Modeli softverskih procesa

Principi softverskog inženjerstva, *Elektrotehnički fakultet Univerziteta u Beogradu*

Modeli softverskih procesa (modeli životnog ciklusa softvera)

- Životni ciklus softvera obuhvata period od definicije zahteva do prestanka upotrebe.
- Model životnog ciklusa opisuje procese iz razvoja, korišćenja i održavanja softverskog proizvoda u toku njegovog životnog ciklusa.
- Za konkretan proizvod potrebno je izabrati konkretan model (implementirati standard).
- Ne postoji jedinstveni optimalan model za sve softverske proizvode, obično se primenjuje neki od standardnih modela ili neka kombinacija istih.

Važnost modela procesa

- Modeli procesa su važni za:
  - Organizaciju projekta – inače se nekoordinisano upravlja projektom; takođe moguće je prenošenje iskustva sa drugih projekata.
  - Planiranje vremena i troškova projekta
  - Analizu projekta: koje su slabe tačke u razvojnom procesu?
  - Procenu kvaliteta softverskih preduzeća: sertifikacija za ISO 9000 (skup pet standarda za upravljanje kvalitetom, ali ima i drugih standarda)

Model procesa

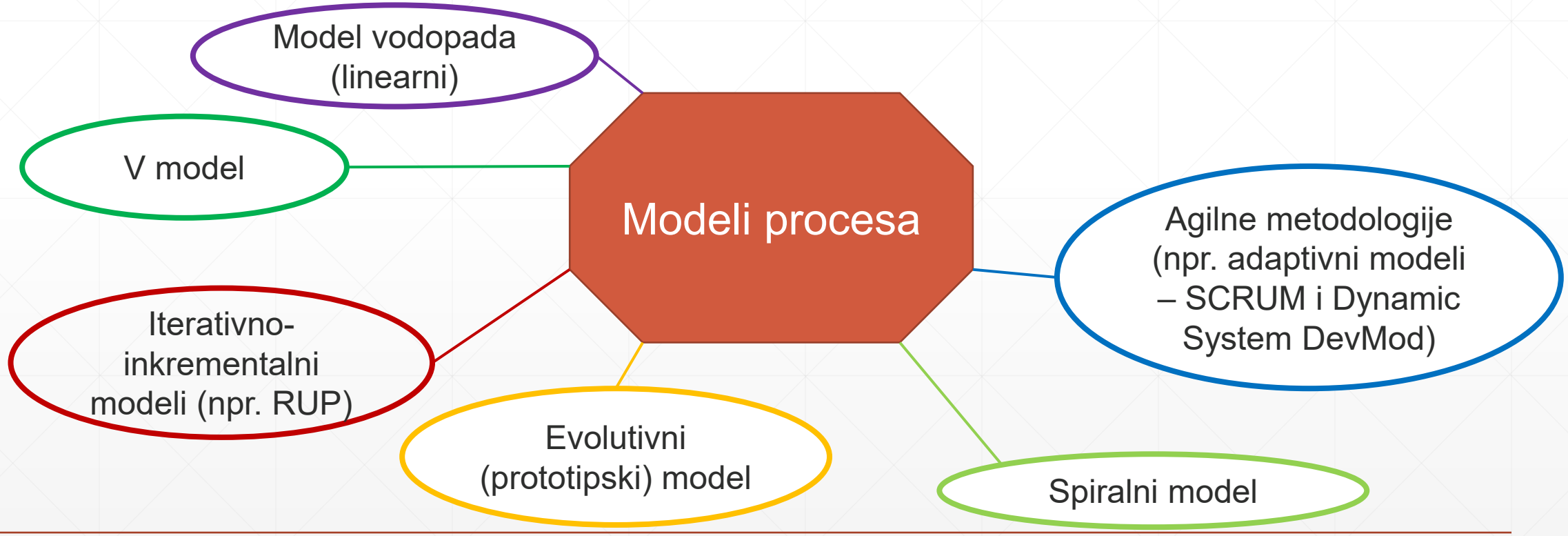
- Uopšteno:
  razvojni plan koji definiše opšti proces razvoja softverskog proizvoda.
- Preciznije:
  Definicija koja određuje koje **aktivnosti** se izvršavaju, od strane kojih osoba – u kojim **ulogama**,
  kojim **redosledom** će aktivnosti biti izvršavane, koji **proizvodi** će biti razvijani i
  kako će se procenjivati njihov **kvalitet**.

Uloga

Saradnik koji izvršava određenu aktivnost
npr. rukovodilac (menadžer) projekta,
menadžer produkta,
projektant (arhitekta) sistema,
projektant baze podataka,
analitičar zahteva, programer,
testni inženjer (softverski tester), itd.

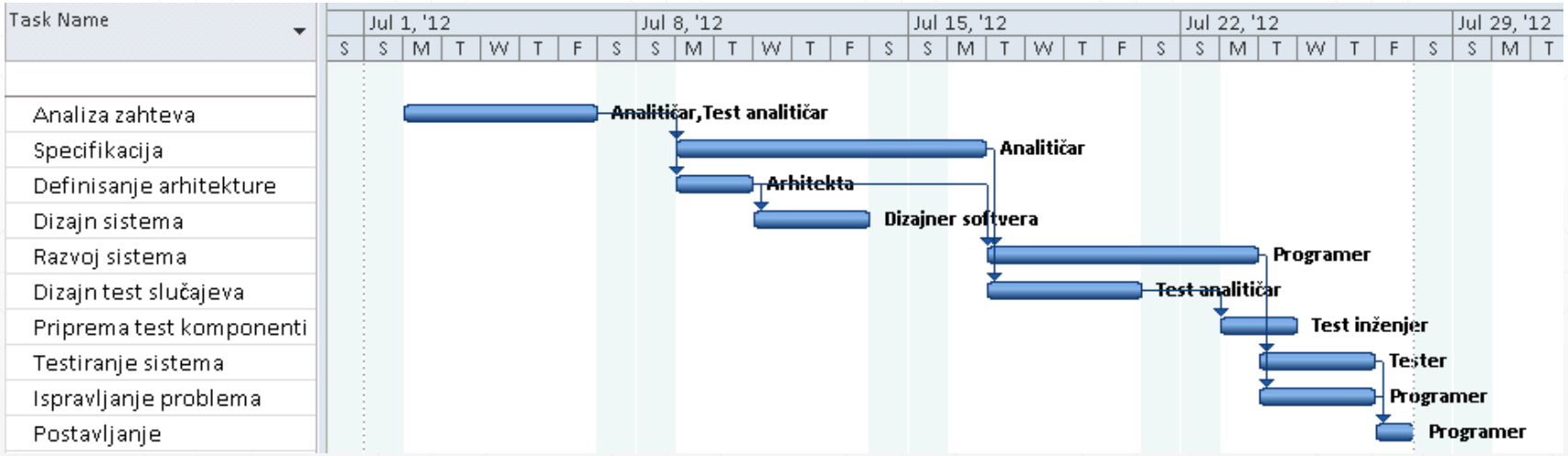
Pregled postojećih modela

- Modeli koje ćemo učiti u okviru našeg predmeta



Šta je Gantov dijagram?

- Gantov dijagram (Gantogram) prikazuje aktivnosti koje je potrebno izvršiti, ko će ih izvršiti, kada će početi, kada će se završiti, kao i koje su međusobne zavisnosti među aktivnostima.
- Gantogram prikazuje: aktivnosti, međusobne zavisnosti i članove tima koji rade aktivnosti.
- Način na koji se organizuju i grupišu ove aktivnosti predstavlja model razvoja softvera.

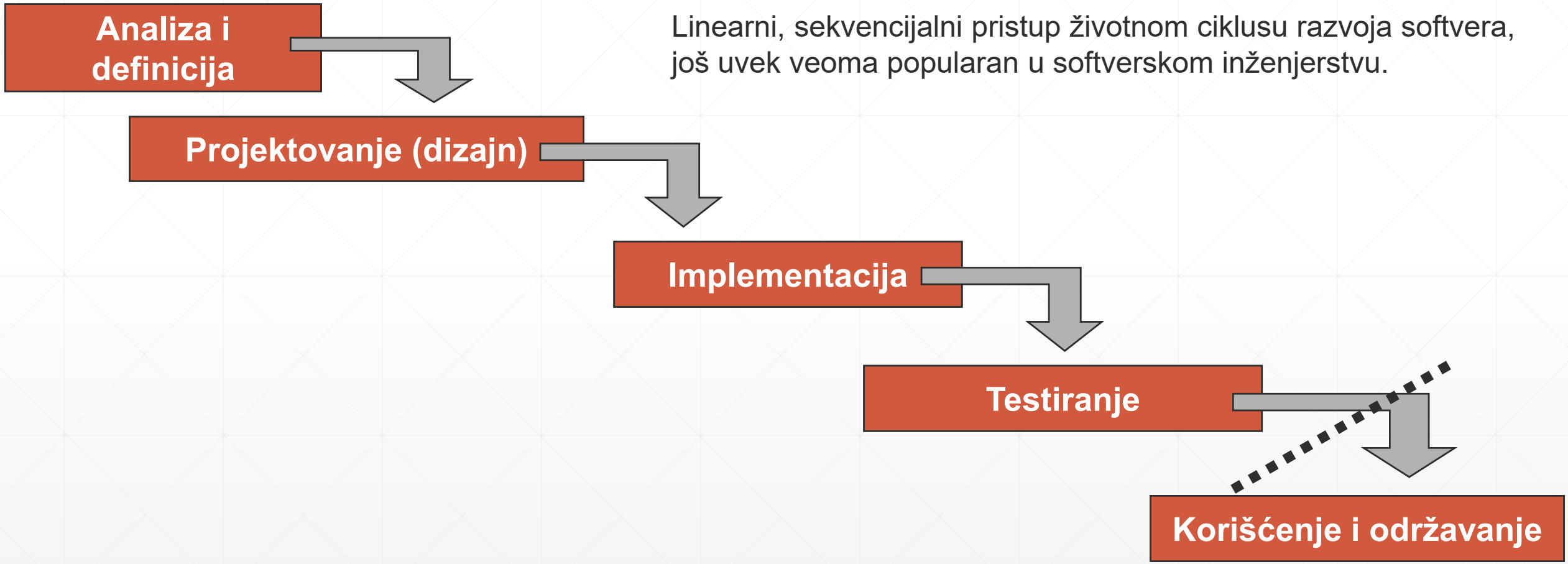


Model vodopada - linearni

(eng. *Waterfall model*)

Klasični model vodopada (iz 1970.)

Linearni, sekvencijalni pristup životnom ciklusu razvoja softvera, još uvek veoma popularan u softverskom inženjerstvu.

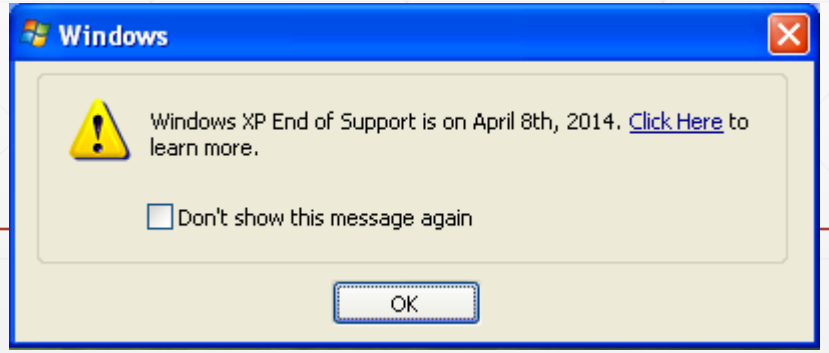


Model vodopada – faze (1)

- **Analiza zahteva**
  - Faza u kojoj se sakupljaju zahtevi krajnjih korisnika.
  - Analizira se šta je potrebno raditi.
  - Kreiraju se ugovori kojima se definiše šta će biti urađeno i potvrđuju zahtevi koji će biti implementirani.
- **Dizajn softvera**
  - Faza gde se detaljnije analiziraju zahtevi prikupljeni tokom analize
  - Specificira se kako će se implementirati softver
  - Kreira se tehnička specifikacija i dizajn softvera
- Dizajn softvera predstavlja konkretan plan kako će biti implementiran sistem od generalne arhitekture softvera do detaljnog opisa implementacije pojedinih komponenti i algoritama.
- Na kraju ove faze je poznato kako će se sistem implementirati.

Model vodopada – faze (2)

- **Implementacija**
  - Faza u kojoj se projektovani softver implementira u određenom programskom jeziku, radnom okviru (*frameworks*), sa kojim bibliotekama, u kom alatu za razvoj (npr. neki IDE), i za koju platformu.
  - Na kraju ove faze softver je završen i spreman za upotrebu (ali ne uvek odmah od strane korisika).



- **Testiranje**
  - Faza u kojoj se planira testiranje, izvršava testiranje implementiranog sistema (iz prethodnog koraka), prijavljuju i otklanjaju problemi nađeni u softveru.
  - Na kraju ove faze softver je testiran i spreman na predaju krajnjim korisnicima (tzv. produkciona verz.).
- **Korišćenje i održavanje**
  - Faza tokom koje se održava softver predat krajnjim korisnicima, uz tehničku podršku (tzv. *Support*), i eventualne dorade u skladu sa izmenama traženim od korisnika (npr. *Upgrade*, *Update*, *Patch/Fix*, itd.)
  - Ova faza traje sve dok korisnici upotrebljavaju softver (a nekad i sami definišu kraj podrške).

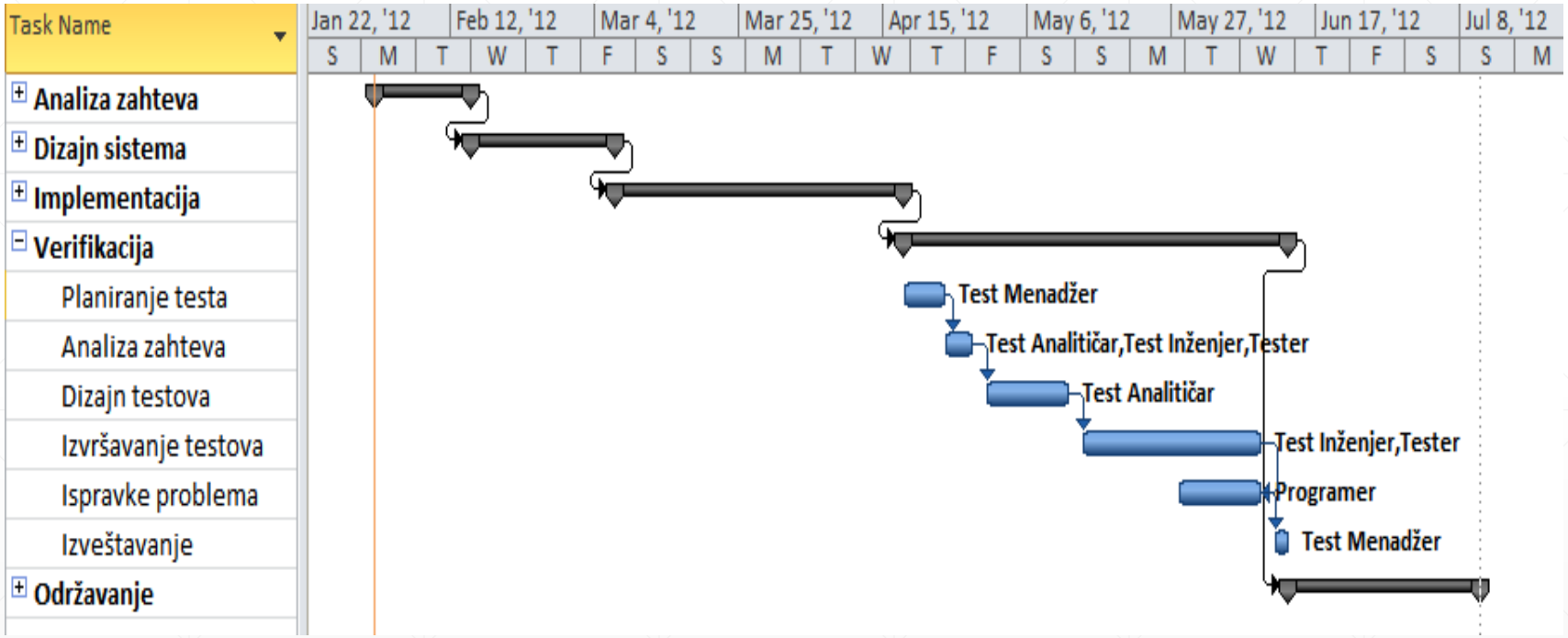
Model vodopada - prednosti i nedostaci

+	-
Veoma dobro definisani dokumenti koje treba napraviti, šabloni za pisanje dokumenata, kao i kriterijumi kojima se procenjuje da li je dokument dobar ili ne.	Teške reakcije na nepredviđene promene i rizik da se dokumenti i kod više puta ponovo rade u slučaju grešaka.
Precizno definisani procesi i kanali komunikacije kojima se prenose informacije	Mogućnost da precizne procene I planirani procesi postanu nevalidni posle nekog vremena.
Stroga podela na uloge gde je tačno definisano ko šta radi.	Teško primenljiv u situacijama gde se zahtevi često menjaju.
Precizno definisanje kada i koliko članova tima određenih uloga je potrebno u svakoj fazi projekta.	Postoji mogućnost da određeni procenat funkcionalnosti više ne bude koristan korisnicima u trenutku kada se isporuči softver.
Mogućnost da se predvidi trajanje, cena I broj članova tima na projektu i da se to lakše predstavi klijentima.	Teško nalaženje neophodnih članova tima u trenucima nepredviđenih prepravki.

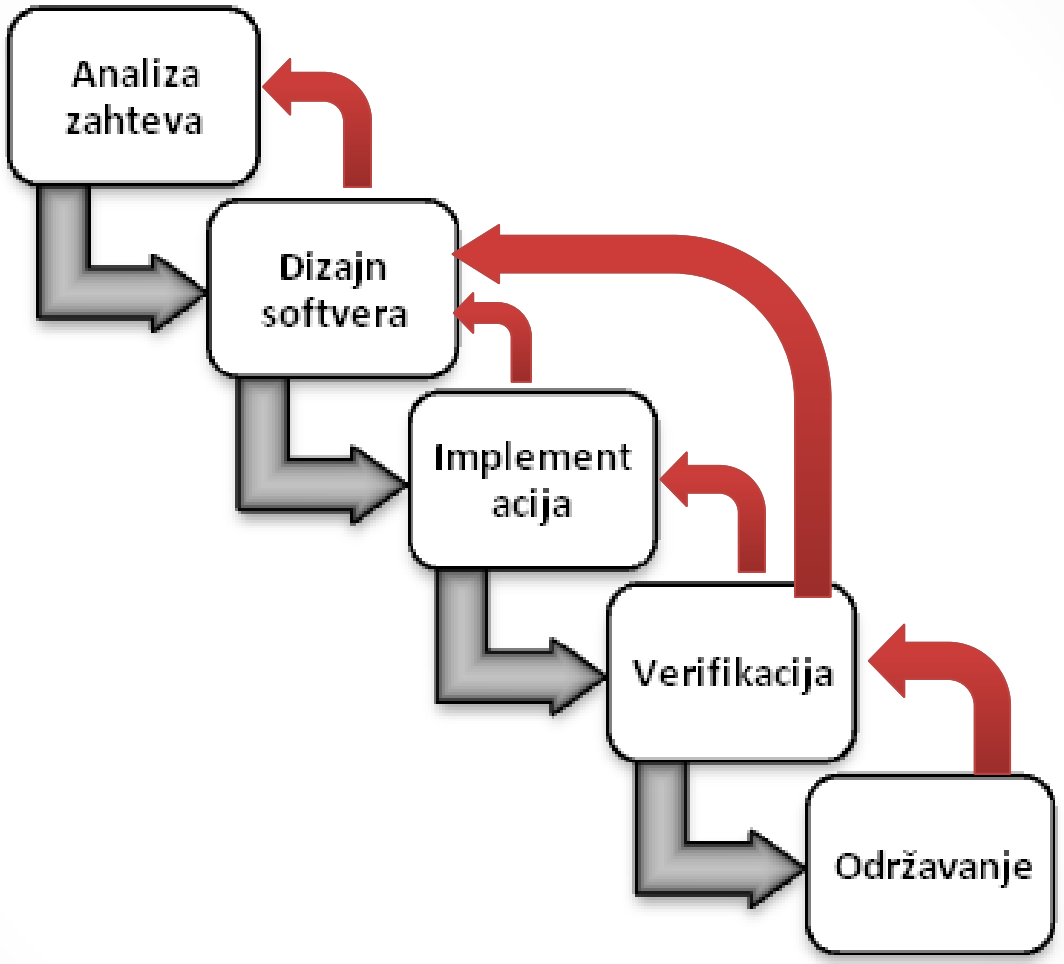
Problemi sa modelom vodopada

- Idealni slučaj:
  prolazi se kroz sve faze razvoja, članovi projektnog tima će se uključivati tačno u one faze u kojima su potrebni, završavaće posao za koji su bili planirani i
  odlaze iz tima kada više nema potrebe da budu na projektu.
- U svakom trenutku će biti poznato dokle se stiglo sa poslom i u kojoj fazi se projekat nalazi.
- Na žalost, ovakav idealan slučaj sekvencijalnog razvoja u praksi nije moguć
  zato što se često neplanski projektni tim vraća u prethodne faze.
- Često se dešava da se tokom kasnijih faza projekta pronalaze propusti i nepredviđene stvari nasleđene iz prethodnih faza, koje uzrokuju povratke u prethodne faze.
- Rezultat => kašnjenje projekta

Problemi sa modelom vodopada - primer



- Detalj plana projekta sa fazom verifikacije (testiranja), koja može da počne pošto se završe prethodne faze.

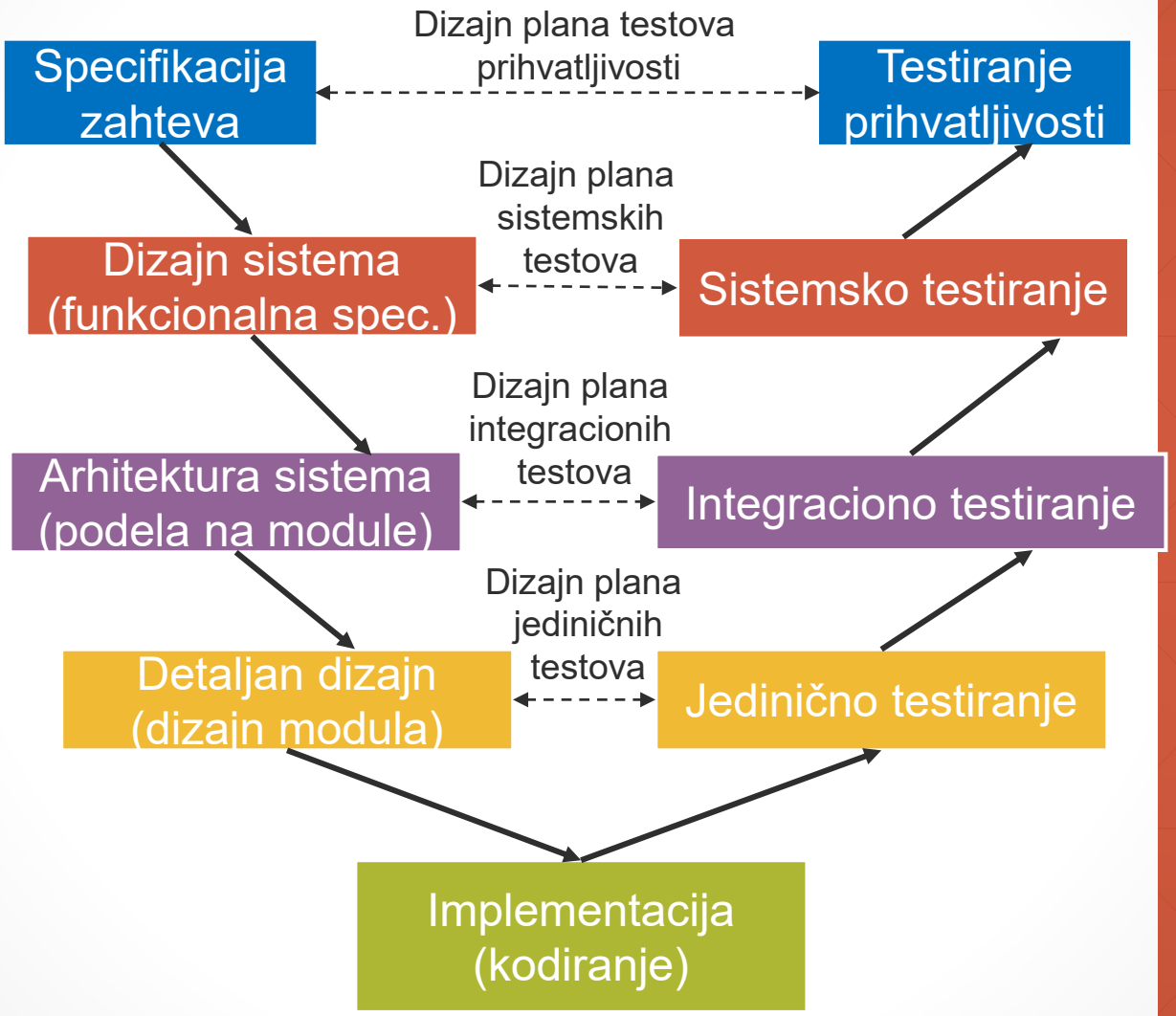


Neželjeni povratni tokovi u modelu vodopada

- Problemi koji se pronađu tokom faze verifikacije mogu biti toliko veliki da vrate projekat nazad u fazu implementacije, a nekada čak iz implementacije nazad u fazu dizajna softvera.
- Na primer, neko može da definiše zahtev u fazi analize, taj zahtev će se dizajnirati i implementirati, a onda tek u fazi verifikacije se zaključi da taj zahtev nema smisla.
- U tom slučaju projekat se vraća nazad na analizu. Kada se utvrdi šta bi u stvari trebao da bude smislen zahtev opet se prolazi kroz sve faze razvoja.
- Planiranje bazirano na količini rada, a ne vremensko (nije pogodno da se dinamički doda ili oduzme deo).

V - model

V - model
(sa naglaskom na kontroli kvaliteta)



- V-model (kao i vodopad) ima sve sekvencijalne faze: analizu zahteva, dizajn softvera, implementaciju i testiranje,
gde se u sledeću fazu prelazi samo ako je završena prethodna, uz 2 značajne izmene:
- Svaka od faza razvoja ima svoju odgovarajuću fazu testiranja, kojom se ta faza validirati.
- Posle svake faze razvoja, pre prelaska u sledeću fazu, se planira kako će se izvršiti verifikacija trenutne faze (iako se odgovarajuća faza verifikacije neće izvršiti sve do kraja projekta).

- Leva grana V: dekompozicija zahteva i kreiranje funkcionalne specifikacije i arhitekture sistema.
- Desna grana V: integracija celina i validacija.

V-model

- **Pravila koja definišu preduslove za prelazak u sledeću fazu:**
  - Iz faze analize zahteva se može preći u fazu specifikacije samo ako su analizirani zahtevi i ako je definisano kako će se ti zahtevi testirati tokom testa prihvatljivosti.
  - U fazu dizajna sistema se može preći ako je završena faza specifikacije sistema i definisano kako će se testirati kompletan sistem.
  - U fazu dizajna pojedinih modula se može preći ako je dizajnirana arhitektura sistema i definisano kako će se testirati komponente tokom integracije.
  - U fazu implementacije se može preći ako su dizajnirani moduli koji će se kodirati i ako je definisano kako će se ti moduli testirati.

- **Druga značajna izmena - definisanje više nivoa testiranja kojima se proveravaju različiti delovi sistema.**
- **Nivoi testiranja po V-modelu su:**
  - Jedinično testiranje (eng. *Unit testing*) kojim se testiranju pojedini delovi sistema (moduli, komponente, forme).
  - Integraciono testiranje (eng. *Integration test*) kojim se testira komunikacija, povezivanje i tokovi među modulima.
  - Sistemsko testiranje (eng. *System test*) kojim se testira sistem u celini.
  - Test prihvatljivosti (eng. *Acceptance test*) kojim krajnji korisnici potvrđuju da aplikacija radi upravo ono što im treba.

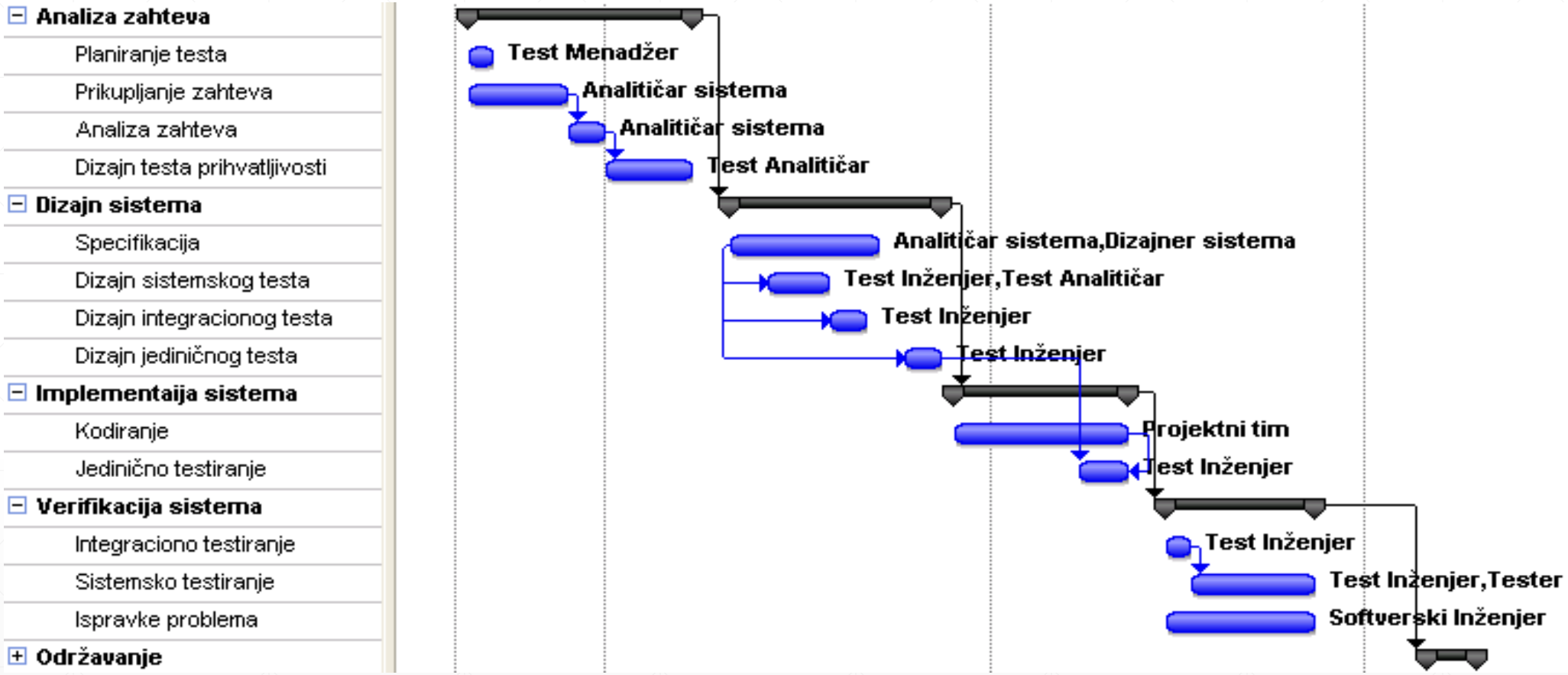
Validacija vs Verifikacija

- **Validacija:**
  - Uverenje da proizvod, usluga ili sistem zadovoljavaju potrebe korisnika i drugih identifikovanih zainteresovanih strana.
  - Treba da odgovori na pitanje:
    Da li pravimo pravi proizvod?
  - Sistem ili deo sistema (komponentu) treba da ocenimo ili u toku i obavezno na kraju procesa razvoja.
  - Aktivnosti pri validaciji: provera i testiranje realnog softverskog proizvoda, odnosno njegovih funkcionalnosti.

- **Verifikacija:**
  - Procena da li je proizvod, usluga ili sistem u skladu sa propisom, zahtevom, specifikacijom ili nametnutim uslovom.
  - To je interni proces, za razliku od validacije.
  - Treba da odgovori na pitanje:
    Da li pravimo proizvod na pravilan način?
  - Aktivnosti pri verifikaciji: provera dizajna, revizija programskog koda, dokumenata,... (eng. *Formal Review*, *Code Review*, i sl.)

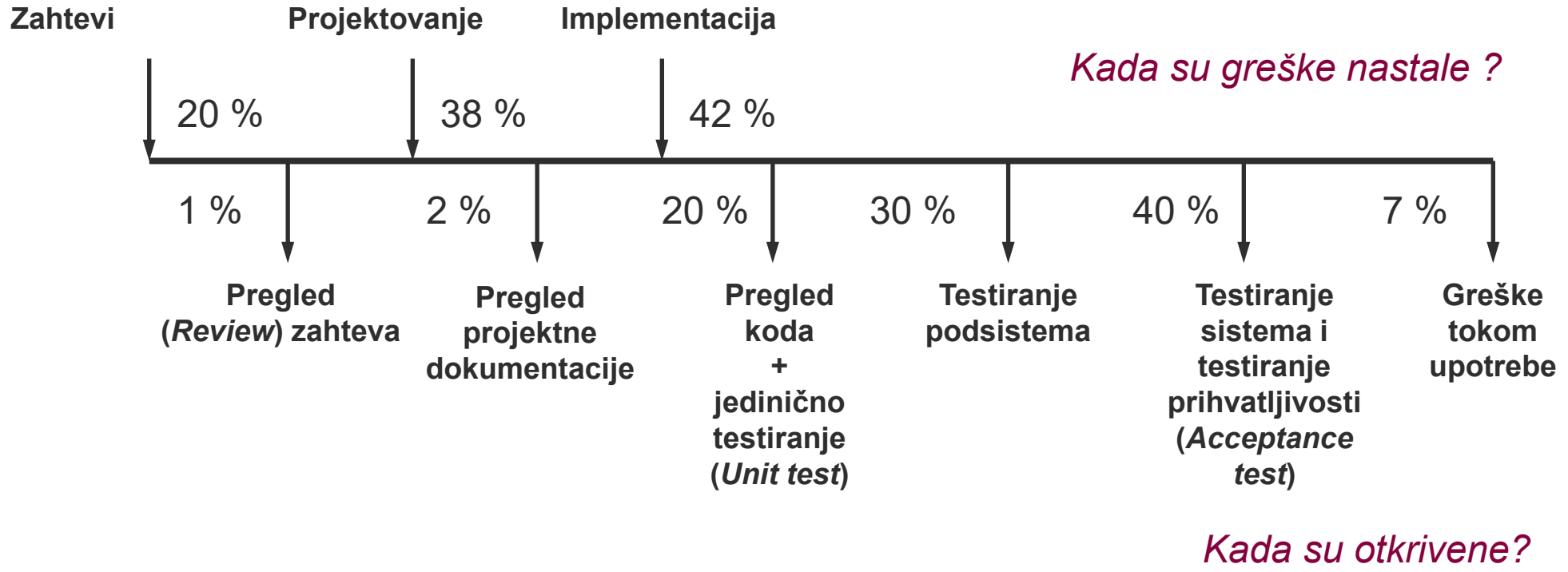
Izvor: PMBOK guide, IEEE.

Primer projektnog plana po V-modelu



- Neke aktivnosti testiranja vrše se u paraleli sa prethodnim fazama.

Statistika grešaka: nastanak i ispravka



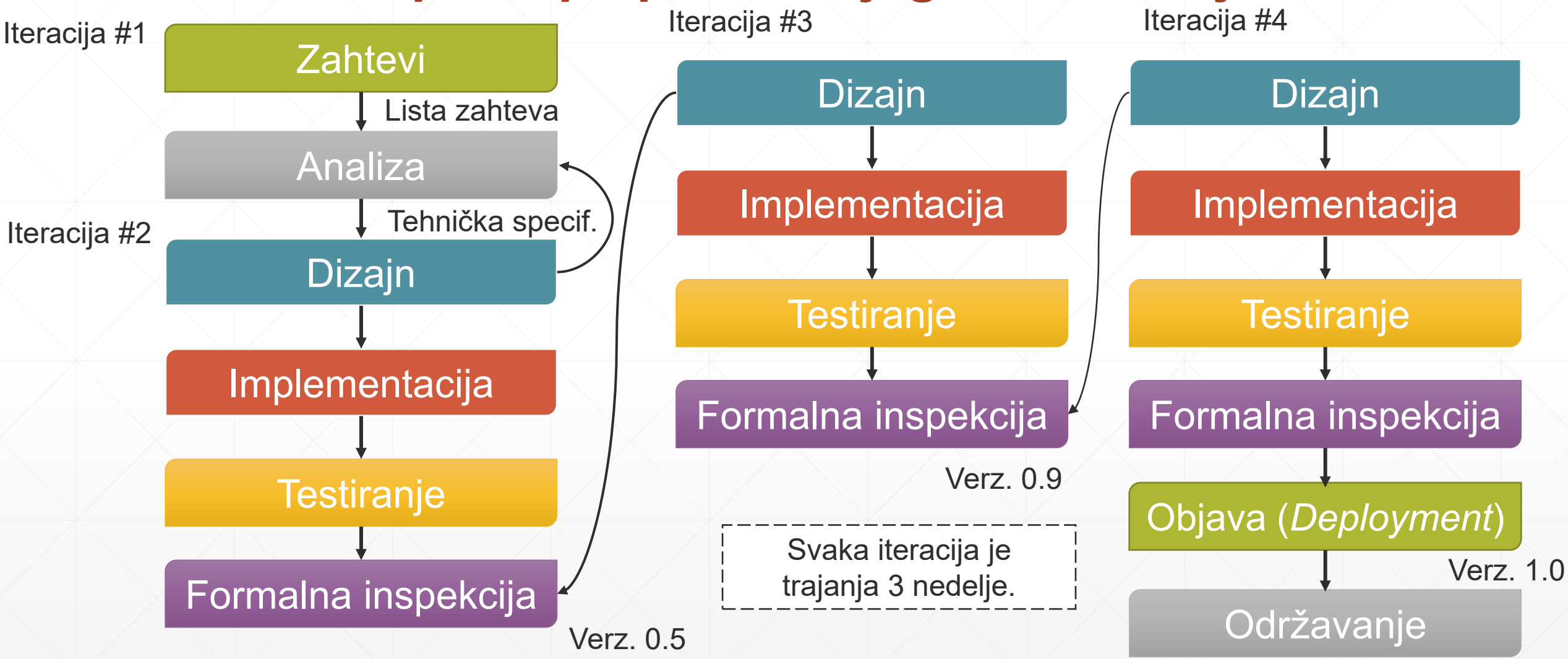
**Preuzeto iz: Software Metrics Symposium 1996, p. 176*

Iterativno inkrementalni model

Iterativni pristup

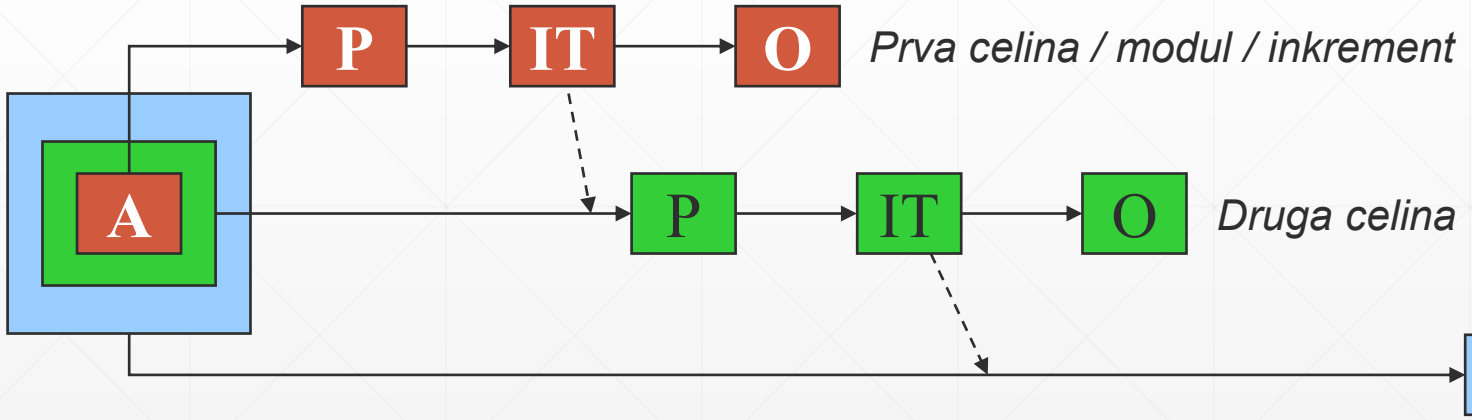
- Iterativni pristup uvodi pojam verzija.
- U ovom pristupu možemo početi sa nekom od softverskih specifikacija i da razvijemo neku polovičnu verziju softvera (npr. ver. 0.5) u prvim iteracijama. Nakon toga, izdizajniramo ostatak sistema, implementiramo, testiramo, i dođemo do prve verzije sistema (ver. 1.0), pa ukoliko postoji potreba za promenom softvera, kreiraće se nova verzija softvera u novoj iteraciji (ver. 2.0), itd.
- Svako izdanje iterativnog modela završava se u tačnom i fiksnom periodu, koji nazivamo iteracijom. Iterativni model nije moguć kod manjih projekata.
- Iterativni pristup omogućava pristup ranijim fazama (iteracijama).
- Kada koristiti iterativni pristup?
  - Kada su zahtevi jasno definisani i laki za razumevanje.
  - Kada je softverski sistem velikog obima.
  - Kada postoji otvoren zahtev za promenama u budućnosti.

Iterativni pristup - primer dijagrama razvoja



Inkrementalni model

- Inkrementalni model: zahtevi se dele u više samostalnih celina (modula / inkremenata).
- Prolazi kroz faze analize zahteva, projektovanja/dizajniranja, implementacije i testiranja (i kao i kod vodopada dodatno: održavanje/korišćenje softvera).
- Svako sledeće izdanje modula (inkrementa) daje pravu funkciju prethodnom izdanju.
- Ovaj model se koristi kada projekat ima dugačak raspored razvoja i kada kupac zahteva brzo oslobađanje celine po celinu proizvoda. U tom slučaju mogu se prioritirati zahtevi (npr. u sistemu prvo hardverski deo razvijati, pa tek onda softverski, ili prvo *backend* deo – srž app, pa onda interfejse - *API*, ka drugim sistemima, na kraju GUI...).



Legenda:
A – analiza zahteva
P – projektovanje (faza dizajna)
IT – implementacija i testiranje
O – održavanje/korišćenje

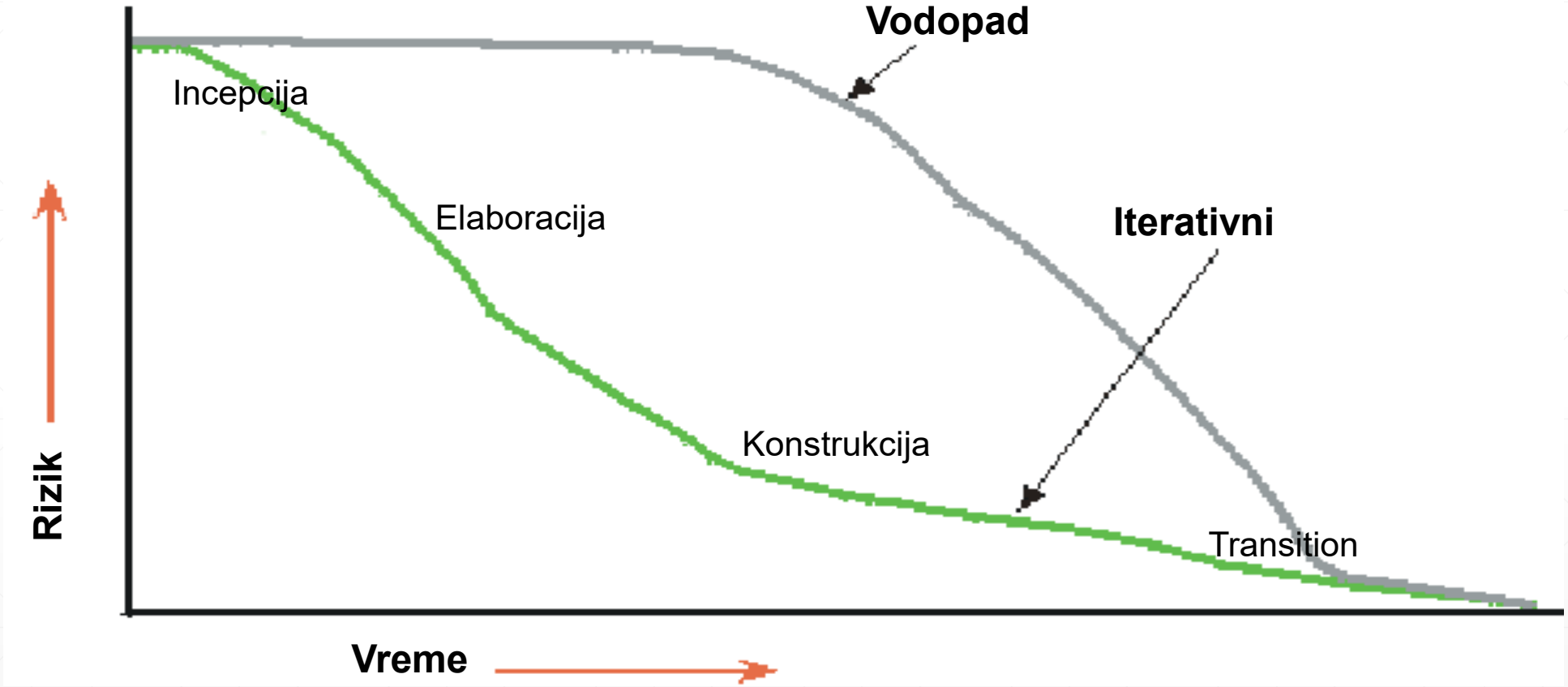
Iterativno inkrementalni model

- Kombinacija dva pristupa.
- Započinje sa poznatim skupom zahteva, a razvoj se odvija po celinama - **inkrementima**.
- Prva faza uključuje deo zahteva, svaka sledeća realizuje deo preostalih zahteva, sve dok se sistem ne kompletira.
- U okviru svake faze aktivnosti razvoja se sprovode sekvencijalno, a među fazama može postojati delimično preklapanje aktivnosti.
- Druga izmena u odnosu na model vodopada – uvode se **iteracije**.
- Problem kod vodopada:
  faze su mogle trajati i po nekoliko meseci.
- Iteracija u ovom modelu je fiksni period koji traje po 2-3 nedelje, u kojima će se proizvesti neki zaokružen rezultat projekta (*deliverables*) ili neka verzija softverskog produkta (npr. mogući završeci iteracije: završena lista korisničkih zahteva, izrađen prototip, završena tehnička specifikacija, završena polovična verzija celokupnog softvera, završena prva alfa verzija, beta verzija, itd.)

Iterativni naspram Vodopad modela (1)

Preuzeto iz: I. Jacobson, G. Booch, J. Rumbaugh:
The Unified Software Development Process, 1999

Iterativni naspram Vodopad modela (2)



IBM/Rational Unified Process, v2003

Iterativno inkrementalni model

+
Korisnici češće vide promene i brže ukazuju na greške koje se brže i efikasnije ispravljaju.
Velika je verovatnoća da će se funkcionalnosti biti upravo to što krajnjim korisnicima treba zato što je kraći period isporuke.
Lako reagovanje na promene, u slučaju da se promeni zahtev koji bi bio implementiran u budućoj iteraciji, tim ne bi primetio promenu.

-
Teže je sagledati širu sliku projekta pošto je fokus uvek na manjem inkrementu koji se dodaje.
Teže je objasniti koliko će trajati i koštati ceo projekat.

Zadatak 1

- Neka kompanija želi da unapredi svoj softverski proizvod.
  Funkcionalnosti koje treba da budu dodate opisane su u 5 slučajeva korišćenja.
  Napor koji je potrebno uložiti, izražen u jedinici čovek/dan (č/d), dat je u zagradi, pored svake faze:
  - Dizajniranje jednog slučaja korišćenja (6 č/d)
  - Implementacija jednog slučaja korišćenja (6 č/d)
  - Pripremanje testova za jedan slučaj korišćenja (1 č/d);
    napomena: kompletan dizajn slučaja korišćenja mora postojati pre početka pripreme testa
  - Testiranje jednog slučaja korišćenja (1č/d)
  - Integracija jednog slučaja korišćenja u postojeći sistem (1č/d); napomena: ova faza obuhvata i integraciono testiranje.
  - Upravljanje projektom (1č/d svake nedelje tokom čitavog trajanja projekta)
- Tim koji realizuje projekat čini 7 članova:
  - 1 menadžer projekta (zadužen za upravljanje projektom)
  - 1 softverski dizajner
  - 2 programera (mogu da rade dizajn, ukoliko je potrebno)
  - 1 tester (koji priprema i pokreće testove)
  - 1 integrator

Zadatak 1 (nastavak)

- Napraviti projektni plan i predstaviti sve aktivnosti u Gantovom dijagramu, za dva različita modela procesa:

  a) Projekat koristi tradicionalni model vodopada (odnosno svi slučajevi upotrebe se obrađuju zajedno u svakoj fazi).

  b) Projekat koristi inkrementalni model (odnosno slučajevi korišćenja se dodaju u sistem po jedan u svakom trenutku).

  Ukoliko je moguće, cilj je smanjiti ukupno vreme, identifikovanjem aktivnosti koje mogu da se preklapaju. Za svaki model, odrediti ukupno kalendarsko vreme trajanja projekta.

Zadatak 1 - rešenje: model vodopada

Faza	I nedelja					II nedelja					III nedelja					IV nedelja					V nedelja					VI nedelja					VII nedelja				
	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P
Dizajniranje																																		
Implementacija																																		
Pripr. testova																																		
Testiranje																																		
Integracija																																		
Upravljanje pr.	X					X					X					X					X					X					X				

- Prepostavka: dizajner tokom prve dve nedelje radi zajedno sa dva programera na fazi dizajniranja aplikacije

Faza	I nedelja					II nedelja					III nedelja					IV nedelja					V nedelja					VI nedelja					VII nedelja				
	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P
Dizajniranje	3	3	3	3	3	3	3	3	3	3																								
Implementacija											2	2	2	2	2	2	2	2	2	2	2	2	2	2	2										
Pripr. testova											1	1	1	1	1																				
Testiranje																										1	1	1	1	1					
Integracija																															1	1	1	1	1
Upravljanje pr.	X					X					X					X					X					X					X				

Faza	I nedelja					II nedelja					III nedelja					IV nedelja					V nedelja					VI nedelja					VII nedelja				
	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P
Dizajniranje																																		
Implementacija																																		
Pripr. testova																																		
Testiranje																																		
Integracija																																		
Upravljanje pr.	X					X					X					X					X					X					X				

Ukupno trajanje projekta:

35 dana

Zadatak 1 - rešenje: inkrementalni model

Faza	I nedelja					II nedelja					III nedelja					IV nedelja					V nedelja				
	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P
Dizajniranje																									
Implementacija																									
Pripr. testova																									
Testiranje																									
Integracija																									
Upravljanje pr.	X					X					X					X					X				

- Svaki slučaj korišćenja je označen drugom bojom.

Faza	I nedelja					II nedelja					III nedelja					IV nedelja					V nedelja				
	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P	P	U	S	Č	P
Dizajniranje	3	3	2	2	2	2	2	2	1	1	1	1	1	1	2	2	2								
Implementacija			1	1	1	1	1	1	2	2	2	2	2	2	1	1	1	2	2	2	2	2			
Pripr. testova							1	1		1					1				1						
Testiranje									1			1				1				1			1		
Integracija										1			1				1				1			1	
Upravljanje pr.	X					X					X					X					X				

Ukupno trajanje projekta:

23.5 dana

Zadatak 2

- Postavljeni ste za menadžera projekta za razvoj jednog malog softverskog proizvoda. Početne procene opterećenja nalaze se u tabeli. U timu imate troje ljudi na raspolaganju, koji su angažovani sa punim radnim vremenom (*full-time*), svi dovoljno dobri da obavljaju bilo kakvu aktivnost na projektu.

- a) Opisati projektni plan rečima i objasniti koje sve mogu biti aktivnosti upravljanja projektom.

- b) Napraviti detaljan projektni plan i nacrtati Gantov dijagram, koji deli ove aktivnosti između tri člana tima i obuhvata prekretnice. U tabeli nisu date aktivnosti upravljanja projektom (sastanci i slično), ali potrebno je da i njih isplanirate.

- c) Koji su potencijalni rizici u vašem planu?

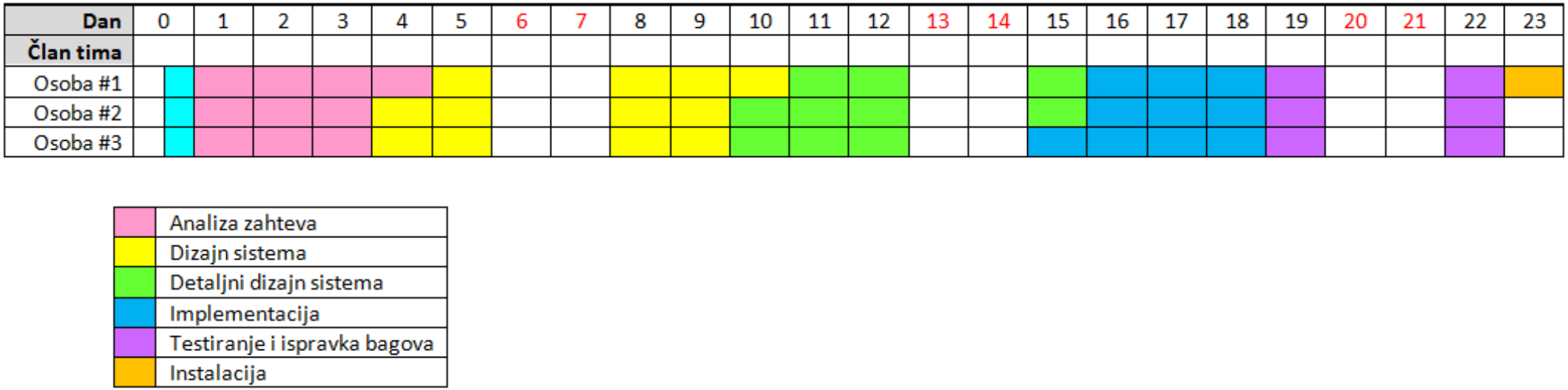
Aktivnost	Obim posla (čovek/dana)
Analiza zahteva	10
Dizajn sistema	12
Detaljni dizajn sistema	10
Implementacija	10
Testiranje i ispravka bagova	6
Instalacija	1

Zadatak 2 - rešenje (a): upravljanje projektom

- Mali projekat. Rešenje bazirano na modelu vodopada sa nekim modifikacijama.
- Prepostavka je da jedna aktivnost prati drugu, ali između dve aktivnosti neki članovi mogu da počnu da rade sa novim aktivnostima
- Trajanje projekta: 10+12+10+10+6+1 = 49 dana / 3 člana tima = 16.3 radnih dana
- Upravljanje projektom - koliko traje?
- Najmanje jednočasovni sastanak nedeljno, u kome učestvuju svi članovi tima.
- Trajanje aktivnosti: 1^h x 3 člana x 3 nedelje **= 9^h**
- Druga aktivnost upravljanja projektom - upravljanje konfiguracijom (koja uključuje i obuku)
- Trajanje aktivnosti: obuka na početku projekta 1^h x 3 člana = 3^h
  + upravljanje konf. tokom projekta 1^h x 3 člana x 3 nedelje = 9^h **Σ = 12^h**
- Upravljanje projektom ukupno: **Σ = 9^h + 12^h = 21^h** Ovo ćemo zaokružuti na ~ 24^h ili 3 č/dan

Zadatak 2 - rešenje (b): Gantov dijagram

- Prva aktivnost se odvija svake nedelje i zbog vidljivosti nije uključena u dijagramu.
- Takođe, ni druge aktivnosti nisu uključene u dijagramu, jer su distribuirane tokom celog projekta. Jedini deo koji je prikazan je deo obuke na početku.



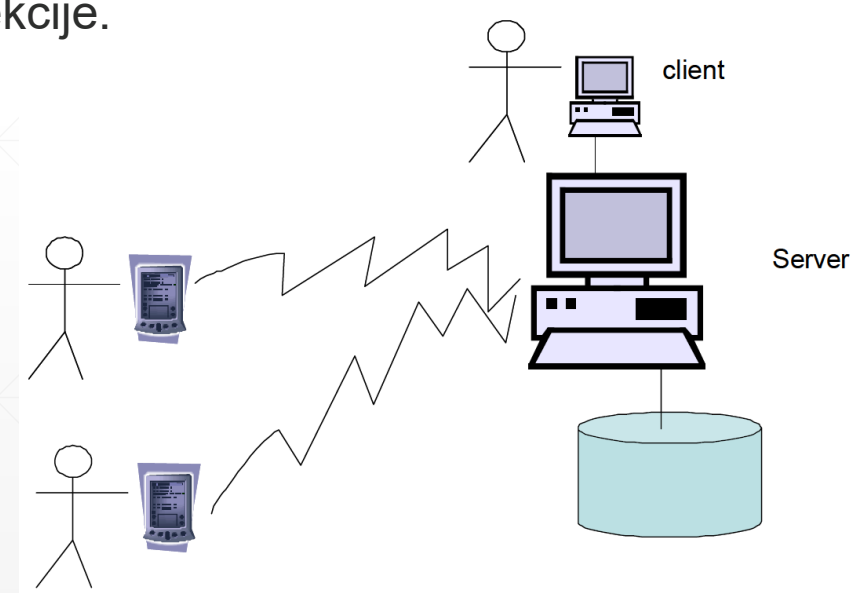
- Prekretnice (*milestones*) predstavljaju svaku promenu boje - promenu aktivnosti.

Zadatak 2 - rešenje (c): rizici

- Potencijalni rizici kašnjenja u ovom planu:
  - Ponekad jedna osoba radi samo jedan dan na kraju aktivnosti.
    Ovo može izazvati kašnjenje u slučaju bolesti na primer.
  - Upravljanje aktivnostima nije jasno navedeno u dijagramu, pa zbog toga može da se izazove blagi zastoj u celom projektu.
  - Jedna osoba ne može biti u stanju da instalira proizvod (zbog nedostatka znanja ili bolesti), pa isporuka može da se odloži.

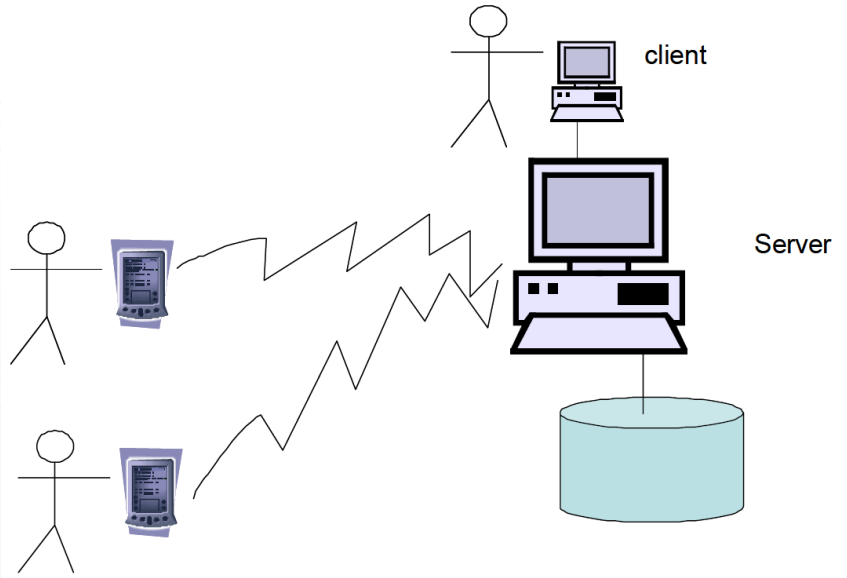
Zadatak 3

- Pretpostavimo da razvijate softver za beogradski metro. Svaki zaposleni ima svoj tablet računar na kome može da vidi sve informacije o svojim aktivnostima i aktivnostima drugih zaposlenih (ljudi na dužnosti, njihove trenutne i planirane aktivnosti, urađene aktivnosti, njihova stanja,... itd.). Svaki zaposleni može da unese/ažurira svoje aktivnosti ili ako ima dozvoljene privilegije može kreirati novu aktivnost za nekog zaposlenog.
- Sve informacije se čuvaju na centralizovanom serveru preko koga zaposleni komuniciraju preko svojih tablet računara (kao na slici). Sistem može da se administrira preko lokalne konekcije.
- Vi ste tim lider projektnog tima i treba da napravite projektni plan.
- Treba da razvijete softver za tablet PC
  (grafički korisnički interfejs i komunikacijski deo), softver za komunikaciju
  i softver potreban za server za upravljanje podacima.



Zadatak 3 (nastavak)

- Imate tim od 6 ljudi - svi oni mogu da budu dizajneri sistema, programeri, testeri ili da pišu dokumentaciju.
- Ugovor koji ste potpisali zahteva da implementirate kompletan softverski sistem u 25 nedelja.
- Sav hardver i razvojni alati su dostupni timu.
- U postupku rešavanja treba koristiti V model.
- a) Prepoznati glavne aktivnosti i podaktivnosti, i rasporediti ih među članovima tima.
- b) Nacrtati Gantov dijagram za ovaj projekat i dijagram raspodele resursa u projektnom timu



Zadatak 3 - rešenje (a)

- Glavne aktivnosti:

ID	Glavna aktivnost	Podaktivnost
A1	Celokupni sistem	
A1.1		Korisnički zahtevi
A1.2		Dizajniranje sistema
A1.3		Integracija sistema
A1.4		Validacija
A1.5		Verifikacija i ispravka bagova
A1.6		Isporuka
A2	Softver za tablet	
A2.1		Korisnički zahtevi
A2.2		Dizajniranje podsistema
A2.3		Implementacija
A2.4		Verifikacija i ispravka bagova

ID	Glavna aktivnost	Podaktivnost
A3	Softver za server	
A3.1		Korisnički zahtevi
A3.2		Dizajniranje podsistema
A3.3		Implementacija
A3.4		Verifikacija i ispravka bagova
A4	Softver za komunikaciju	
A4.1		Korisnički zahtevi
A4.2		Dizajniranje podsistema
A4.3		Implementacija
A4.4		Verifikacija i ispravka bagova
A5	Upravljanje projektom	
	Obuhvata kompletan menadžment, projektnu dokumentaciju, proveru kvaliteta, tehničku podršku	

Zadatak 3 - rešenje (a) - diskusija

- Opterećenje treba distribuirati na najefikasniji način
- V model se koristi u svakoj aktivnosti (A1-A4)
- Softver za komunikaciju će zahtevati manje posla, nego razvoj softvera za tablete i serversku stranu.
- Alokacija resursa u timu, na primer:
  - Aktivnosti A1, A4 - inženjeri I1 i I2
  - Aktivnost A2 - inženjeri I3 i I4
  - Aktivnost A3 - inženjeri I5 i I6
- Kako se radi o V modelu, aktivnosti validacije i verifikacije dolaze sa fazom zahteva i dizajniranja sistema, a nakon implementacije i integracije.
- Inženjeri iz A2 i A3 treba da pomognu u verifikaciji/validaciji sistema.
- Stvaranje dokumentacije je deo aktivnosti u skladu sa V modelom.

Zadatak 3 - rešenje (b) - Gantov dijagram

GLAVNA AKTIVNOST	PODAKTIVNOST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Celokupni sistem																										
	Zahtevi		I1-I6																							
	Dizajn sistema					I1, I2																				
	Integracija sistema																			I1, I2					I1, I2	
	Validacija					I1, I2																I1-I6				
	Verifikacija							I1, I2																	I1-I6	
	Isporuka																									I1-I6
Softver za tablet																										
	Zahtevi				I3, I4																					
	Dizajn sistema									I3, I4																
	Implementacija															I3, I4										
	Verifikacija					I3, I4													I3, I4							
Softver za server																										
	Zahtevi				I5, I6																					
	Dizajn sistema									I5, I6																
	Implementacija															I5, I6										
	Verifikacija					I5, I6													I5, I6							
Softver za komunikaciju																										
	Zahtevi									I1, I2																
	Dizajn sistema											I1, I2														
	Implementacija																I1, I2									
	Verifikacija												I1, I2						I1, I2							
Upravljanje projektom																										

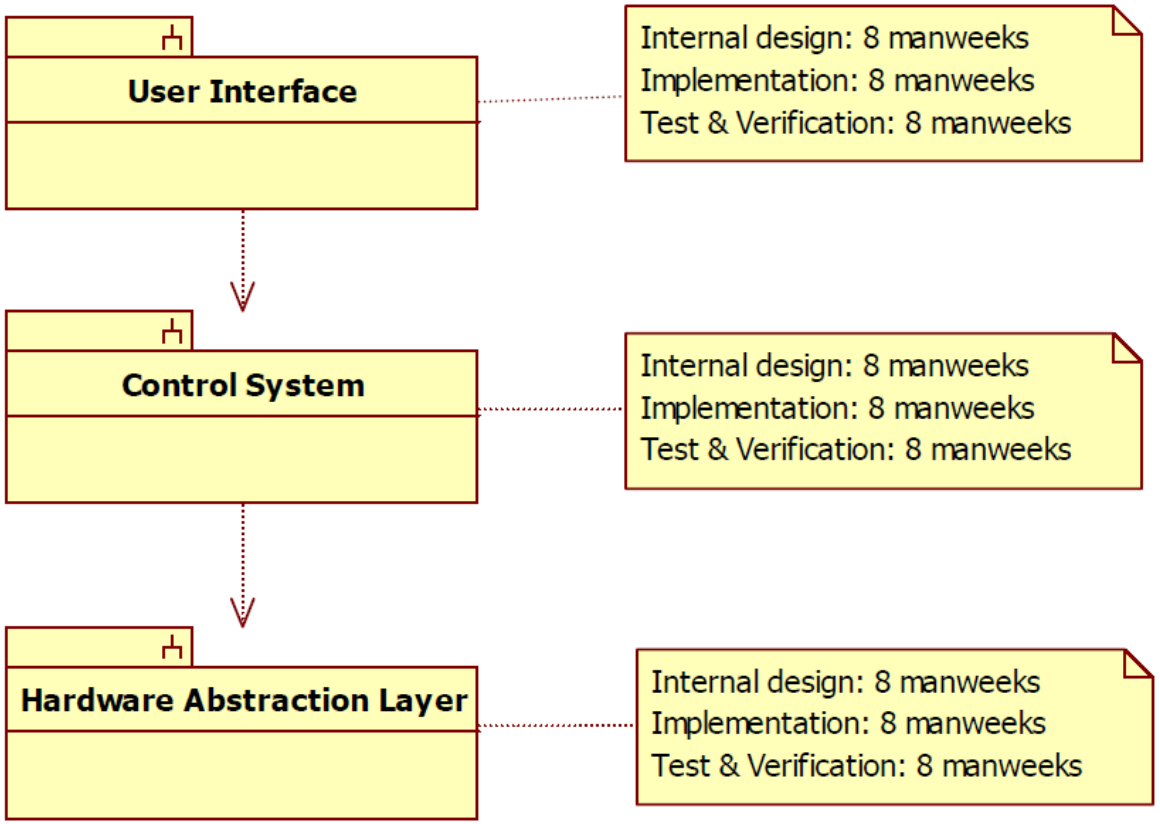
Zadatak 3 - rešenje (b) - Raspodela resursa

- Svi članovi počinju sa zahtevima i dizajniranjem sistema.
- I1 i I2 nastavljaju na sistemskom nivou, drugi podtimovi rade svoje podsisteme.
- Svi članovi rade verifikaciju i validaciju sistema.
- Poslednje dve nedelje su uzete za dodatni rad, ako bi bilo potrebno!

Inženjer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
I1	a1	a1	a1	a1	a1	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a1	a1	a1	a1	a1	a1	a1	a5
I2	a1	a1	a1	a1	a1	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a4	a1	a1	a1	a1	a1	a1	a1	a5
I3	a1	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a1	a1	a1	a1	a1	a1	a5	a5
I4	a1	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a2	a1	a1	a1	a1	a1	a1	a5	a5
I5	a1	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a1	a1	a1	a1	a1	a1	a5	a5
I6	a1	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a3	a1	a1	a1	a1	a1	a1	a5	a5
Menadžer projekta	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5	a5

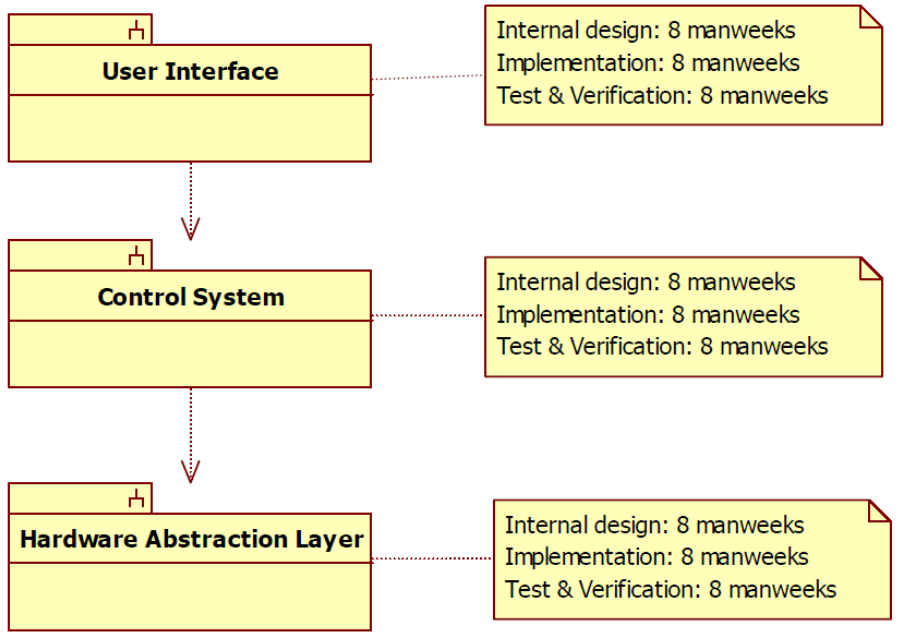
Zadatak 4

- Radite na projektu koji će razviti softver za seriju mašina za pranje veša. Zajedno sa klijentima, definisani su svi sistemski zahtevi.



- Vi ste se složili koja dugmad je potrebna i šta će biti prikazano na prednjem delu mašine za veš, i definisali ste hardverski apstrakcioni sloj, koji treba da obezbedi prenosivost za novi hardver u budućnosti (samo jedan sloj će morati da se menja ukoliko se hardver menja). Pogledati dijagram.

Zadatak 4 (nastavak)



- Tim čine tri čoveka (uključujući Vas) koji rade puno radno vreme na projektu. Svako može uraditi bilo koji zadatak (aktivnost) na projektu. Do danas, Vi i Vaš tim ste proveli 4 nedelje u dogovorima sa klijentom oko sistemskih zahteva, i možete proceniti vreme razvoja za svaki deo projekta, kao što je opisano u napomenama na slici.
- Potrebno je prikazati modele razvoja za ostatak projekta:
  - a) iterativni
  - b) inkrementalni
- Za svaki od tih modela razvoja, definisati prekretnice (*milestones*) i krajnje rezultate (*deliverables*), sa datumima. Takođe ukratko opisati kako svaki model razvoja utiče na rizike i koji su mogući načini saradnje sa klijentima. Nije neophodno crtati Gantov dijagram ili dijagram alokacije resursa, samo listu prekretnica i krajnje rezultate, kao i kratku diskusiju.

Zadatak 4 - rešenje

- Pretpostavke:
  - Mi ćemo razmatrati da svi ljude uvek zajedno rade na istom zadatku.
  - Pretpostavljamo da će mesečno biti potrošeno 2 dana na sastanke o upravljanju projektom (ukupno 2 nedelje za ceo projekat).
  - Pretpostavljamo da je dokumentacija za svaki zadatak uključena u date cifre.

Zadatak 4 - rešenje (a): Iterativni model

- Pretpostavke:
  - Svaki zadatak delimo u 2 podzadatka, npr. Implementacija 1 + Implementacija 2
  - Svaki podzadatak taje 4/3 kalendarske nedelje
  - Pretpostavimo da nam je potrebno 6 čovek-nedelja (tj. 2 kalendarske nedelje) za integraciono testiranje, ispravku bagova i validaciju celog sistema
  - Vršimo dve iteracije, prva se završava sa isporukom osnovnih funkcionalnosti svake komponente (verzija 0.5) i poslednja se završava sa isporukom kompletnog, ispravnog sistema (verzija 1.0)
- Prekretnice i rezultati:
  - 4. nedelja: završen Dizajn1 od UI, Dizajn1 od CS i Implementacija1 od CS
  - 8. nedelja: završen Dizajn1 od HAL, Implementacija1 od UI i Testiranje1 od CS
  - 13. nedelja: završena Implementacija1 i testiranje1 od HAL, i Testiranje1 od UI
  - **REZULTAT: Verzija sistema 0.5 (Podzadatak1 završen za sve komponente)**
  - 16. nedelja: završen Dizajn2 od UI, CS, HAL
  - 21. nedelja: završena Implementacija2 od CS i HAL, i Testiranje2 od CS
  - 26. nedelja: završeno Testiranje2 od HAL, Implementacija2 od UI i Testiranje2 od UI
  - **REZULTAT: Verzija sistema alfa (Podzadatak2 završen za sve komponente)**
  - 28. nedelja: završeno integraciono testiranje, validacija, ispravke
  - **REZULTAT: Konačna verzija sistema (1.0)**

Zadatak 4 - rešenje (b): Inkrementalni model

- Pretpostavke:
  - Svaki zadatak traje 8/3 kalendarskih nedelja
  - Odredili smo 2 dana mesečno za sastanke oko upravljanja projektom
  - Prepostaviti da postoji potrebne 2 nedelje za testiranje integracije i zato je rizik veoma veliki u ovom razvojnom modelu, pa ćemo dodati 2 nedelje za otklanjanje grešaka
  - Smatramo da je dobro da se izgradi sistem od dna ka vrhu, odnosno sloj HW apstrakcije prvo, u suprotnom ćemo morati da izgradimo neke stabove, da bi se obavljali najjednostavniji testovi na gornjim slojevima

- Prekretnice i rezultati:
  - 8. nedelja: završen HW verzija 1.0 (dizajn, implementacija i testiranje HW)
  - **REZULTAT: HW verzija 1.0**
  - 17. nedelja: završen CS v 1.0 (dizajn, implementacija i testiranje CS)
  - **REZULTAT: CS verzija 1.0 (CS+HW)**
  - 26. nedelja: završen UI v 1.0 (dizajn, implementacija i testiranje UI)
  - **REZULTAT: Alfa verzija sistema (UI+CS+HW)**
  - 28. nedelja: završeno integraciono testiranje, validacija, ispravke
  - **REZULTAT: Konačna verzija sistema (1.0)**

Zadatak 4 - rešenje - diskusija

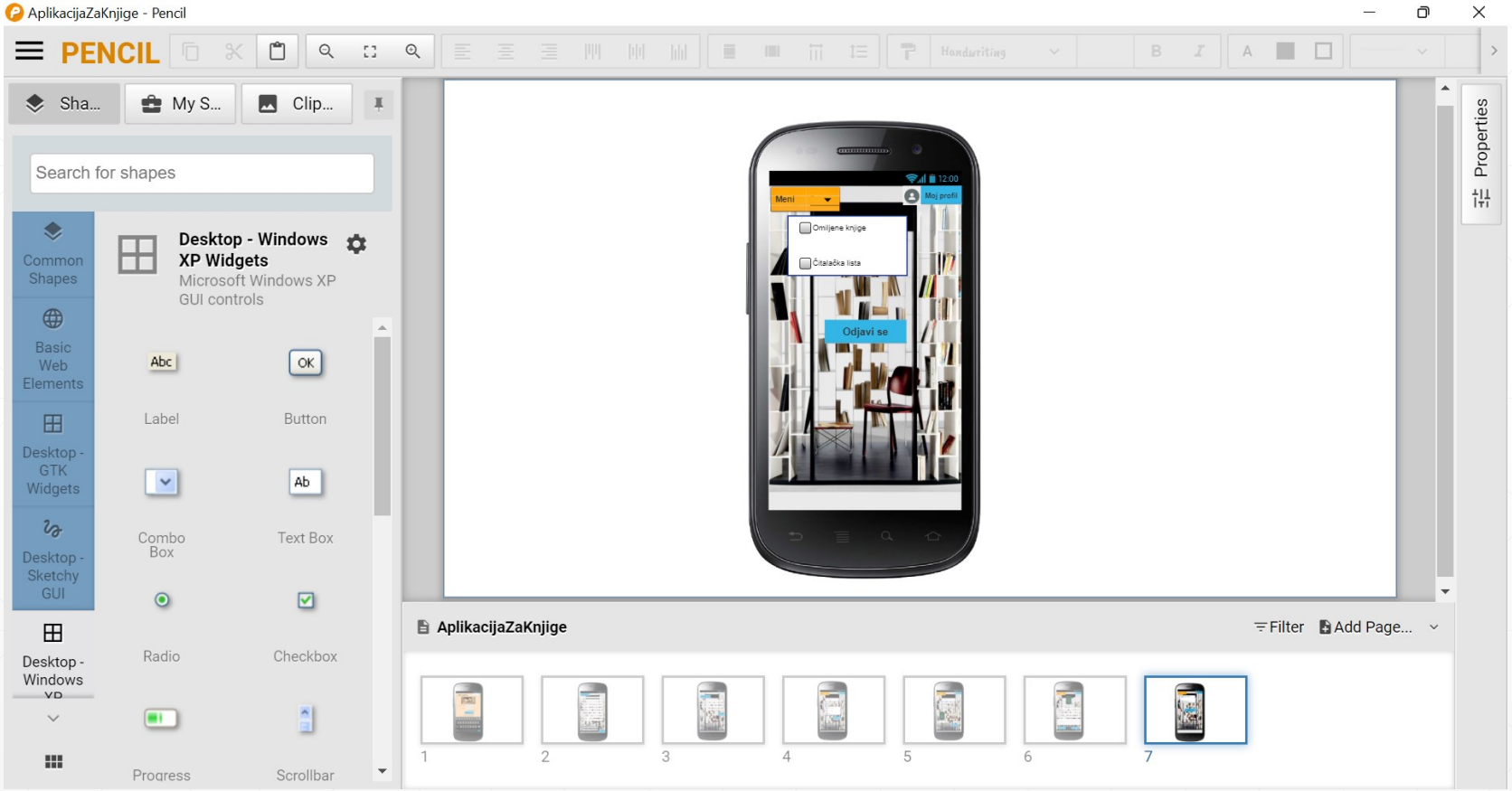
- Ukupno vreme:
  - 3 komponente x 3 zadatka x 8 čovek/nedelja po zadatku / 3 čoveka
    + 2 nedelje za upravljanje projektom + 2 nedelje za validaciju i ispravke ~ 28 nedelja
- Rezime:
  - Oba modela traju približno 28 nedelja, ali su prekretnice i rezultati koji su definisani na projektu različiti za ove modele, takođe načini komunikacije sa klijentom su različiti, što dovodi do različitih vrsta rizika projekata, pa su različiti i načini rukovanja tim izuzecima.
  - U iterativnom modelu isporučujemo ceo sistem na pola projekta, koji pokazuje kupcu kompletan sistem (prototip faza: sistem bez svih funkcionalnosti i koji još uvek nije pravilno testiran), što omogućava povratne informacije (npr. kupac želi nešto da promeni).
  - U inkrementalnom modelu, za slučaj da kasnimo, moramo da smanjimo karakteristike komponenti ugrađenih u kasnijim fazama (UI i eventualno CS). Drugi potencijalni rizik ovog modela je da kupac u kasnijim fazama traži da se promeni nešto u nižim slojevima.

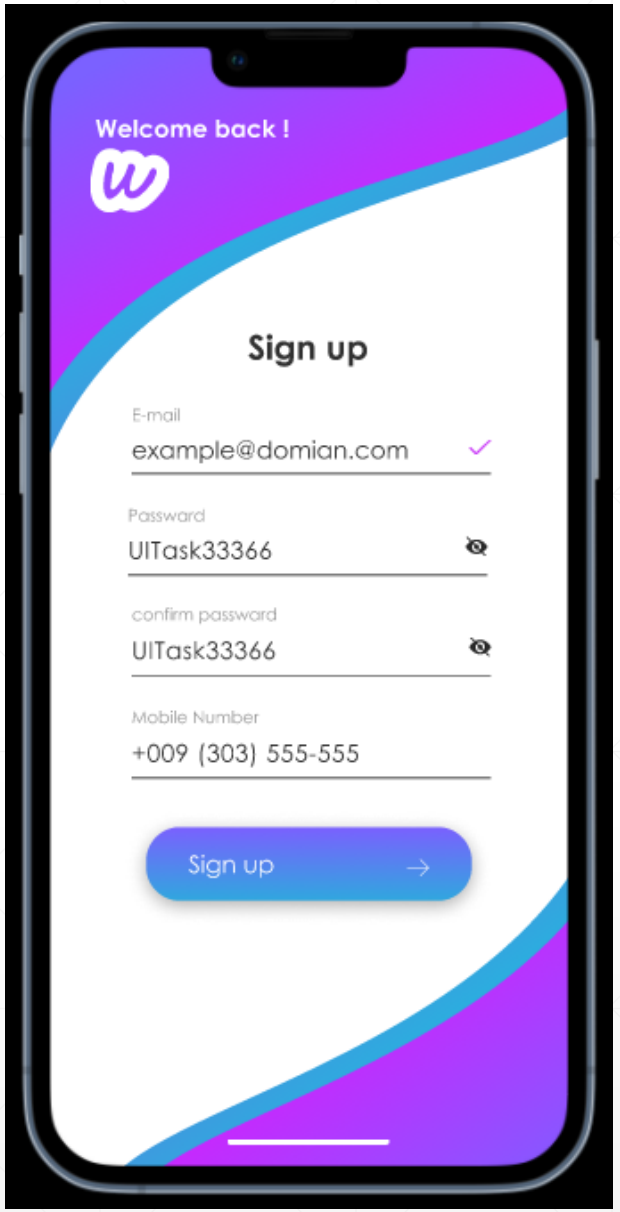
Evolutivni (prototipski) model

Evolutivni (prototipski) model

- **Područja primene:** Zahtevi u početku nisu precizni ili se često menjaju.
- **Prototip - izvršivi softverski sistem**,
  - Značajni delovi završnog proizvoda su već završeni
    (npr. grafički korisnički interfejs - GUI, neke osnovne funkcionalnosti),
  - Ostali delovi tek treba da se realizuju (npr. specijalni slučajevi).
- **Proces primene:**
  - Prototip, koji može biti odbačen
    (radi se kao dodatak analizi zahteva, brzo pravljenje prototipa - *rapid prototyping*)
  - Postupno napredovanje ka finalnom proizvodu
    (evolutivni razvoj softvera - *evolutionary software development*)

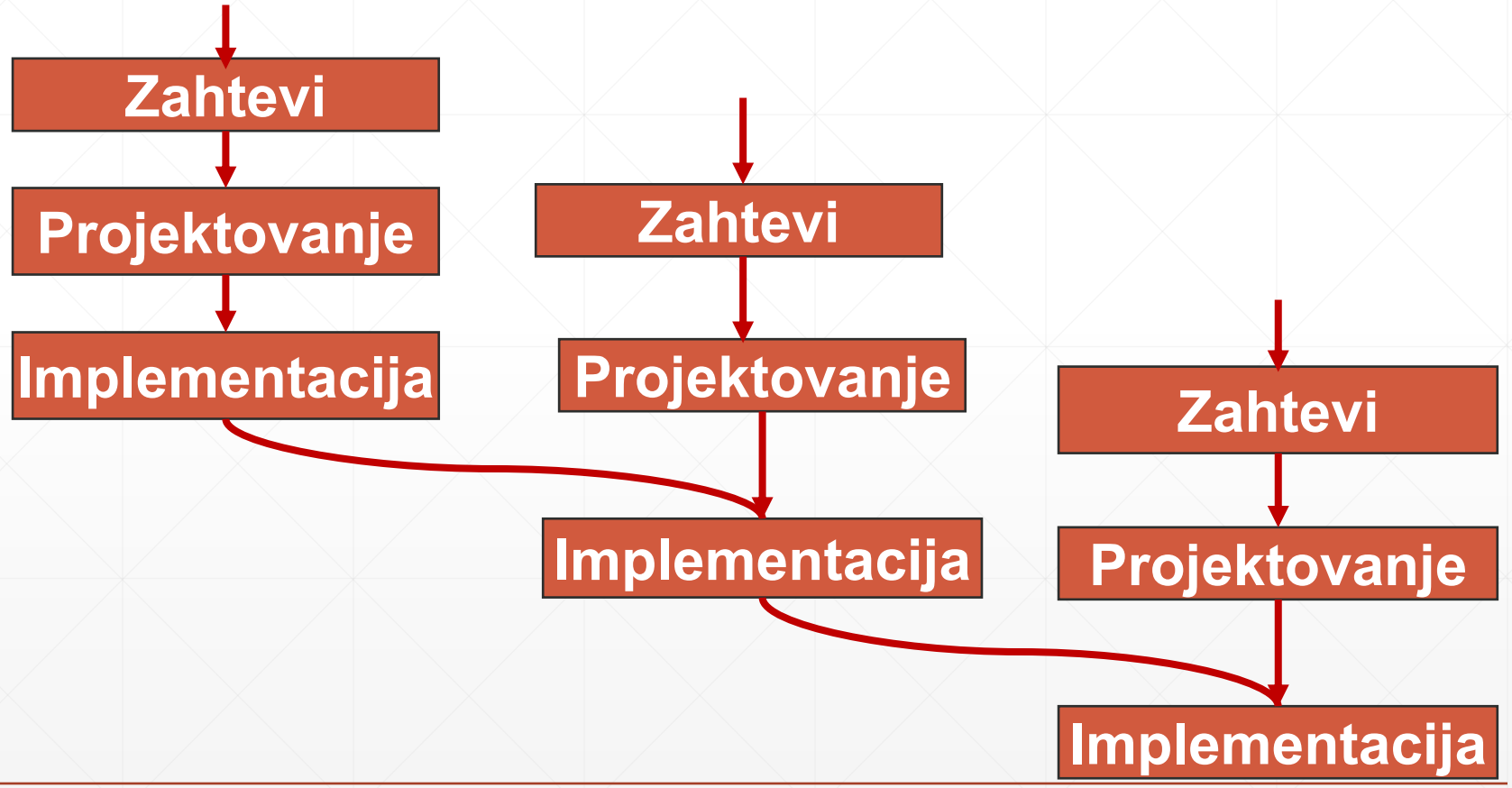
Primeri prototipa mobilnih aplikacija u razvojnim alatima *Pencil* i *Figma*





Evolutivni (prototipski) model

- Razvija sistem po fazama, ali za razliku od inkrementalnog modela dopušta da zahtevi inicijalno nisu sasvim precizirani i definisani. Zahtevi se inicijano parcijalno definišu i preciziraju u kasnijim fazama.



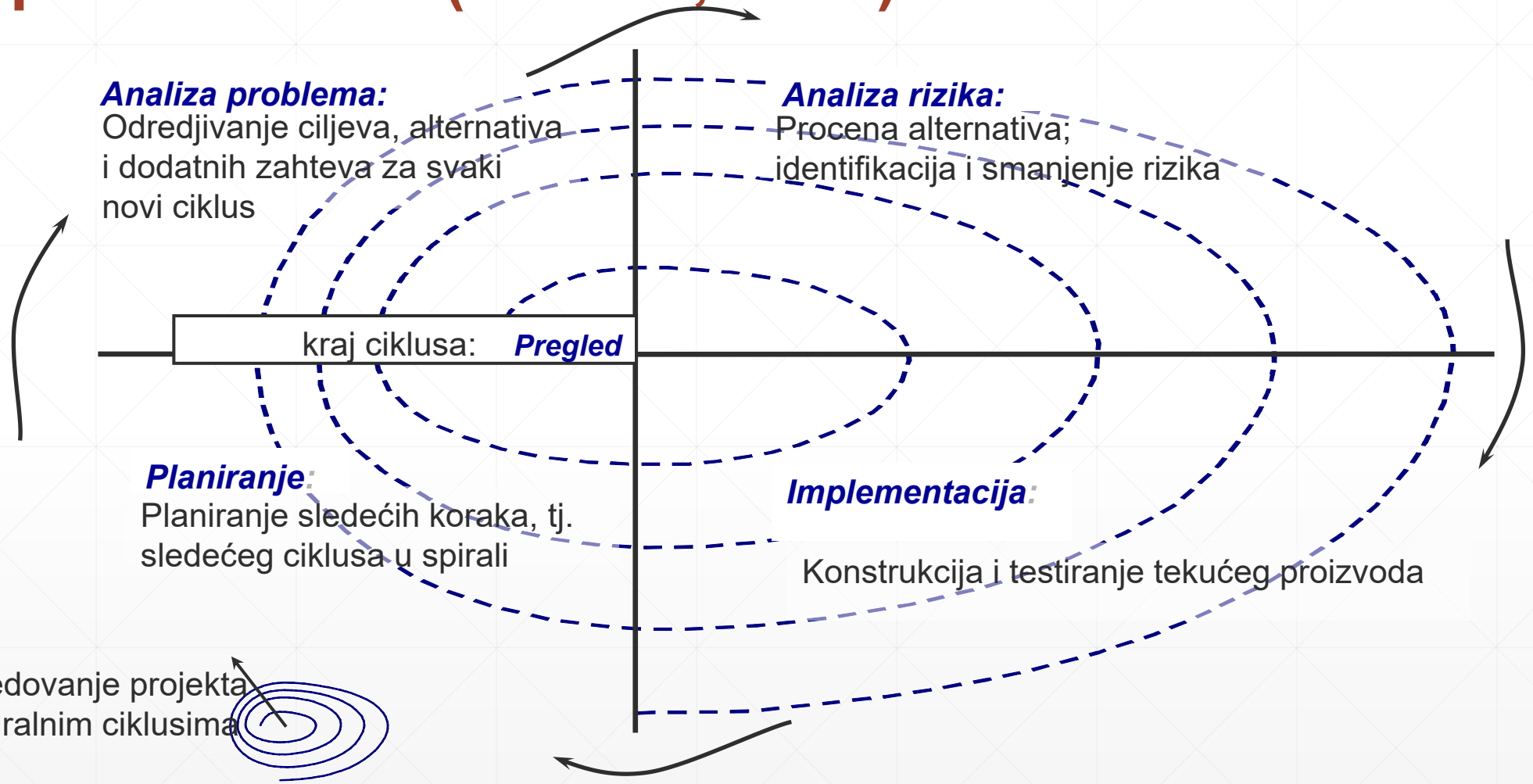
Evolutivni (prototipski) model

+
Potrebna je konstantna povratna veza sa korisnikom da bi se u potpunosti sagledali zahtevi.
Omogućava praćenje tehnoloških promena.

-
Loša struktura softvera, teškoća za održavanje.
Teškoće u praćenju napredovanja razvoja.

Spiralni model

Spiralni model (Boehm, 1988)



Spiralni model

- Dopunjava evolutivni model uzimajući u obzir potrebe upravljanja velikim projektima.
- Razvoj ide po fazama i vođen je ciljem smanjivanja rizika neuspeha.
- Uspešan završetak faze znači smanjenje rizika. Najriskantniji delovi se prvo realizuju.
- U pojedinim fazama se mogu koristiti makete (bacaju se) ili prototipovi (prerastaju u proizvod) za preciziranje specifikacije, procenu rizika i slično.

Šta je DevOps?

- Metodologija u razvoju softvera i IT industriji, koja sugeriše skup praksi i alata u razvoju.
- Integriše i automatizuje rad razvoja softvera (Dev) i IT operacija (Ops) i koristi se kao sredstvo za skraćivanje životnog ciklusa razvoja softvera (nastao iz saradnje razvojnog i operativnog tima).
- DevOps je komplementaran agilnom razvoju softvera. Nije samo model životnog ciklusa, već filozofija novog razmišljanja u organizaciji.
- Glavne odlike DevOps:
  - Programeri i operativni timovi blisko sarađuju, često kao jedan tim – da bi ubrzali inovacije
  - Ažuriranje proizvoda su mala i česta.
  - Desciplina, stalne povratne informacije (*feedback*), poboljšanje procesa, automatizacija ručnog razvoja
- AWS: „DevOps je kombinacija kulturnih filozofija, praksi i alata koji povećavaju sposobnost organizacije da isporuči aplikacije i usluge velikom brzinom, razvijajući i poboljšavajući proizvode bržim tempom od organizacija koje koriste tradicionalni razvoj softvera i procese upravljanja.“

Pitanja?

Hvala na pažnji.