** što su potrebni kompromisi zahteva i ovo može voditi ka sistemu koji neće u potpunosti ispuniti stvarne potrebe korisnika, jer se upotrebljavaju unapred definisane komponente. Takođe, gubi se i kontrola nad evolucijom softvera sa upotrebom nižih verzija komponenata.

6. Navesti četiri glavne faze procesa inženjeringa zahteva?

   – **Glavne faze** su:
     - Studija izvodljivosti,
     - Izvođenje i analiza zahteva,
     - Specifikacija zahteva
     - Validacija zahteva.

7. Zbog čega je inženjering zahteva jako bitna faza u procesu razvoja softvera?

   – Zato što greške u ovoj fazi neizbežno vode ka kasnijim problemima u projektovanju i implementaciji sistema.

8. Kako razvijati softver u uslovima stalnih promena?

   – U uslovima stalnih promena, primenjuju se dva pristupa u razvoju:
     - Izrada prototipa softvera, radi provere zahteva.
     - Inkrementalni razvoj softvera.

9. Šta je inkrementalna isporuka softvera? Navesti jedan primer upotrebe inkrementalne isporuke softvera?

   – **Inkrementalna isporuka** softvera je isporuka u kojoj se naručiocu softvera isporučuju inkrementi sistema radi komenatarisanja i eksperimentisanja. Primer je Android aplikacija (seti se Theme.io-a).

10. Objasnite zašto je inkrementalni razvoj najefikasniji pristup za razvoj poslovnih softverskih sistema?

    – Zato što omogućava izbegavanje prevremenih i čestih promena i daje toleranciju promena. A time se izbegava prerano opredeljivanje da se nešto u sistemu menja, dok se to ne proveri na inkrementu. Takođe, umesto da promene obuhvate ceo sistem, one se odnose samo na inkrement što je znatno jeftinije.

11. Navesti primer upotrebe spiralnog modela razvoja?

    – Primer razvoja sistema za centralizaciju dokumenata na teritoriji jedne države.

12. Objasnite faze RUP-a?

- Rup model se bavi analizom rizika i podržava razvoj koji primenjuje slučajeve korišćenja.
  Faze su:
  1) **Početak** – cilj ove faze je postavljanje poslovnog scenarija sistema t.j. određivanje šta sistem treba da radi. Utvrđuju se spoljni akteri (ljudi i sistemi) koji su u interakciji sa sistemom.
  2) **Razrađivanje** – (elaboracija) cilj je razvoj razumevanja domena problema, postavljanje arhitektonskog okvira sistema, razvoj plana projekta i utvrđivanje ključnih rizika projekta.
  3) **Konstrukcija** – obuhvata projektovanje sistema, programiranje i testiranje.
  4) **Tranzicija** – je končna faza RUP-a koja se bavi prenosom sistema iz razvojnog okruženja u korisničko okruženje i stavlja ga u rad u stvarnom okruženju.

13. Šta je inkrementalna isporuka softvera? ***

- **Inkrementalna isporuka** softvera je pristup razvoju softvera koji omogućava da se kupcima isporučuju razvijeni delovi sistema koji se onda instaliraju i puštaju u operativnu upotrebu.

14. Kada inkrementalni pristup nije najbolje rešenje? (označite tri odgovora) ***

- Nije dobro rešenje kod:
  - Razvoja ugrađenih sistema koji zavise od hardvera.
  - Razvoja kritičnih sistema
  - Kada se sistem razvija na više lokacija.

15. Koji su nedostaci inkrementalnog razvoja softvera? ***

- **Nedostaci** su:
  - Teško je utvrditi osnovne funkcije koje će biti realizovane od strane svih inkremenata.
  - Korisnici često nisu voljni da eksperimentišu sa nekompletnim novim sistemom.
  - Sve do poslednjeg inkrementa ne postoji kompletna specifikacija sistema, što zahteva nove forme ugovora koje često ne odgovaraju nekim organizacijama.

16. Šta je Beta testiranje? ***

- **Beta testiranje** je testiranje koje se vrši kod grupe odabranih korisnika sistema, tj. u njihovom radnom okruženju.

17. Koja je razlika između faza i radnih tokova RUP-a? ***

- **Faze** su dinamičke i imaju svoje ciljeve. **Radni tokovi** su statički i predstavljaju tehničke aktivnosti koje nisu povezane sa pojedinačnom fazom, već se mogu upotrebiti za vreme celog razvoja radi ostvarivanja ciljeva svake faze.

18. Šta je Alfa testiranje? ***

- Testiranje koje vrši razvojni tim, sam ili sa naručiocem softvera.

19. Šta od navedenog NE spada u četiri osnovne faze razvoja softvera po RUP-u? (Označite 2 odgovora) ***

- Programiranje
- Integracija

20. Zašto je inkrementalni razvoj najefektivniji pristup u razvoju poslovnih softverskih sistema? ***

- Kod poslovnih sistema se često traže promene u softveru tokom njegovog razvoja. U tom slučaju, inkrementalni razvoj je jeftiniji i lakši za rad.

21. Kako se vrše testiranja u slučaju planskog razvoja softvera? ***

- Testiranje se vrše u skladu sa planovima testiranja, koje priprema nezavistan tim na osnovu specifikacije i projekta sistema.

22. Koje su dve osnovne vrste softverskih procesa? ***

- Softverski procesi se mogu podeliti na:
  - Procese vođene planom
  - Agilne procese

TEST 3

1. Koja tehnika prikupljanja zahteva je po vašem mišljenju najefikasnija?

   - Najefikasnija je Tehnika **Intervijua**. Zato što se uz pomoć ove tehnike prikupi najveći broj zahteva i to direktno od aktera u sistemu. Dobijeni odgovori daju priliku za postavljanje dodatnih pitanja, čime se utvrđuju i pribavljaju nove informacije. Dobri su za razumevanje kako će se sistem ponašati u realnim scenarijima upotrebe i šta sve korisnici očekuju od njega.

2. Da li se sekvencijalni dijagram može koristiti za grafičko predstavljanje scenarija slučaja upotrebe?

   - DA

3. Kako se određuje stepen detaljnosti dokumenta sa specifikacijom zahteva?

   - U zavisnosti od vrste sistema koji se razvija

4. Šta NE spada u opis UML slučaja korišćenja?

   - Kreiraju se u ranoj fazi da bi se razumeli zahtevi i način njihove realizacije.

5. Šta je kompleksnost zahteva?

   - Kompleksnost zahteva znači da su definisani svi servisi koje korisnik zahteva.

6. Zašto softverski inženjeri treba da razgovaraju sa korisnicima ili kupcima softvera?

   - Da bi utvrdili domen primene softvera, servise koje softver treba da obezbedi, zahtevane performanse, hardverska ograničenja i slično.

7. Šta su akteri softverskog sistema?

   - Svi koji imaju neki direktni ili indirektni interes na zahteve softverskog sistema.

8. Koje su teškoće u primeni metrike nefunkcionalnih zahteva?
   - Neka svojstva nije lako izmeriti, a i kada je to moguće korisnicima je teško da odrede poželjne vrednosti.

9. Šta je konzistentnost zahteva?
   - Konzistentnost zahteva znači da nema kontradiktornosti između zahteva.

10. Čime se bavi upravljanje zahtevima?
    - Razumevanjem i kontrolom promena u zahtevima sistema.

11. Šta NE spada u odluke koje treba doneti da bi se planski upravljalo zahtevima?
    - Upoznavanje sa organizacijama odakle dolaze akteri sistema.

TEST 4

1. Koja arhitektura je po vašem mišljenju najzastupljenija u web sistemima?
   - Servisno-orjentisana arhitektura.

2. Šta su aplikacioni sistemi aplikacije?
   - Softverski sistemi koji se koriste radi zadovoljenja poslovnih i organizacionih potreba neke firme.

3. Šta je koncepcijski pogled na arhitekturu softverskog sistema?
   - Apstraktan pogled na sistem koji može da služi kao osnova za dekompoziciju uopštenih zahteva na detaljnije specifikacije zahteva.

4. Šta su informacioni sistemi?
   - Sistemi koji koriste interakciju sa zajedničkom bazom podataka.

5. Koja je struktura sistema za transakcionu obradu? Redosled elemenata je bitan.
   - UI obrada, logika aplikacije, menadžer transakcije, baza podataka.

6. Da li je detaljna specifikacija arhitekture potrebna?
   - Zavisi od sistema (ili Da)

7. Šta je MVC šablon?
   - Arhitektura koja razvrstava sve podsisteme u tri kategorije: Model, View i Controller.

8. Šta NE spada u nefunkcionalne zahteve o kojima se mora voditi računa prilikom izbora arhitektonskog šablona?
   - Kohezija

9. Kakva je arhitektura kompajlera, tj. Prevodioca programskih jezika? Redosled je bitan.
   - Leksička analiza, Sintaksna analiza, Semantička analiza, Generisanje koda.

10. Zašto se mogu javiti konfliktne situacije pri projektovanju arhitekture u kojima su zahtevi za raspoloživost i bezbednost najvažniji nefunkcionalni zahtevi?
    - Raspoloživost zahteva redundantne komponente pa postaje teže obezbediti visok nivo zaštite.

11. Šta su aktivna skladišta podataka?
    - Skladišta koja mogu da podstaknu rad komponenti.

TEST 5

1. Koja je razlika između dijagrama komponenata i dijagrama klasa? Šta se može videti na dijagramu komponenti a nije prikazano kroz dijagam klasa sistema?

   - **Dijagram komponenata**, predstavlja jedan fizički pogled na sistem. Njegova svrha je da pokaže zavisnost između softvera i drugih softverskih komponenata u sistemu.
   - **Dijagram klasa,** prikazuje skup klasa, interfejsa i njihovih relacija, tj. opisuje strukturu sistema.
   - **Razlika** je u tome što Dijagram klasa prikazuje logičku strukturu apstrakcija (tj. softv. kompon.) a Dijagram komponenata prikazuje fizičku organizaciju i zavisnosti između skupa kompnenata. Dijagram komponenti omogućava prikaz fizičke komponente koja već postoji u računaru (za razliku od klasa koje su samo logičke apstrakcije).

2. Šta je to izvršni UML? ***

   - UML koji sadrži informacije potrebne za lakše pisanje izvršnog koda.

3. Šta je model softverskog sistema? ***

   - Model je apstrakcija, tj. uprošćenje sistema

4. Šta je modelima vođeno inženjerstvo? ***

   - Pristup razvoju softvera u kome su modeli a ne program glavni rezultati razvoja. Programski kod se automatski generiše iz modela.

5. Koji su argumenti protiv primene modelima vođenog inženjerstva? ***

   - Apstrakcija je dobra za diskusiju, ali ne mora biti dobra za implementaciju. Nezavisnost od platforme nije uvek bitan factor. Kod velikih sistema, glavni problem nije u implementaciji, već u inženjerstvu zahteva, bezbednosti, načinu integracije, i slično.

6. Zašto se UML dijagrami strukture koriste? ***

   - Radi pokazivanja zavisnosti sistema i drugih softverskih komponenti u sistemu.

TEST 6

1. Objasniti preslikavanje sekvencijalnog modela u dijagram klasa. Na osnovu kog UML modela je moguće generisati programski kod?

   – Nakon kreiranja sekvencijalnog dijagrama, identifikovani objekti postaju klase a poruke postaju metode u dijagramu klasa. Na osnovu klasnog dijagrama koji je prethodno nastao od sekvencijalnog dijagrama vrši se generisanje Java programskog koda (tj. Java klasa).

2. O čemu se mora voditi računa pri promeni razvojnih platformi? ***

   – O tome kako će se kasnije softver preneti na izvršnu platformu.

3. Šta su to integrisane razvojne platforme (IDE)? ***

   – Skup alata koji podržavaju različite aspekte razvoja softvera?

4. Koje su specifičnosti razvoja ugrađenih softverskih sistema? ***

   – Kod ugrađenih sistema je poznata izvršna platforma, ali često je potrebno simulirati rad hardverskih komponenti sistema.

5. Navedite specifičnosti razvoja sa posredničkim softverom (middleware)? ***

   – Ovaj softver mora da se koristi prilikom testiranja i to se vrši na izvršnoj platformi, često zbog problema sa licencama.

6. Šta NE spada u troškove primene ponovljivog softvera, odnosno ranije razvijenog softvera/softverskog proizvoda? ***

   – Troškovi implementacije procesa razvoja novih komponenti.

TEST 7

1. Da li testiranje jednice ubrzava ili usporava proces testiranja softvera? Šta možemo saznati iz dobijenih rezultata testiranja jedinice?

   – Iz dobijenih rezultata (jediničnih testova) možemo utvrditi da određene metode izračunavaju tačno ono što je korisnik tražio. Tako da sa većom pouzdanošću možemo reći da sistem radi ono što treba da radi. Testiranje jedinice usporava proces testiranja pa i ukupno razvoja softvera.

2. Šta je Regresiono testiranje?

   – Ponavljanje ranijih testova posle promene softvera.

3. Koja je razlika između testiranja sistema i testiranja komponenti?

   – Testiranje komponenti se fokusira na interfejs jedne komponente, a testiranje sistema na interakciju između komponenti.

4. Zašto je korisničko testiranje potrebno?

   – Stvarni zahtevi korisnika ne moraju da se poklapaju sa specifikacijom.

5. Šta je korisničko testiranje?

   – Testiranje koje vrši korisnik kako bi odlučio da li prihvata proizvod.

6. Koliko se zahteva proverava jednim scenarijom?

   – Više zahteva.

7. Šta je cilj testiranja softvera za isporuku?

   – Da proizvođač uveri sebe da je softver spreman za isporuku.

8. Šta je “Stres” testiranje?

   – Generisanje mnogo više poruka od očekivanog broja

9. Ko vrši testiranje sistema spremnog za isporuku?

   - Tim koji nije uključen u razvoj.

10. Kada se koristi razvoj softvera vođen testovima?

    - Kod razvoja novog softvera

11. Koja je razlika između alfa testiranja, beta testiranja i testa prihvatanja softvera?

    - Alfa testiranje se vrši u razvojnom okruženju, beta testiranje se vrši kod korisnika, a test prihvatanja je formalan proces gde korisnik odlučuje da li prihvata proizvod.

TEST 8

1. Šta podrazumeva Evolucija softvera? Ko predlaže promene koje iniciraju evoluciju softvera?

   - **Evolucija softvera** je stalna promena softvera posle njegovog inicijalnog razvoja i isporuke naručiocu ili tržištu. Kompanija koja je razvila softver najčešće predaje odgovornost za dalji razvoj i modifikacije softvera kompaniji koja ga je naručila i koja mora da ima odgovarajući kadar za takav razvoj. Takođe, postoji mogućnost i da kompanija angažuje neku drugu organizaciju da podržava evolutivan razvoj njenog softvera.

2. Koje su specifičnosti evolucije softvera kod primene agilnih metoda razvoja softvera?

   - Evolucija samo nastavlja proces agilnog razvoja.

3. Kada se primenjuje restruktuiranje projektnog rešenja?

   - Kada restruktuiranje koda nije dovoljno jer je struktura previše degradirana.

4. Šta je reinženjering softvera?

   - Poboljšavanje strukture i razumljivosti sistema kako bi se olakšalo održavanje.
   - Pisanje koda ponovo u modernom jeziku kako bi se olakšalo održavanje.

5. Kada se publikuje (objavljuje) novo izdanje softvera?

   - Kada su promene izvršene i potvrđene.

6. Kakvi su troškovi održavanja u odnosu na troškove razvoja softvera?

   - Veći

7. Kada se prihvataju predložene promene?

   - Posle procene troškova i efekta promene

8. Kako se troškovi održavanja mogu smanjivati?
   - Softver se razvija tako da se što lakše održava.

9. Šta nije tip održavanja softvera?
   - Tehnička podrška korisnicima.

10. Koja je korist od primene reinženjeringa softvera?
    - Smanjuju se rizici i troškovi u odnosu na razvoj novog sistema.

11. Šta ne spada u faktore za utvrđivanje poslovne vrednosti softverskog sistema?
    - Brzina otkaza.

TEST 9

1. U kom slučaju je potrebno odabrati agilni razvoj pre nego planom vođen razvoj softvera? Navedi konkretan primer.

   – **Agilne metode** razvoja softvera se koriste:
   - Kada je potrebno da primenimo inkrementalnu isporuku softvera kupcu i kada od kupca očekujemo brz odgovor.
   - Kada se sistem razvija sa malim timom koji ima članove koji mogu međusobno neformalno da komuniciraju.
   - Kada raspolažemo dobrim alatima za održavanje verzija projektnog rešenja softvera.
   - Kada se u toku procesa razvoja softvera upotrebljava neformalno znanje, sa glavnim fokusom na znanje programera.

2. Šta je to programiranje u parovima?

   – Dva programera rade za istom jedinicom

3. Da li kulturološki i istorijski razlozi mogu da utiču na primenu agilnih metodologija?

   – Da

4. Kako testovi prate novu funkcionalnost koju donosi svaka nova verzija softvera?

   – Pokreće se izvršavanje i novih i starih testova.

5. Kod agilnih metoda procesi pripreme specifikacije, projektovanja i implementacije su jasno definisani i ne preklapaju se?

   – Ne

6. Kakav je odnos produktivnosti i kvaliteta rada u paru?

   – Manja produktivnost ali veći kvalitet koda.

7. Šta je to hibridni pristup u razvoju softvera?

   – Kombinovanje agilnog i planski vođenog pristupa.

8. Uključivanje svih aktera u razvoj sistema predstavlja problem kod velikih sistema?

   – Da

9. Za uspešnu primenu agilnih metoda neophodno je da članovi tima imaju vrlo dobre veštine?

   – Da

10. Koje su perspektive primene agilnih metoda kod razvoja velikih sistema?

    – Scaling up i Scaling out

11. Kada se definiše test softvera kod ekstremnog programiranja?

    – Pre pisanja koda