1. Navedi generičke aktivnosti u procesima programskog inženjerstva.

* specifikacija, razvoj i oblikovanje, validacija i verifikacija , evolucija

2. Navedi barem 2 značajke kakvoće programskog produkta.

* prihvatljivost, pouzdanost, održavanje

3. Navedi 3 problema evolucijskog modela razvoja i oblikovanja PP.

* proces razvoja nije jasno vidljiv, sustavi su loše strukturirani, često potrebne posebne vještine

4. Oblikovanje PP obuhvaća akciju inženjerstva zahtjeva. Navedi sve načine na koji se mogu izraziti zahtjevi.

* strukturiranim prirodnim jezikom, specifičnim jezikom za oblikovanje (npr. SDL) , grafičkom notacijom (npr. ERA, UML) i matematičkom specifikacijom (npr.vremenska logika)

5. Na proizvoljnom primjeru dijagrama „use case" objasnite razliku između *include* i *extended.*

* include → 'nužno sadrži' , bazni obavezno obavlja i njega (bazni --> dodatni)
extend → 'proširenje', moguće proširenje baznog, opcionalno (dodatni --> bazni)
npr.
odobri novu karticu - - include- - > zatraži novu karticu
ostvari bonus karticu - - extend - - > zatraži novu karticu (ostvari bonus)

6. Kako se u sekvencijskom dijagramu označava petlja?

* na strelicu: *[uvjet]povr:=naziv(parametri)
→ šalje se dok vrijedi uvjet, bez zvjezdice samo jednom

7. Nacrtaj UML dijagram razreda koji povezuje razrede: Knjiga, SadržajKnjige, PoglavljeUKnjizi, IndeksUKnjizi.

* zbunjuje me što misle pod sadržaj i indeks.. ako su to ona čuda koja dolaze an početak/kraj knjige, moj prijedlog bi bio:
Knjiga naravno glavno, iz nje idu kompozicije na ostala tri. (višestrukost na strani knjige 1, kao i na strani Indeksa i Sadržaja, a na Poglavlje 1..n)
Sadržaj i Indeks možda ovisnost prema Poglavlju ? (ako sam dobro shvatila njihovu svrhu, to bi moglo imati smisla)

8. Preslikaj 2 rečenice prirodnog jezika u ispravne formule logike predikata:
a) Ana voli svu Marijinu braću.
b) Ana voli jednog Marijinog brata.
Koristi predikate *voli(x,y), brat(x,y)*

** a) Vx [brat(x, Marija) → voli (Ana , x)]
b) Ex [brat(x, Marija) & voli (Ana,x)]

9. Ako su istinite formule:
(1) nonP
(2) Q
(3) (P ili nonQ ili R)
Općim postupkom dokazivanja teorema (opovrgavanje, obaranje) pokaži da li je R logička posljedica navedenih formula.

* R logička posljedica skupa {(1),(2),(3)} ako je svaki model tog skupa ujedno i model formule R.
* model formule R je bilo koji u kojem je R=T
* iz (1) P=F , iz (2) Q=T, iz (3) (P v nonQ v R) ako uvrstimo P i Q => F v F v R => da bi bilo točno mora R biti T
* dakle jedini model za skup {(1), (2), (3)} je : P=F, Q=T, R=T , što je i model za R
→ R je logička posljedica skupa {(1), (2), (3)}

10. Zaokruži neispravne formule CTL logike:
a) A[p U EF nonr]
b) AEFr
c) FGr
d) AF[(rUg) i (pUr)]
e) A[nonp U A[qUr]]
f) AnonGnonp

- neispravne: b) , c) , d) , f)
- ispred svake zagrade je Aili E. A unutar zagrade može biti U
- X- sljedeće stanje, G- duž puta, F- neko buduće stanje.
- A uz implikaciju, E uz 'and'

11. Napiši formulu CTL logike koja izražava: „Iz svakog stanja moguće je doći do početnog stanja."

** AG(EF početak)

12. U okviru procesa ispitivanja PP postoje faze 1-funkcijski test, 2-test performansi, 3-test instalacije, 4-test uporabe, 5-test komponenti, 6-test prihvatljivosti, 7-test integracije. Poredajte faze po vremenskom slijedu.

* 5-7-1 -2-6-3-4

13. Tijekom procesa strukturnog ispitivanja (white box) modula PP generiran je graf tijeka programa koji sadrži 8 čvorova i 12 lukova bez dodatno povezanih komponenti. Koliko je najmanje potrebno testova da se ispitaju svi temeljni putovi?

* CV(G) = Lukovi – Čvorovi + 2*P = 12 – 8 + 2*1 = 6

14. U arhitekturi protoka podataka pažnja je usredotočena na prijenos podataka između aktora. Upravljački tok je implicitan. Navedi barem 3 mehanizma upravljanja u protoku podataka.

* Guranje (PUSH) , Povlačenje (PULL) , Guranje/Povlačenje (PUSH/PULL) , Pasivni

15. U arhitekturi PP koja se zasniva na repozitoriju podataka postoje 2 velike podskupine: baze podataka i oglasna ploča. Navedi temeljnu razliku između ovih podskupina.

* baza podataka - vanjski procesi iniciraju promjenu sadržaja
oglasna ploča - promjena sadržaja inicira vanjske procese

upravljanje pogreškama – prevencija, detekcija, oporavak
aktivnosti testiranja –a) oblikovanje testiranja b) automatizacija c) testiranje d) valorizacija
tehnike testiranja (klasifikacija) – zasnovano na pokrivenosti/pogreškama/kvarovima
strukturno testiranje (shite box), funkcijsko testiranje (black box)

1
TEMELJNA PITANJA OKO PROGRAMSKOG INŢENJERSTVA
1. Što je programska potpora (software) ?

PP je računalni program i pridruţena dokumentacija.
2. Što je programsko inženjerstvo ?

Disciplina koja se bavi metodama i alatima za profesionalno oblikovanje i produkciju programske potpore uzimajući u obzir cijenu.
3. Koje je razlika između programskog inženjerstva i računarske znanosti ?

RZ se bavi teorijskim dijelom, PI se bavi praktičnom primjenom.
4. Koja je razlika između programskog inženjerstva i inženjerstva sustava ?

Inţenjerstvo sustava se bavi svim aspektima sustava dok PI se bavi razvojem programske infrastrukture, upravljanja te primjene.
5. Što je proces programskog inženjerstva ?

Skup aktivnosti čiji je razvoj ili evolucija programskog produkta. (specifikacija – analizom zahtjeva specificirati što sustav treba činiti i koja su ograničenja razvoja, razvoj i oblikovanje – izbor arhitekture i produkcija sustava, validacija i verifikacija – provjera da li sve radi kako treba, evolucija – rukovanje promjenama sukladno novih zahtjevima)
6. Što je model procesa programskog inženjerstva ?

Pojednostavljeno predstavljanje procesa iz određene perspektive.

7. Kakva je struktura cijene (troška) u programskom inženjerstvu ?

Cijena sadrţi cijenu razvoja, oblikovanja, ispitivanja i odrţanja.
8. Što su metode programskog inženjerstva ?

To su strukturni pristupi u razvoju i oblikovanju programske potpore što uključuje izbor modela sustava, pravila, preporuke i naputke.
9. Što je CASE (engl. *Computer-Aided Software Engineering*) ?

To su programski produkti namijenjeni automatiziranoj podršci aktivnostima u procesu programskog inţenjerstva.
10. Koje su značajke (atributi) dobre programske potpore ?

Prihvatljivost – razumljiv koristan i kompatibilan o ostalim sustavima, pouzdanost – korisnik mora vjerovati u pouzdanost sustava, odrţavanje – evolucija je sukladna izmijenjenim zahtjevima.
11. Koje su osnovne poteškoće i izazovi u programskom inženjerstvu ?

Heterogenost – različite tehnike i metode razvoja
Vrijeme isporuke, povjerenje.
12. Koje vrste projekata postoje u programskom inženjerstvu ?

Korektivni, adaptivni, re-inţenjerstvo, integrativni, unaprjeđujući.
13. Što je profesionalna i etička odgovornost ?

Povjerljivost, kompetencije, poštivanje prava intelektualnog vlasništva, ne zlorabiti računalne sustave.
Inţenjerstvo zahtjeva
- proces pronalaženja, analiziranja, dokumentiranja i provjere zahtijevanih usluga, te ograničenja u uporabi

- Zahtjevi sami za sebe su specifikacija

Zahtjevi obzirom na razinu detalja:
 Korisničke zahtjeve – moraju biti razumljivi ne-tehničkom osoblju, pišu se prirodnim jezikom te grafovima

 Zahtjeve sustava – pišu se strukturnim prirodnim jezikom, jezikom za oblikovanje sustava i matematičkom notacijom

 Specifikacija programske potpore – najdetaljniji opis i objedinjuje korisničke te sustavske zahtjeve

Zahtjevi obzirom na sadrţaj:
 Funkcionalni – izjave o uslugama koje sustav mora sadržavati, kako sustav reagira na poticaje, te kako se mora ponašati u određenim situacijama

 Nefunkcionalni – ograničenja u uslugama i funkcijama

 Zahtjevi domene primjene – proizlaze iz domene sustava kao i oni koji karakteriziraju tu domenu

Klasifikacija nefunkcionalnih zahtjeva:
 Zahtjevi programskog produkta – ponašanje na određen način

 Organizacijski zahtjevi

 Vanjski zahtjevi – zahtjevi izvan sustava i razvojnog procesa

Izraţavanje zahtjeva sustava:
 Strukturirani prirodan jezik

 Jezik za opis oblikovanja

 Grafička notacija

 Matematička specifikacija

Kompletnost i konzistencija zahtjeva
 Kompletni zahtjevi – sadrže opis svih zahtijevanih mogućnosti

 Konzistentni zahtjevi – nemaju konflikata u opisima
3**Izraţavanje zahtjeva**
 Strukturirani prirodni jezik – definiranje standardnih obrazaca i formulara kojima se zahtjevi izražavaju

 Jezik za opis oblikovanja (npr. SDL)

 Grafička notacija – grafički jezik proširen tekstom

 Matematička specifikacija – notacija na osnovu matematičkog koncepta

Zahtjevi sustava:
 Proceduralno sučelje – API

 Struktura podataka

 Predstavljanje podataka

Procesi inţenjerstva zahtjeva
 NEMA JEDINSTVENOG PROCESA!

 Generičke aktivnost:
- Studija izvedivosti
- Izlučivanje zahtjeva
- Validacija zahtjeva
- Upravljanje promjenama zahtjeva

Modeli procesa inţenjerstva zahtjeva:
 Klasični

 Spiralni

Spiralni proces:
- Trostupanjska aktivnost (specifikacija, validacija, izlučivanje)

- U svakoj iteraciji različit je intenzitet aktivnosti

- Svaka iteracija ima različitu razinu detalja

Generičke aktivnosti u procesu inţenjerstva zahtjeva:
 Studija izvedivosti

 Izlučivanje i analiza zahtjeva

 Validacija zahtjeva

 Upravljanje promjenama zahtjeva

Studija izvedivosti:
- U početku određuje da li se predloženi sustav isplati

- Kratka studija koja u pisanom dokumentu provjera pitanja:
- Da li sustav doprinosi ciljevima organizacije?
- Da li se sustav može izvesti postojećom tehnologijom?
- Da li se sustav može integrirati u već postojeći?
- Provedba se temelji na određivanju koje informacije su potrebne za studiju, prikupljanje informacija i pisanje izvješća
4 **Aktivnosti u spirali:**
 Izlučivanje zahtjeva

 Klasifikacija i otkrivanje zahtjeva

 Ustanovljavanje prioriteta i pregovaranje

 Dokumentiranje zahtjeva

Pogledi – način strukturiranja zahtjeva da oslikavaju perspektivu i fokus različitih sudionika
Tipovi pogleda:
 Pogledi interakcije – direktni kontakt npr. bankomat

 Indirektni pogledi – dionici ne koriste sustav direktno

 Pogledi domene primjene

Metode u izlučivanju i analizi zahtjeva:
 Intervjuiranje (zatvoreni, otvoreni) – sa pripremljenim skupom pitanja, bez pripremljenog skupa pitanja

 Scenariji
- opis početne situacije, opis normalnog tijeka događaja, opis što se može krivo dogoditi, informacije o paralelnim aktivnostima, opis gdje scenarij završava

- Primjeri iz stvarnog života o načinu korištenja sustava
- **Obrasci uporabe**
- temelji se na ideji scenarija
- Model obrasca je pogleda koji ističe ponašanje sustava kako ga vide vanjski korisnici
- Model razdjeljuje funkcionalnost sustava u transakcije razumljive korisnicima
- Tri temeljna elementa u modelima obrazaca:
  - Obrasci uporabe
  - Aktori
  - Odnosi
- Obrascima modelirati zahtjeve korisnika te scenarije ispitivanja sustava

Zaključci o UML obrascima:
- Modeliraju funkcionalne zahtjeve sustava
- Temeljeni su na ideji scenarija
- Služe za izlučivanje zahtjeva prema pogledu interakcije
- **UML dinamički dijagrami interakcija:**
- Sekvencijski
- Kolaboracijski

Validacija zahtjeva:
- Pokazati da zahtjevi odgovaraju sustavu koji naručitelj želi
- **Tehnike validacije:**
- Recenzija – sistematska ručna analiza
- Izrada prototipa
- Generiranje ispitnih slučaja

5**Što se u zahtjevima provjerava?**
- Valjanost
- Kompletnost
- Adaptabilnost
- Realnost
- Konzistencija
- Provjerljivost
- Razumljivost
- Slijednost

Etnografija:
- Znanost koja istražuje kako ljudi stvarno rade, a ne kako bi definicija poslovnog procesa to propisivala
- Koristi se za izlučivanje zahtjeva uzimajući u obzir aktivnosti drugih sustavom involviranih ljudi

Klasifikacija promjene zahtjeva:
- Okolinom promijenjeni zahtjevi – promjena zbog okoline u kojoj organizacija posluje
- Novonastali zahtjevi – zahtjevi kupca
- Poslječični zahtjevi – nastaju nakon uvođena u sustav, promjena procesa rada
- Zahtjevi kompatibilnosti – ovise o procesima drugih sustava

Proces programskog inženjerstva:
- Strukturirani skup aktivnosti potreban da se oblikuje i razvija programski produkt

Generičke aktivnosti:
- Specifikacija
- Razvoj i oblikovanje
- Validacija i verifikacija
- Evolucija

Model procesa programskog inženjerstva:
- Apstraktna reprezentacija procesa, predstavlja opis procesa iz određene perspektive

Najznačajniji procesi i modeli:
1. Vodopadni – odvojene i specifične faze specifikacije i razvoja

2. Evolucijski – isprepleteni specifikacija, evolucija i razvoj

3. Komponentno usmjeren – sustav se gradi od postojećih komponenata

4. RUP proces – temelji se na oblikovanju preko modela

6**Primjer vodopadnog modela:**
Procesne faze vodopadnog modela:
- Analiza zahtjeva i definicije
- Razvoj i oblikovanje sustava i programske potpore
- Implementacija i ispitivanje modula
- Integracija i testiranje sustava
- Uporaba sustava i održavanje

TEMELJ VODOPADNOG SUSTAVA: SVAKA FAZA SE MORA ZAVRŠITI PRIJE POKRETANJA NOVE FAZE!!!!!!!!!!!!!!!
Problemi vodopadnog modela:
- Dijeljenje projekta na particije je teško zbog promjena koje kupac želi
- Model je prikladan ako su zahtjevi razumljivi i promjene svedene na minimum
- Vodopadni model se koristi kod velikih inženjerskih projekata gdje se sustav razvija na nekoliko odvojenih mjesta

Evolucijski model:
- Koristi dva postupka:
  - Istraživački razvoj i oblikovanje:
    - Kontinuiran rad s kupcem
    - Započinje se dobro definiranim početnim zahtjevima te se nove specifikacije dodaju prijedlozima kupca
  - Metoda odbacivanja prototipa:
    - Započinje se grubo definiranim zahtjevima koji se tijekom postupka razjašnjavaju

7**Primjer evo modela:**
Problemi evo modela:
- Proces razvoja nije jasno vidljiv
- Sustavi su loše strukturirani

Primjena evo modela:
- Male i srednje interaktivne sustave
- Dijelove velikih sustava
- Sustavi s kratkim vijekom trajanja

CBSE – model zasnovan na komponentama:
- **Sustav se integrira višestrukom upotrebom postojećih komponenata ili uporabom gotovih**
- Specifikacija i analiza zahtjeva
- Analiza komponenata
- Modifikacija zahtjeva
- Oblikovanje sustava s višestrukom uporabom
- Razvoj i integracija

RUP – rational unified process
- Moderan model procesa programskog inženjerstva izveden na temelju jezika za modeliranje UML-a u pridruženih aktivnosti

- Dijagrami su modelirani UML standardom

- Modeli su dokumentirani s jednim ili više dijagrama

- **Opisuje se kroz tri perspektive:**
- **Dinamička** – pokazuje slijed faza procesa kroz vrijeme
- **Statička** – pokazuje aktivnosti unutar pojedinih faza procesa
- **Praktična** – sugerira aktivnosti kroz iskustvo i dobru praksu
- **Faze RUP procesa:**
- Početak
- Elaboracija – plan projekta, specifikacija značajki i arhitekture sustava
- Konstrukcija
- Prijenos produkta korisnicima

Iteracije u modelima procesa programskog inženjerstva:
- Postoje dva međuovisna pristupa:
- Inkrementalni pristup
- Spiralni razvoj i oblikovanje
- **Inkrementalni pristup:**
- Sustav se ne isporučuje u cjelini korisniku nego se dijeli u inkrementalne dijelove koji predstavljaju djelomične funkcionalnosti
- Zahtjevi se dijele prema prioritetima te tako oni najviši prioriteti se isporučuju u ranim dijelovima sustava
- Prednosti inkrementalnog pristupa:
- Kupac sa svakim dijelom dobiva svoju vrijednost
- Smanjen rizika za neuspjeh
- **Ekstremno programiranje:**
  - Pristup se bazira na oblikovanju i isporuci malih funkcionalnih inkremenata
  - Postupak uključuje kontinuirano poboljšanje koda
  - U razvoju se koristi programiranje u paru

Spiralni razvoj i oblikovanje:
- Proces se predstavlja spiralom
- Svaka petlja u spirali predstavlja jednu fazu procesa
- Nema ranih završenih faza
- **Sektori u spiralnom modelu:**
- Postavljanje ciljeva
- Procjena i smanjivanje rizika
- Razvoj i validacija
- Planiranje

Generičke aktivnosti u procesu programskog inženjerstva:
- **Faze:**
- Specifikacija i izlučivanje zahtjeva
- Oblikovanje i implementacija programskog produkta
- Validacija i verifikacija
- Evolucija
- **Specifikacija i izlučivanje zahtjeva:**
- Aktivnosti:
  - Studij izvedivosti
  - Izlučivanje zahtjeva
  - Validacija zahtjeva
  - Upravljanje promjenama zahtjeva
- **Oblikovanje i implementacija programskog produkta:**
- Oblikovanje programskog produkta:
- Implementacija
- **Ispitivanje programske potpore:**
- Testni podatci (I) – postavljeni ulazi za testiranje
- Očekivani izlaz (O)
- Testni slučaj – uređen par (I, O)
- Stvarni izlaz – rezultat dobiven provođenjem testa
- Kriterij prolaza testa – kriterij usporedbe očekivanog i stvarnog izlaza određen prije provođenja testa
- **Formalna verifikacija:**
- Postupak provjere da formalni model dijela izvedenog sustava (I) odgovara formalnoj specifikaciji (S) sa matematičkom izvjesnošću
- **Case – computer aided software engineering**
- Case podupire automatizaciju oblikovanja raznim alatima
- Funkcionalna perspektiva
  - Alati se klasificiraju prema specifičnoj funkciji
- Procesna perspektiva
  - Alati se klasificiraju prema aktivnostima koje podupiru u procesu
- Integracijska perspektiva
  - Alati se klasificiraju prema njihovoj organizaciji u integralne cjeline
  - Alati – podupiru individualne zadatke u procesu
  - Radne klupe (workbench) – podupiru pojedine faze procesa
  - Razvojne okoline (enviroments) – podupiru cijeli ili značajan dio procesa programskog inženjerstva

Pretvorite slijedeće rečenice u predikatnu logiku:

1. Svaki auto ima 4 kotača:
$$\forall X ((auto\ X) \Rightarrow (ima\_4\_kotača\ X))$$

2. Postoji auto plave boje
$$\exists X ((auto\ X) \wedge (plavi\ X))$$

3. Ferrari je brzi auto
$$(auto\ Ferrari) \Rightarrow (brzi\ Ferrari)$$

4. Plavi ferrari ima 4 kotača
$$(plavi\ Ferrari) \Rightarrow (ima\_4\_kotača\ X)$$

5. Interpretirati formulu (rečenicom):
$$\exists X\ (\neg(prosti\ X) \wedge (parni\ X))$$

Postoji X takav da nije prost, ali je paran.
Postoji paran broj koji nije prost.
Pretvorite slijedeće rečenice u vremensku logiku:
6. Jednog će dana sigurno biti sunčano
$$\mathbf{AF(sunčano)}$$

7. Moguće je da ako se grijanje isključi temperatura padne ispod 0.
$$\mathbf{EF(isključeno\_grijanje \Rightarrow AF(t<0))}$$

8. Interpretirati formulu:
$$E(brza\_vožnja\ U\ (kazna \vee sudar))$$
Postoji mogućnost da brza vožnja uzrokuje sudar ili kaznu.

9. Ako je žarulja upaljena za sekundu će se ugasiti, i obrnuto:
$$AG((upaljena \Rightarrow AX(\neg upaljena)) \vee (\neg upaljena \Rightarrow AX(upaljena)))$$

10. Svaki dan je ili sunčano ili oblačno:
$$AG(sunčano \vee oblačno)$$

11. Do Rijeke se vozimo autoputom:
$$A(autoput\ U\ u\_Rijeci)$$
12. Prvih 10 iteracija uvijek je t<0:
$$\mathbf{A((t<0)\ U\ (iter==11))}$$

13. Prvih 10 iteracija t može konstantno biti manje od 0:
$$\mathbf{E((t<0)\ U\ (iter==11))}$$

14. Kad se pojavi signal poziva se funkcija obradi:
AG(signal ⇒ poziv_fje_obradi)
AF(signal ⇒ poziv_fje_obradi) ◊ nije dobro, zašto?
EF(signal ⇒ poziv_fje_obradi) ◊ nije dobro, zašto?
AG(signal ⇒ AX(poziv_fje_obradi))

15. Što je to „testiranje"?
otkrivanje grešaka
16. Što se koristi za testiranje? (kakvi postupci)
eksperimentiranje; logika, matematika; modeli; alati
17. Što je to EULA i kakve veze ima s testiranjem?
End Users Licence Agreement,(sporazum proizvođača ii korisnika o korištenju aplikacije)
dokument koji objašnjava kako se software ne smije koristiti zbog ograničenja koje proizvođač postavlja te ukoliko se to dogodi proizvođač se ograđuje od bilo kakvih incidenata (npr. Igrate call of duty i odlučite otić van upucat nekoga u čelo, e nakon toga završite u zatvoru no proizvođač ne jer postoji EULA), veza s testiranjem je takva da sve šta treba radit aplikacija je testirano i garantira se da će ta funkcionalnost raditi
18. U kojem su odnosu kvar(error), pogreška(fault) i zatajenje(failure)?
npr. kvar=greška u programu, pogreška=kriva vrijednost varijable
zatajenje=svjetla ugašena (sustav ne radi dobro)
19. Koji se kriterij koristi za određivanje „prolaznosti" nekog testa?
izlaz za odabrani ulaz treba odgovarati očekivanom izlazu
20. Upravljanje greškama može se ostvariti sa prevencijom, detekcijom i postupkom oporavka od pogreške. Testiranje može biti dio koje navedene grane postupaka?
svih; detekcije
21. S obzirom na izvor informacija, tehnike testiranja možemo podijeliti u dvije kategorije. Koje?
strukturno (white box) i funkcijski (black box)
22. Koji je najveći nedostatak postupka testiranja?
ulaz je beskonačan
23. Testiranje kod kojeg su ulazi podijeljeni u grupe pokazuje bolje rezultate. Kada je rečeno istina i zašto?
Ako je podjela dobro napravljena, ponašanje programa za elemente pojedinih grupa je slično (prolazi iste if-ove, petlje i sl.) pa …
24. Koji od modela testiranja V ili W uključuje više testiranja? Zašto?
W, testiranje se vrši nakon završetka svake faze/dijela projekta

25. Tko obavlja koja od slijedećih testiranja: Testiranje komponenti, Integracijsko testiranje, Testiranje sustava, Test prihvatljivosti, Test instalacije, Alpha test, Beta test
test komp => programer; integr, test. sustava => testni tim;
prihv., inst => naručitelj; alpha => interni potencij. "korisnici", beta=>vanjski
26. Ako nam na raspolaganju stoji dijagram stanja, kako treba oblikovati testove?
tako da se prođu svi prijelazi i stanja
27. Pri funkcijskom testiranju kako odabrati ulaze? (načelno)
izvan dozvoljenih vrijednosti, na rubovima (prije granice i poslije granice), te naravno unutar granica

28. Pri strukturnom testiranju kako odabrati ulaze? (načelno)
da se obuhvati sav kod, svi mogući prolazi,
29. U čemu se regresijsko testiranje razlikuje od osnovnog testiranja?
obavlja se nakon ispravljanja greške, nije potrebno SVE ponavljati
30. Načini automatizacije testiranja:
autom. ručnog testiranja
"record & playback" (snimanje i reprodukcijom korisnika)
čitanjem podataka iz datoteka
funkcionalna dekomp. (moguće testiranje dijelova i funkcija)
testna okružja (alati)
31. Navesti neka svojstva arhitekture protoka podataka. -> Razdvajanje procesnih dijelova programa i minimizacija dijelova programa koji se odnose na eksplicitno povezivanje varijabli, funkcija, modula
32. Navesti par primjera u kojima se koristi skupno sekvencijska arhitektura (batch seq.). On Diskretne transakcije unaprijed određenog tipa i koje se pojavljuju u periodičnim intervalima, periodični ispisi i dopune, obrada koja nije pod strogim vremenskim ograničenjima, skupna obrada (npr: obračun plaća). Transformacijska analiza podataka: Sakupljanje i analiza sirovih podataka u koračnom i skupnom modu (npr. crna kutija u aviu).
33. U arhitekturi cjevovoda i filtara, što može biti filter?

34. Kako se ostvaruje „guranje" podataka, a kako „povlačenje"?
Guranje-izvor gura podatke od sebe pozivom receive(data) pasivnog čitatelja, povlačenje-aktivni čitatelj inicira prijenos pozivom data_supply() pasivnom pisatelju
35. Naredba: „cat bodovi.txt | grep 100 | sort > prazna.txt" u UNIX sustavima pokrenuti će koliko procesa i koliko cjevovoda?
Cat -> bodovi -> grep -> sort -> prazna => jedan cjevovod, 5 procesa
36. Sa stanovišta rasuđivanja (razumljivosti) arhitektura protoka podataka je:

37. Dodavanje nove komponente u programski sustav zasnovan na događajima je lagano ili teško? Zašto?
Omogućuje razdvajanje i autonomiju komponenata., Snažno podupire evoluciju i ponovno korištenje. Jednostavno se uključuju nove komponente bez utjecaja na postojeće
38. Gdje se najčešće koristi arhitektura „model view controller"?
Proračunske tablice, web aplikacije
39. Dodavanje nove komponente u programski sustav zasnovan na repozitoriju podataka je lagano ili teško? Zašto? Prednost: jednostavno integriranje autonomnih sustava, oglasna ploča je inherentno fleksibilna (šta god se stavi na nju iduci podatci se mogu lako nadovezati)
40. Arhitektura repozitorija podataka koji koristi ER model baze podataka sliči arhitekturi opisanoj dijagramom klasa. Međutim ima razlika. Koje su najbitnije? Dozvoljava višestruke (nstruke) relacije između dva razreda, atributi s više vrijednosti su dozvoljeni, er specificiraju identifikatore,
41. Razlika između baze podataka i oglasne ploče:
Baza podataka: vanjski procesi iniciraju promjenu sadržaja. Oglasna ploča: promjena sadržaja inicira vanjske procese (KS).
42. Mehanizmi upravljanja tokom podataka: guranje, povlačenje, guranje/povlačenje, pasivni mehanizam
43. Generičke aktivnosti u programskom inženjerstvu:
razvoj i oblikovanje, validacija i verifikacija, specifikacija, evolucija
44. Značajke programskog produkta:
prihvatljivost, pouzdanost, održavanje
45. Evolucijski model problemi: proces razvoja nije jasno vidljiv, sustavi su loše strukturirani, potrebne su posebne vještine
46. Evolucijski model uporaba: mali i srednji interaktivni sustavi, sustavi s kratkim vijekom trajanja
47. Vodopadni model problemi: dioba na particije čini implementaciju promjena težim, prikladan je samo ako su zahtjevi dobro specificirani
48. Vodopadni model primjena: veliki projekti gdje se sustav razvija na nekoliko mjesta
49. Da li je formula predikatne logike (Q ∨ ¬Q) logička posljedica propozicijsko simboličke varijable P i zašto?
to vrijedi jer za svaku interpretaciju P koji je istinit P=T istinita je i desna strana
- (P ∧ Q) I= P
- lijeva strana = T samo za (P=T, Q=T), a to daje i desnoj strani =T, dakle gornji izraz vrijedi (P je logička posljedica (P ∧ Q).
- 2. (P ∨ Q) I= P
- lijeva strana je istinita za (P=F,Q=T; P=T,Q=F; P=T,Q=T), ali desna za interpretaciju (P=F,Q=T) nije istinita, te P nije logička posljedica (P ∨ Q).
- 3. {¬Q, (P ∨ Q)} I= P
- skup Γ na lijevoj strani je istinit samo za Q=F, P=T, a to daje istinitost i desnoj strani, te je P logička posljedica navedenog skupa Γ.

- Formalan sustav {Γ, L} je *ispravan* (*engl. sound*)
- ako Γ =| ωi kadgod je Γ –| L ωi ,
- t.j. svaka pravilima dokazana formula je ujedno i logička posljedica skupa Γ.

- Γ –| L ωi implicira Γ =| ωi

- Formalan sustav {Γ, L} je *kompletan* (engl. complete)
- ako Γ –| L ωi kadgod je Γ =| ω ,
- t.j. svaku logičku posljedicu skupa Γ moguće je dokazati pravilima L.

- Γ =| ω implicira Γ –| L ωi

50. SAT(zadovoljivost) Problem?
- Tražimo model skupa formula Γ (interpretaciju koja evaluira sve formule u skupu Γ u istinito). To je ekvivalentno traženju modela jedne složene formule koja se sastoji iz konjunkcije svih formula u Γ.
- Γ skup formula je najčešće dan u CNF obliku:
- (k11 ∨ … ∨ k1p) ∧ (k21 ∨ … ∨ k2r) ∧ … ∧ (kp1 ∨ … ∨ kps)

51.
- A- vrijedi za sve putove
- E- postoji put za kojeg vrijedi
- Dakle A ćeš upotrijebiti ako ti piše da ćeš sigurno doći u neko stanje jer ti možeš krenuti bilo kojim putem pa moraš koristiti da vrijedi za sve puteve(počevši od stanja u kojem si naravno---dakle početno stanje je u kojem si trenutno)...
- E ćeš upotrijebiti ako ti piše da je moguće da dođeš u neko stanje ili postoji stanje itd....
- Iza jednog od ta dva moraš upotrijebiti neki od X,G,F,U...
- G-vrijedi za sva stanja u putu..
- F-postoji barem jedno stanje u odabranom putu..

51. Proces testiranja redoslijed faza:
Test integracije -> Funkcijski test -> Test Performansi -> Test Prihvatljivosti -> Test Instalacije -> Uporaba Sustava
52. Izražavanje zahtjeva sustava:
Prirodan jezik, jezik za opis oblikovanja, grafička notacija, matematička specifikacija
53. Tijekom testiranja generiran je graf tijeka programa sa 12 lukova i 8 čvorova. Koliko minimalno testova treba napraviti da se ispitaju svi temeljni putovi?
12-8+2 = 6
- **Teorija grafova – izračun broja linearno neovisnih puteva**
  - **CV(G)** - Ciklomatička kompleksnost (engl. **Cyclomatic complexity)**
  - Broj neovisnih puteva u temeljnom skupu
  - *CV(G)* = Lukovi – Čvorovi + 2*P

Lukovi = broj lukova u grafu
Čvorovi = Broj čvorova u grafu
P = Broj povezanih komponenti (podprograma, prekidnih rutina i sl.)