1. **Šta predstavlja verifikacija, a šta validacija?**

* *Validacija* predstavlja odgovor na pitanje: „Da li gradimo pravi proizvod?“,zato što treba biti siguran da softver radi ono što korisnik stvarno traži.
* *Verifikacija* predstavlja odgovor na pitanje: „Da li na pravi način gradimo proizvod?“, zato što softver treba da bude usaglašen sa svojom specifikacijom.

1. **Objasniti statičku verifikaciju softvera.**

* *Statička verifikacija* softvera predstavlja analizu statičkih reprezentacija Sistema da bi se otkrili problemi (naziva se I *inspekcija softvera*). Za statičku verifikaciju se mogu koristiti alati za analizu koda I alati za pregledavanje dokumentacija koja je predmet inspekcije.   
  Njen cilj je ispitivanje izvorne reprezentacije sotvera radi otkrivanja anomalija i defekata. Ne zahteva izvršenje softvera tako da se može izvoditi pre njegove implementacije.

1. **Navesti, kratko opisati i uporediti različite metode testiranja softvera.**

* Testiranje softvera se odnosi na izvršenje i posmatranje ponašanja proizvoda (dinamička verifikacija). Sistm se podvrgava test podacima i posmatra se kako se ponaša u radu. Postoje razne metode za testiranje softvera, a neke od njih su:
  + - **Defektno testiranje** – podrazumeva da se testovi projektuju da otkriju defekte u sistemu. Uspešan defect test je onaj koji otkriva defekte u sistemu.
    - **Validaciono testiranje –** njegov cilj je pokazati da su ispunjeni zahtevi. Uspešan test je onaj koji pokazuje da su zahtevi implementirani na pravi način.

1. **Opisati i uporediti *top-down i bottom-up* pristupe projektovanja softvera.**
2. **Navesti i kratko opisati tipove održavanja softvera.**

* ***Održavanje softvera*** podrazumeva izmenu programa nakonpuštanja u upotrebu. Ovaj termin se uglavnom koristi da opiše izmene softvera razvijenog po narudžbini. Za generičke softverske proizvode se kaže da evoluiraju kroz nove verzije. Tipovi održavanja softvera su:
  + - ***Održavanje u cilju ispravke softverskih grešaka –*** izmena sistema kako bi se otklonili nedostaci koji sprečavaju da sistem radi u skladu sa svojom specifikacijom.
    - ***Održavanje sa ciljem prilagođavanja softvera za drugačije radno okruženje –*** promena sistema tako da može da radi u drugačijem okruženju (računar, OS, itd.) od onog za koje je inicijalno implementiran.
    - ***Održavanje u cilju dodavanja nove ili izmene postojeće funkcionalnosti sistema –*** modifikacija sistema kako bi zadovoljio nove zahteva.

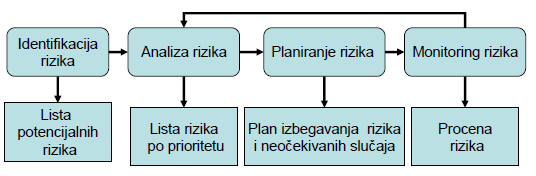
1. **Ukratko opisati i navesti prednosti i nedostatke slojevitog arhitekturnog modela.**

* **Slojevit arhitekturni model** se koristi kod modeliranja interfejsa između podsistema. Sistem se organizuje u skup slojeva od kojih svaki obezbeđuje jedan skup funkcionalnosti sloju iznad i služi kao klijent sloju ispod. Omogućava inkrementalni razvoj podkomponenti u različitim slojevima.

**\* Prednosti –** *promena interfejsa jednog sloja može da utiče na maksimalno još dva sloja. Laka zamena jednog sloja drugim ukoliko su im interfejsi identični. Baziran je na visokom nivou apstrakcije.*

**\* Nedostaci -** *ne mogu svi sistemi da se lako organizuju po ovom modelu.*

1. **Kratko opisati i grafički ilustrovati proces upravljanja rizikom.**

* **Upravljanje rizikom** obuhvata identifikovanje rizika i skiciranje plana minimizacije uticaja rizika na projekat, a sam ***rizik*** je verovatnoća da se neka nepovoljna prilika pojavi. Rizici se svrstavaju u tri kategorije:
  + - ***Rizici projekta –*** utiču na raspored ili resurse.
    - ***Rizici proizvoda –*** utiču na kvalitet ili performanse softvera koji se razvija
    - ***Rizici poslovanja –*** utiču na organizaciju koja razvija ili nabavlja softver.

1. **Navesti osnovne metode projektovanja softvera i objasniti svaku od njih.**
2. **Navesti osnovne arhitekturne modele projektovanja softvera. Objasniti Control arhitekturni model.**

* Osnovni arhitekturni modeli projektovanja softvera su:
  + ***Repository (Skladište)***
  + ***Pipe and filter (Protočna obrada)***
  + ***OO model***
  + ***Client/Server model***
  + ***Slojeviti model (Layered)***
  + ***Event-driven model (implicitno pozivanje)***
  + ***Control model***
* ***Control arhitekturni model –*** koristi se kod sistema gde je potrebna centralizovana kontrola. Kontrolni porsistem upravlja tokom informacija između ostalih podstistema. Postoje 4 osnovne grupe ovih modela: *call-return modeli, manager modeli, feedback modeli, open-loop modeli.*
  + ***Komponente*** ovog sistema čine kontrolni algoritam i podsistemi.
  + ***Konektore*** ovog sistema obuhvataju relacije između tokova podataka.
* ***Call-return modeli –*** kod ove grupe modela, kontrola kreće od vršnih podisstema i proteže se najniže (top-down) pristup, a pogodni su za sekvencijalne sisteme.
* ***Manager modeli –*** ova grupa modela se primenjuje kod konkurentnih sistema. Jedna sistemska komponenta određuje njen početak, zaustavljanje i koordinaciju rada svih procesa i sistemu.

1. **Šta predstavlja upravljanje projektima i po čemu su softverski projekti osobeni?**
2. **Ukratko opisati i navesti prednosti i nedostatke klijent-server arhitekturnog modela.**
3. **Navesti i kratko opisati osobine dobro projektovanog softvera.**
4. **Šta predstavlja i šta se dobija reinženjeringom sistema?**
5. **Kada se, po RUP metodologiji vrši detaljno projektovanje i šta se pod tim podrazumeva? Koje su prednosti a koji nedostaci „event-driven“ arhitekturnog modela?**
6. **Objasniti alfa i beta testiranja.**
7. **Napisati preporuke kada treba koristiti „feedback“, a kada „event-driven“ arhitekturni model?**
8. **Objasniti razlike između inspekcije i testiranja, kao i između testiranja i debagiranja?**
9. **Šta je to „test-slučaj“ i objasniti strukturu „test-slučaja“?**
10. **Šta su kontrolne tačke, a šta izlazni produkti? Ko određuje kontrolne tačke, a ko izlazne produkte projekta i na osnovu kojih parametara?**
11. **Koje aktivnosti obuhvata upravljanje rizicima? Objasnite tipove rizika.**