Uvodu u informatiku – Proces razvoja softvera

Danijela Simić
RS

1. februar 2026.

Sadržaj i

Sadržaj ii

Uvod

Životni ciklus razvoja softvera

Razvoj softvera i životni ciklus

- Razvoj softvera nije samo pisanje programa.

Razvoj softvera i životni ciklus

- Razvoj softvera nije samo pisanje programa.
- U širem smislu obuhvata procese pre i posle kodiranja.

Razvoj softvera i životni ciklus

- Razvoj softvera nije samo pisanje programa.
- U širem smislu obuhvata procese pre i posle kodiranja.
- Taj širi okvir zove se životni ciklus razvoja softvera.

Faze životnog ciklusa

- **Planiranje**

Faze životnog ciklusa

- Planiranje
- Realizacija

Faze životnog ciklusa

- Planiranje
- Realizacija
- Eksploatacija

Faze životnog ciklusa

- Planiranje
- Realizacija
- Eksploatacija

Faze životnog ciklusa

- **Planiranje**
- **Realizacija**
- **Eksploatacija**

Koja faza je najbliža naručiocu, a koja krajnjim korisnicima?

Životni ciklus razvoja softvera

Planiranje

Planiranje

- Prikupljanje i analiza zahteva od naručioca.

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.
- Kreiranje precizne specifikacije problema i dizajna rešenja.

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.
- Kreiranje precizne specifikacije problema i dizajna rešenja.

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.
- Kreiranje precizne specifikacije problema i dizajna rešenja.

- Analiza i specifikovanje problema

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.
- Kreiranje precizne specifikacije problema i dizajna rešenja.

  - Analiza i specifikovanje problema
  - Modelovanje rešenja

Planiranje

- Prikupljanje i analiza zahteva od naručioca.
- Razrešavanje nepotpunih, višesmislenih ili kontradiktornih zahteva.
- Kreiranje precizne specifikacije problema i dizajna rešenja.

  - Analiza i specifikovanje problema
  - Modelovanje rešenja
  - Dizajn softverskog rešenja

Životni ciklus razvoja softvera

Realizacija

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.
- Priprema dokumentacije za naručioca.

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.
- Priprema dokumentacije za naručioca.

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.
- Priprema dokumentacije za naručioca.

  - Implementiranje (kodiranje)

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.
- Priprema dokumentacije za naručioca.

- Implementiranje (kodiranje)
- Evaluacija (analiza ispravnosti i analiza efikasnosti)

Realizacija

- Implementiranje dizajniranog rešenja u konkretnom jeziku.
- Analiza efikasnosti i ispravnosti: pouzdanost i upotrebljivost.
- Priprema dokumentacije za naručioca.

  - Implementiranje (kodiranje)
  - Evaluacija (analiza ispravnosti i analiza efikasnosti)
  - Izrada dokumentacije (korisnička i tehnička)

Životni ciklus razvoja softvera

Eksploatacija

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

U održavanje se obično uloži više od tri četvrtine ukupnog rada.

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

U održavanje se obično uloži više od tri četvrtine ukupnog rada.

- Obuka i tehnička podrška

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

U održavanje se obično uloži više od tri četvrtine ukupnog rada.

- Obuka i tehnička podrška
- Puštanje u rad

Eksploatacija

- Počinje nakon adekvatne provere ispravnosti i odobrenja za upotrebu.
- Puštanje u rad: instaliranje, podešavanja, testiranje u realnom okruženju.
- Obuka korisnika i održavanje: ispravke grešaka i manje dopune.

U održavanje se obično uloži više od tri četvrtine ukupnog rada.

- Obuka i tehnička podrška
- Puštanje u rad
- Održavanje

Standardi i kontrola kvaliteta

Standardi životnog ciklusa

- Postoje međunarodni standardi koji opisuju životni ciklus softvera: **ISO/IEC 12207** i **ISO/IEC 15504**.

Standardi životnog ciklusa

- Postoje međunarodni standardi koji opisuju životni ciklus softvera: **ISO/IEC 12207** i **ISO/IEC 15504**.
- Kroz precizno opisane postupke: izbor, implementacija, nadgledanje razvoja.

Standardi životnog ciklusa

- Postoje međunarodni standardi koji opisuju životni ciklus softvera: **ISO/IEC 12207** i **ISO/IEC 15504**.
- Kroz precizno opisane postupke: izbor, implementacija, nadgledanje razvoja.
- Kvalitet se često ocenjuje prema nivou usklađenosti sa standardima.

Kontrola kvaliteta (SQA)

- Pokriva kompletan proces razvoja i sve faze i podfaze.

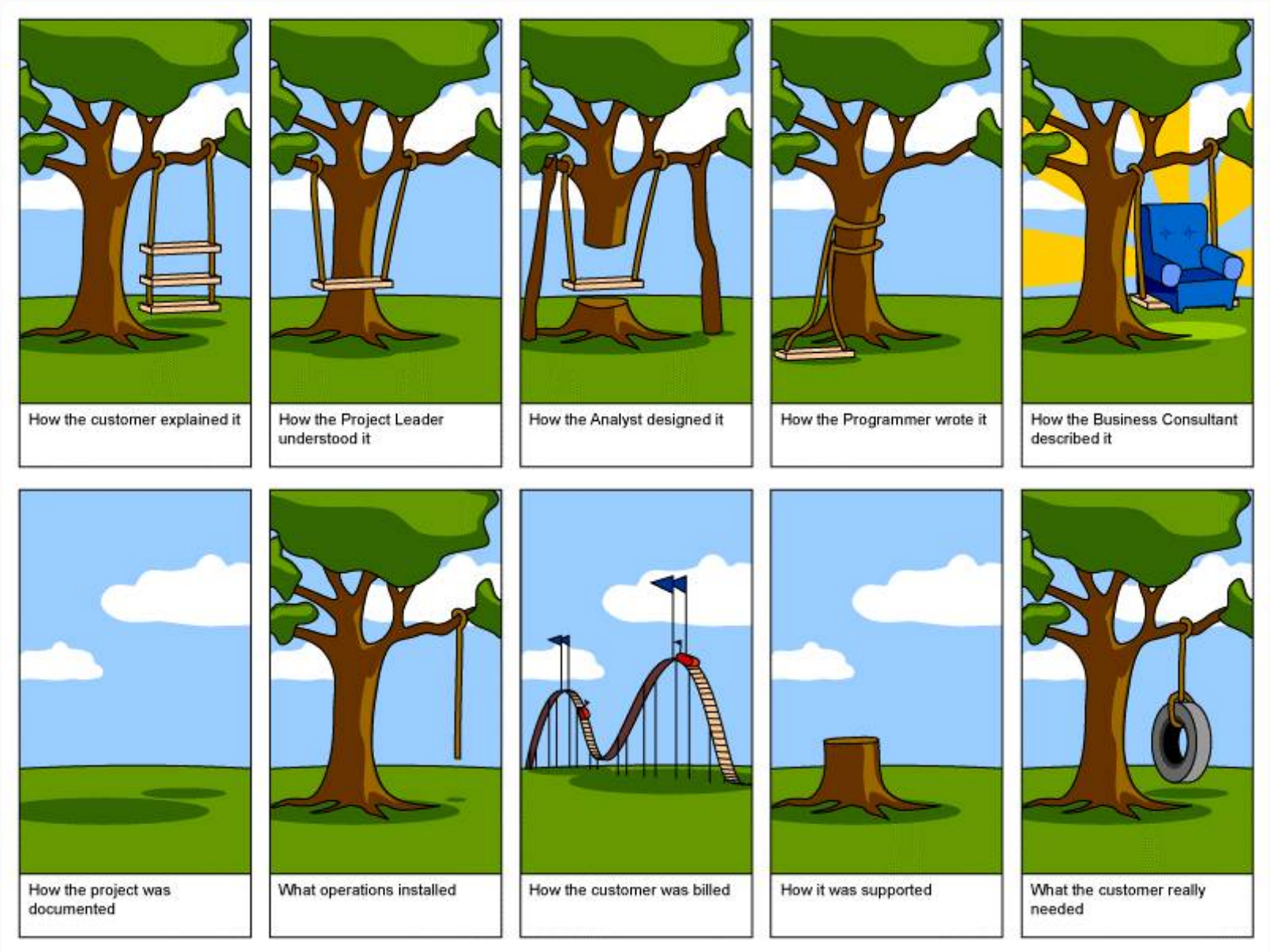
Kontrola kvaliteta (SQA)

- Pokriva kompletan proces razvoja i sve faze i podfaze.
- Treba da osigura nezavisnu potvrdu da su proizvodi, aktivnosti i procesi u skladu sa planovima i standardima.

Kontrola kvaliteta (SQA)

- Pokriva kompletan proces razvoja i sve faze i podfaze.
- Treba da osigura nezavisnu potvrdu da su proizvodi, aktivnosti i procesi u skladu sa planovima i standardima.
- Proces kontrole kvaliteta je takođe opisan standardom **ISO/IEC 15504**.

Karikatura: faze i problemi



Planiranje

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.
- **Modelovanje i dizajn**: razrada projekta definisanog zahtevima.

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.
- **Modelovanje i dizajn**: razrada projekta definisanog zahtevima.
- Koriste se dijagramske tehnike i alati za dijagrame: **CASE alati**.

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.
- **Modelovanje i dizajn**: razrada projekta definisanog zahtevima.
- Koriste se dijagramske tehnike i alati za dijagrame: **CASE alati**.

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.
- **Modelovanje i dizajn**: razrada projekta definisanog zahtevima.
- Koriste se dijagramske tehnike i alati za dijagrame: **CASE alati**.

- Planiranjem strateški rukovodi **arhitekta sistema** (EA).

Planiranje: ko šta radi

- **Poslovna analiza**: precizna postavka i specifikovanje zahteva.
- **Modelovanje i dizajn**: razrada projekta definisanog zahtevima.
- Koriste se dijagramske tehnike i alati za dijagrame: **CASE alati**.

- Planiranjem strateški rukovodi **arhitekta sistema** (EA).
- Zadatak EA: opšti, apstraktan plan procesa koji treba softverski podržati.

Planiranje: uloge i izlazi

- **BA** (poslovni analitičar): zahtevi $\rightarrow$ specifikacija problema.

Planiranje: uloge i izlazi

- **BA** (poslovni analitičar): zahtevi $\rightarrow$ specifikacija problema.
- **SA** (arhitekta rešenja): specifikacija $\rightarrow$ modeli rešenja.

Planiranje: uloge i izlazi

- **BA** (poslovni analitičar): zahtevi $\rightarrow$ specifikacija problema.
- **SA** (arhitekta rešenja): specifikacija $\rightarrow$ modeli rešenja.
- **EA** (arhitekta sistema): „velika slika“ procesa u sistemu.

Planiranje: uloge i izlazi

- **BA** (poslovni analitičar): zahtevi $\rightarrow$ specifikacija problema.
- **SA** (arhitekta rešenja): specifikacija $\rightarrow$ modeli rešenja.
- **EA** (arhitekta sistema): „velika slika“ procesa u sistemu.

Planiranje: uloge i izlazi

- **BA** (poslovni analitičar): zahtevi $\rightarrow$ specifikacija problema.
- **SA** (arhitekta rešenja): specifikacija $\rightarrow$ modeli rešenja.
- **EA** (arhitekta sistema): „velika slika“ procesa u sistemu.

Planiranje

Analiza i specifikovanje problema

Analiza i specifikovanje: poslovni analitičar (BA)

- Analizu obično sprovodi **poslovni analitičar (BA)**.

Analiza i specifikovanje: poslovni analitičar (BA)

- Analizu obično sprovodi **poslovni analitičar (BA)**.
- BA ne mora biti informatičar, ali mora poznavati relevantne procese.

Analiza i specifikovanje: poslovni analitičar (BA)

- Analizu obično sprovodi **poslovni analitičar (BA)**.
- BA ne mora biti informatičar, ali mora poznavati relevantne procese.
- U praksi: intenzivna komunikacija sa naručiocima / korisnicima / predstavnicima.

Analiza i specifikovanje: poslovni analitičar (BA)

- Analizu obično sprovodi **poslovni analitičar (BA)**.
- BA ne mora biti informatičar, ali mora poznavati relevantne procese.
- U praksi: intenzivna komunikacija sa naručiocima / korisnicima / predstavnicima.

Analiza i specifikovanje: poslovni analitičar (BA)

- Analizu obično sprovodi **poslovni analitičar (BA)**.
- BA ne mora biti informatičar, ali mora poznavati relevantne procese.
- U praksi: intenzivna komunikacija sa naručiocima / korisnicima / predstavnicima.

Šta BA radi u komunikaciji sa naručiocima

- Analiza postojećih rešenja i mogućnosti unapređenja novim softverom.

Šta BA radi u komunikaciji sa naručiocima

- Analiza postojećih rešenja i mogućnosti unapređenja novim softverom.
- Zahtevi su često neprecizni ili kontradiktorni.

Šta BA radi u komunikaciji sa naručiocima

- Analiza postojećih rešenja i mogućnosti unapređenja novim softverom.
- Zahtevi su često neprecizni ili kontradiktorni.
- Zadatak BA: da zahteve precizira i uobliči u saradnji sa naručiocima.

Šta BA radi u komunikaciji sa naručiocima

- Analiza postojećih rešenja i mogućnosti unapređenja novim softverom.
- Zahtevi su često neprecizni ili kontradiktorni.
- Zadatak BA: da zahteve precizira i uobliči u saradnji sa naručiocima.

Šta BA radi u komunikaciji sa naručiocima

- Analiza postojećih rešenja i mogućnosti unapređenja novim softverom.
- Zahtevi su često neprecizni ili kontradiktorni.
- Zadatak BA: da zahteve precizira i uobliči u saradnji sa naručiocima.

- Rezultat: **opšta specifikacija problema** (procesi + funkcionalnosti + potrebna efikasnost i druga svojstva).

Poslovna analiza: procene u planiranju

- Procena **obima posla** (npr. u čovek-mesecima) i jasna granica: šta ulazi/šta ne.

Poslovna analiza: procene u planiranju

- Procena **obima posla** (npr. u čovek-mesecima) i jasna granica: šta ulazi/šta ne.
- Identifikacija **rizika** i definisanje reakcija ako stvari krenu drugačije.

Poslovna analiza: procene u planiranju

- Procena **obima posla** (npr. u čovek-mesecima) i jasna granica: šta ulazi/šta ne.
- Identifikacija **rizika** i definisanje reakcija ako stvari krenu drugačije.
- Procena **resursa** (ljudskih i materijalnih).

Poslovna analiza: procene u planiranju

- Procena **obima posla** (npr. u čovek-mesecima) i jasna granica: šta ulazi/šta ne.
- Identifikacija **rizika** i definisanje reakcija ako stvari krenu drugačije.
- Procena **resursa** (ljudskih i materijalnih).
- Procena **cene** realizacije (i delova projekta).

Poslovna analiza: procene u planiranju

- Procena **obima posla** (npr. u čovek-mesecima) i jasna granica: šta ulazi/šta ne.
- Identifikacija **rizika** i definisanje reakcija ako stvari krenu drugačije.
- Procena **resursa** (ljudskih i materijalnih).
- Procena **cene** realizacije (i delova projekta).
- **Plan rada po fazama** koji se poštuje.

Planiranje

Modelovanje rešenja

Modelovanje rešenja: arhitekta rešenja (SA)

- Modelovanje obično sprovodi **arhitekta rešenja (SA)**.

Modelovanje rešenja: arhitekta rešenja (SA)

- Modelovanje obično sprovodi **arhitekta rešenja (SA)**.
- SA razume specifikaciju zahteva i izrađuje modele problema.

Modelovanje rešenja: arhitekta rešenja (SA)

- Modelovanje obično sprovodi **arhitekta rešenja (SA)**.
- SA razume specifikaciju zahteva i izrađuje modele problema.
- SA bira adekvatna softverska rešenja: jezik, baza, biblioteke, strukture podataka, algoritamska rešenja, itd.

Šta može biti model rešenja

- Matematički model (optimizacioni model, sistemski graf, formalna specifikacija).

Šta može biti model rešenja

- Matematički model (optimizacioni model, sistemski graf, formalna specifikacija).
- Simulacija, heuristički opis, pseudokod, vizuelna skica.

Šta može biti model rešenja

- Matematički model (optimizacioni model, sistemski graf, formalna specifikacija).
- Simulacija, heuristički opis, pseudokod, vizuelna skica.
- U nekim domenima: domen-specifični jezici.

Šta može biti model rešenja

- Matematički model (optimizacioni model, sistemski graf, formalna specifikacija).
- Simulacija, heuristički opis, pseudokod, vizuelna skica.
- U nekim domenima: domen-specifični jezici.

Šta može biti model rešenja

- Matematički model (optimizacioni model, sistemski graf, formalna specifikacija).
- Simulacija, heuristički opis, pseudokod, vizuelna skica.
- U nekim domenima: domen-specifični jezici.

- Model treba da bude dovoljno precizan za dizajn, ali i dovoljno apstraktan za fleksibilnost tehnologija.

Cilj modelovanja i odnos prema dizajnu

- Cilj: razlaganje složenog problema na jasne logičke celine.

Cilj modelovanja i odnos prema dizajnu

- Cilj: razlaganje složenog problema na jasne logičke celine.
- Modelovanje pomaže komunikaciji u timu i proceni troškova, složenosti i rizika.

Cilj modelovanja i odnos prema dizajnu

- Cilj: razlaganje složenog problema na jasne logičke celine.
- Modelovanje pomaže komunikaciji u timu i proceni troškova, složenosti i rizika.
- Modelovanje: razumevanje ključnih komponenti; **dizajn**: kako će se tehnički realizovati na konkretnoj platformi.

Primer: elektronska narudžbina hrane

- Model: relacija između **korisnika**, **restorana** i **narudžbina**.

Primer: elektronska narudžbina hrane

- Model: relacija između **korisnika**, **restorana** i **narudžbina**.
- Tok narudžbine kao automat sa stanjima: *kreirana* $\rightarrow$ *potvrđena* $\rightarrow$ *u pripremi* $\rightarrow$ *u dostavi* $\rightarrow$ *isporučena*.

Planiranje

Dizajn softverskog rešenja

Dizajn: arhitekta softvera i arhitektura

- U dizajnu, **arhitekta softvera** precizira rešenje.

Dizajn: arhitekta softvera i arhitektura

- U dizajnu, **arhitekta softvera** precizira rešenje.
- Opisuje **arhitekturu softvera**.

Dizajn: arhitekta softvera i arhitektura

- U dizajnu, **arhitekta softvera** precizira rešenje.
- Opisuje **arhitekturu softvera**.

Dizajn: arhitekta softvera i arhitektura

- U dizajnu, **arhitekta softvera** precizira rešenje.
- Opisuje **arhitekturu softvera**.

> Celokupna struktura softvera i način na koji ta struktura obezbeđuje integritet sistema i željeni ishod projekta (ispravan softver, performanse, rokovi, troškovi). Uključuje komponente, njihove odnose i interakcije, kao i principe/smernice za dizajn i evoluciju.

Šta dizajn radi (u odnosu na prethodne faze)

- Razrađuje pojmove ranije opisane nezavisno od tehnologija.

Šta dizajn radi (u odnosu na prethodne faze)

- Razrađuje pojmove ranije opisane nezavisno od tehnologija.
- Daje opšti plan kako sistem da bude izgrađen na konkretnoj hardverskoj i softverskoj platformi.

Šta dizajn radi (u odnosu na prethodne faze)

- Razrađuje pojmove ranije opisane nezavisno od tehnologija.
- Daje opšti plan kako sistem da bude izgrađen na konkretnoj hardverskoj i softverskoj platformi.
- Često koristi unapred ponuđene obrasce: **design patterns**.

Kako se dizajn zapisuje: jednostavni vs kompleksni sistemi

- Jednostavniji slučajevi: neformalni tekst ili dijagram protoka podataka.

Kako se dizajn zapisuje: jednostavni vs kompleksni sistemi

- Jednostavniji slučajevi: neformalni tekst ili dijagram protoka podataka.
- Dijagram protoka podataka prikazuje tok podataka i funkcionalne transformacije, ali ne opisuje implementaciju.

Kako se dizajn zapisuje: jednostavni vs kompleksni sistemi

- Jednostavniji slučajevi: neformalni tekst ili dijagram protoka podataka.
- Dijagram protoka podataka prikazuje tok podataka i funkcionalne transformacije, ali ne opisuje implementaciju.
- Kompleksniji slučajevi: standardizovane grafičke notacije (grafički jezici), npr. **UML**.

Tema 1: Apstrahovanje (abstraction)

- Proces generalizacije: odbacivanje nebitnih informacija.

Tema 1: Apstrahovanje (abstraction)

- Proces generalizacije: odbacivanje nebitnih informacija.
- Zadržavaju se samo informacije bitne za softver.

Tema 1: Apstrahovanje (abstraction)

- Proces generalizacije: odbacivanje nebitnih informacija.
- Zadržavaju se samo informacije bitne za softver.
- Primer: boja očiju studenta nije relevantna u IS fakulteta $\rightarrow$ odbacuje se.

Tema 2: Profinjavanje (refinement)

- Razvoj odozgo-naniže: nerazrađeni koraci se postepeno preciziraju.

Tema 2: Profinjavanje (refinement)

- Razvoj odozgo-naniže: nerazrađeni koraci se postepeno preciziraju.
- Svaki zadatak se razlaže na sitnije zadatke.

Tema 2: Profinjavanje (refinement)

- Razvoj odozgo-naniže: nerazrađeni koraci se postepeno preciziraju.
- Svaki zadatak se razlaže na sitnije zadatke.
- Krajnji rezultat: precizan opis u obliku programskog koda.

Profinjavanje: primer razlaganja

- Jedan zadatak razlaže se na podzadatke / pomoćne funkcije.

Profinjavanje: primer razlaganja

- Jedan zadatak razlaže se na podzadatke / pomoćne funkcije.

Profinjavanje: primer razlaganja

- Jedan zadatak razlaže se na podzadatke / pomoćne funkcije.

```
obradi_podatke_iz_datoteke()
  -> otvori_datoteku()
  -> procitaj_podatke()
  -> obradi_podatke()
  -> zatvori_datoteku()
```

- Apstrahovanje i profinjavanje su suprotni procesi.

Tema 3: Dekompozicija (decomposition)

- Cilj: razlaganje sistema na komponente koje je lakše razumeti, realizovati i održavati.

Tema 3: Dekompozicija (decomposition)

- Cilj: razlaganje sistema na komponente koje je lakše razumeti, realizovati i održavati.
- Proizvod dekompozicije nije implementacija, već **opis arhitekture**.

Tema 3: Dekompozicija (decomposition)

- Cilj: razlaganje sistema na komponente koje je lakše razumeti, realizovati i održavati.
- Proizvod dekompozicije nije implementacija, već **opis arhitekture**.
- Pristupi zavise od paradigme (OO, funkcionalna, ...).

Tema 4: Modularnost (modularity)

- Softver se deli na komponente: **moduli**.

Tema 4: Modularnost (modularity)

- Softver se deli na komponente: **moduli**.
- Svaki modul ima precizno definisanu funkcionalnost.

Tema 4: Modularnost (modularity)

- Softver se deli na komponente: **moduli**.
- Svaki modul ima precizno definisanu funkcionalnost.
- Poželjno: malo međuzavisnosti, da moduli mogu da se koriste i u drugim programima.

Planiranje

Objedinjeni jezik za modelovanje: UML dijagrami

UML: objedinjeni jezik za modelovanje

UML (Unified Modeling Language) je vizuelna tehnika i standardizovani jezik za modelovanje softvera: opisuje zahteve, akcije i fizičku distribuciju rešenja.

UML: objedinjeni jezik za modelovanje

UML (Unified Modeling Language) je vizuelna tehnika i standardizovani jezik za modelovanje softvera: opisuje zahteve, akcije i fizičku distribuciju rešenja.

- UML je pre svega **grafički jezik**.

UML: objedinjeni jezik za modelovanje

UML (Unified Modeling Language) je vizuelna tehnika i standardizovani jezik za modelovanje softvera: opisuje zahteve, akcije i fizičku distribuciju rešenja.

- UML je pre svega **grafički jezik**.
- Može se koristiti i u tekstualnom obliku (samo neki elementi po standardu).

UML: objedinjeni jezik za modelovanje

UML (Unified Modeling Language) je vizuelna tehnika i standardizovani jezik za modelovanje softvera: opisuje zahteve, akcije i fizičku distribuciju rešenja.

- UML je pre svega **grafički jezik**.
- Može se koristiti i u tekstualnom obliku (samo neki elementi po standardu).
- Postoji mnogo UML dijagrama; ovde prikazujemo samo neke.

Dve velike grupe UML dijagrama

- **Strukturni dijagrami**: prikazuju strukturu sistema i odnose između komponenti (klase, objekti, komponente, raspored, ...).

Dve velike grupe UML dijagrama

- **Strukturni dijagrami**: prikazuju strukturu sistema i odnose između komponenti (klase, objekti, komponente, raspored, ...).
- **Dijagrami ponašanja**: prikazuju kako se sistem ponaša tokom izvršavanja (aktivnosti, stanja, sekvence, upotrebe, ...).

Strukturni UML dijagrami

Strukturni dijagrami su statički prikaz: opisuju elemente (klase, pakete, komponente, uređaje) i njihove odnose; ne sadrže vremenski tok ni dinamičke promene stanja. Koriste se od analize do implementacije i isporuke.

Najčešći strukturni dijagram: dijagram klasa

- Prikazuje **klase**, njihove **atribute** i **metode**.

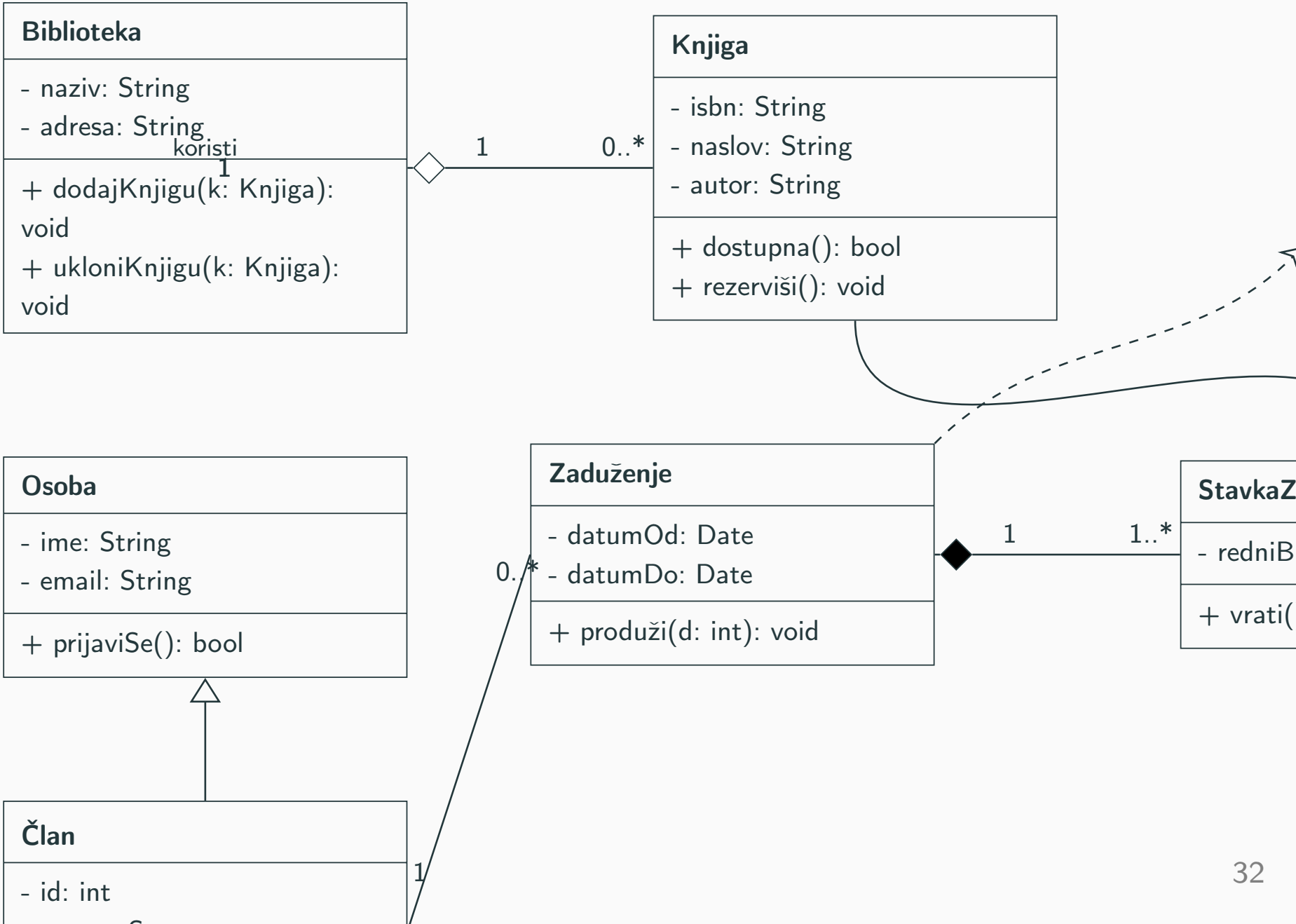
Najčešći strukturni dijagram: dijagram klasa

- Prikazuje **klase**, njihove **atribute** i **metode**.
- Prikazuje odnose: **nasleđivanje**, **asocijacija**, **kompozicija**, **agregacija**, **zavisnost**.

Najčešći strukturni dijagram: dijagram klasa

- Prikazuje **klase**, njihove **atribute** i **metode**.
- Prikazuje odnose: **nasleđivanje**, **asocijacija**, **kompozicija**, **agregacija**, **zavisnost**.
- Primer: sistem biblioteke (`Knjiga`, `Član`, `Zaduženje`).

Primer: UML dijagram klasa (biblioteka)



Još strukturnih dijagrama

- Dijagrami objekata

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti
- Dijagrami rasporeda

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti
- Dijagrami rasporeda
- Dijagram slučajeva upotrebe

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti
- Dijagrami rasporeda
- Dijagram slučajeva upotrebe

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti
- Dijagrami rasporeda
- Dijagram slučajeva upotrebe

- **Dijagram slučajeva upotrebe** prikazuje funkcionalnosti sistema kroz interakcije između korisnika (aktera) i osnovnih scenarija korišćenja.

Još strukturnih dijagrama

- Dijagrami objekata
- Dijagrami komponenti
- Dijagrami rasporeda
- Dijagram slučajeva upotrebe

- **Dijagram slučajeva upotrebe** prikazuje funkcionalnosti sistema kroz interakcije između korisnika (aktera) i osnovnih scenarija korišćenja.
- Na višem nivou: manji broj generalizovanih poslovnih scenarija (glavne uloge grupa korisnika).

Dijagram sekvence: šta prikazuje

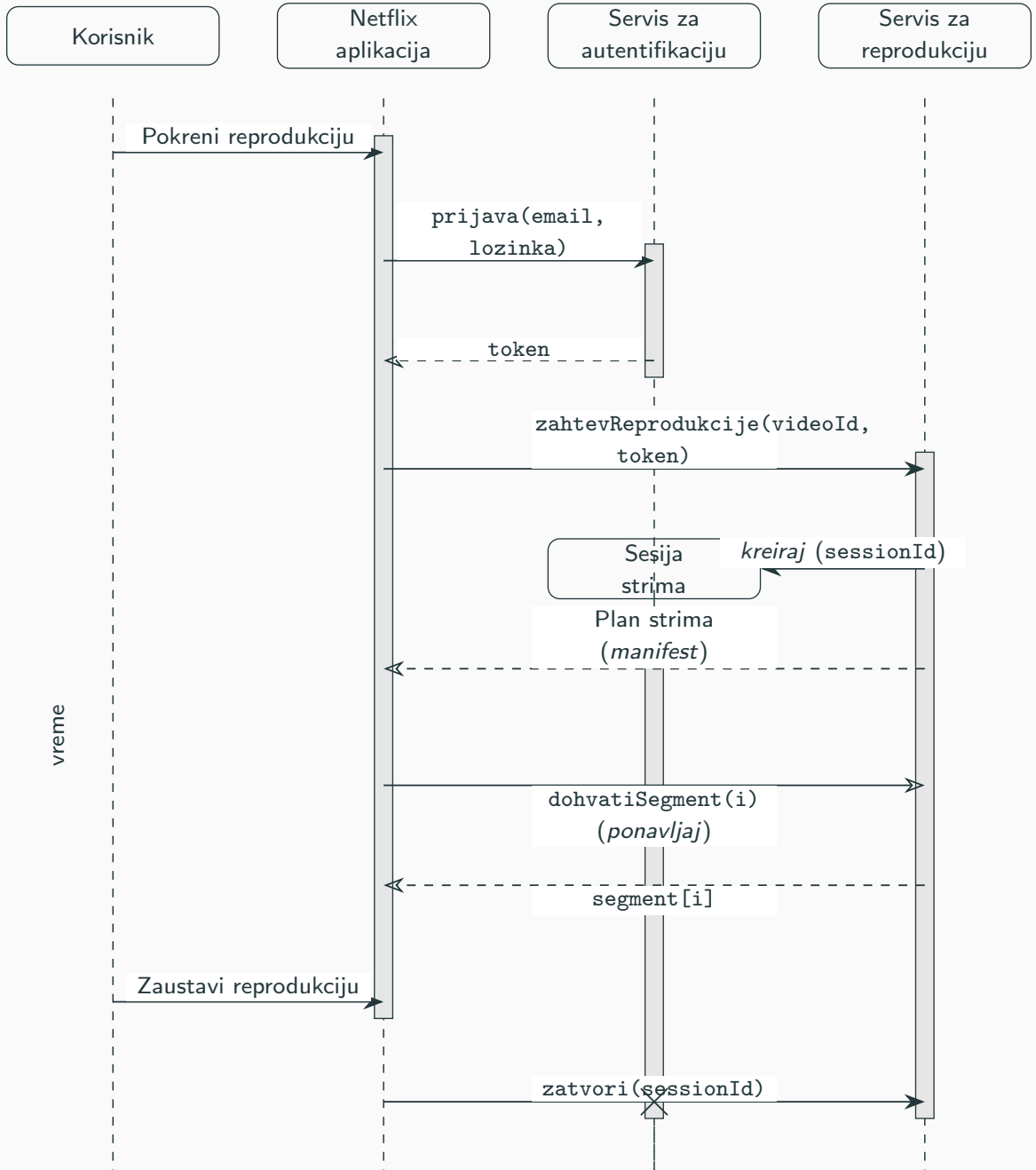
Dijagram sekvence prikazuje ponašanje sistema kroz vremenski redosled događaja: kako objekti komuniciraju kroz vreme tokom izvršavanja aktivnosti.

Dijagram sekvence: šta prikazuje

Dijagram sekvence prikazuje ponašanje sistema kroz vremenski redosled događaja: kako objekti komuniciraju kroz vreme tokom izvršavanja aktivnosti.

- Prikazuje: ko učestvuje, kojim redosledom se odvija, kako se prenose odgovornosti.

Primer: dijagram sekvence



Metodologije razvoja softvera

Metodologije razvoja softvera

- Postoji mnogo metodologija razvoja softvera (u teoriji i praksi).

Metodologije razvoja softvera

- Postoji mnogo metodologija razvoja softvera (u teoriji i praksi).
- U praksi su često pomešane i teško je projekte striktno svrstati.

Metodologije razvoja softvera

- Postoji mnogo metodologija razvoja softvera (u teoriji i praksi).
- U praksi su često pomešane i teško je projekte striktno svrstati.
- U nastavku: nekoliko često korišćenih metodologija i ključne ideje.

Metodologije razvoja softvera

Metodologija vodopada

Metodologija vodopada (waterfall)

Stroga varijanta: faze redom, bez povratka

- zahtevi

Metodologija vodopada (waterfall)

Stroga varijanta: faze redom, bez povratka

- zahtevi
- dizajn

Metodologija vodopada (waterfall)

Stroga varijanta: faze redom, bez povratka

- zahtevi
- dizajn
- implementacija

Metodologija vodopada (waterfall)

Stroga varijanta: faze redom, bez povratka

- zahtevi
- dizajn
- implementacija
- testiranje

Metodologija vodopada (waterfall)

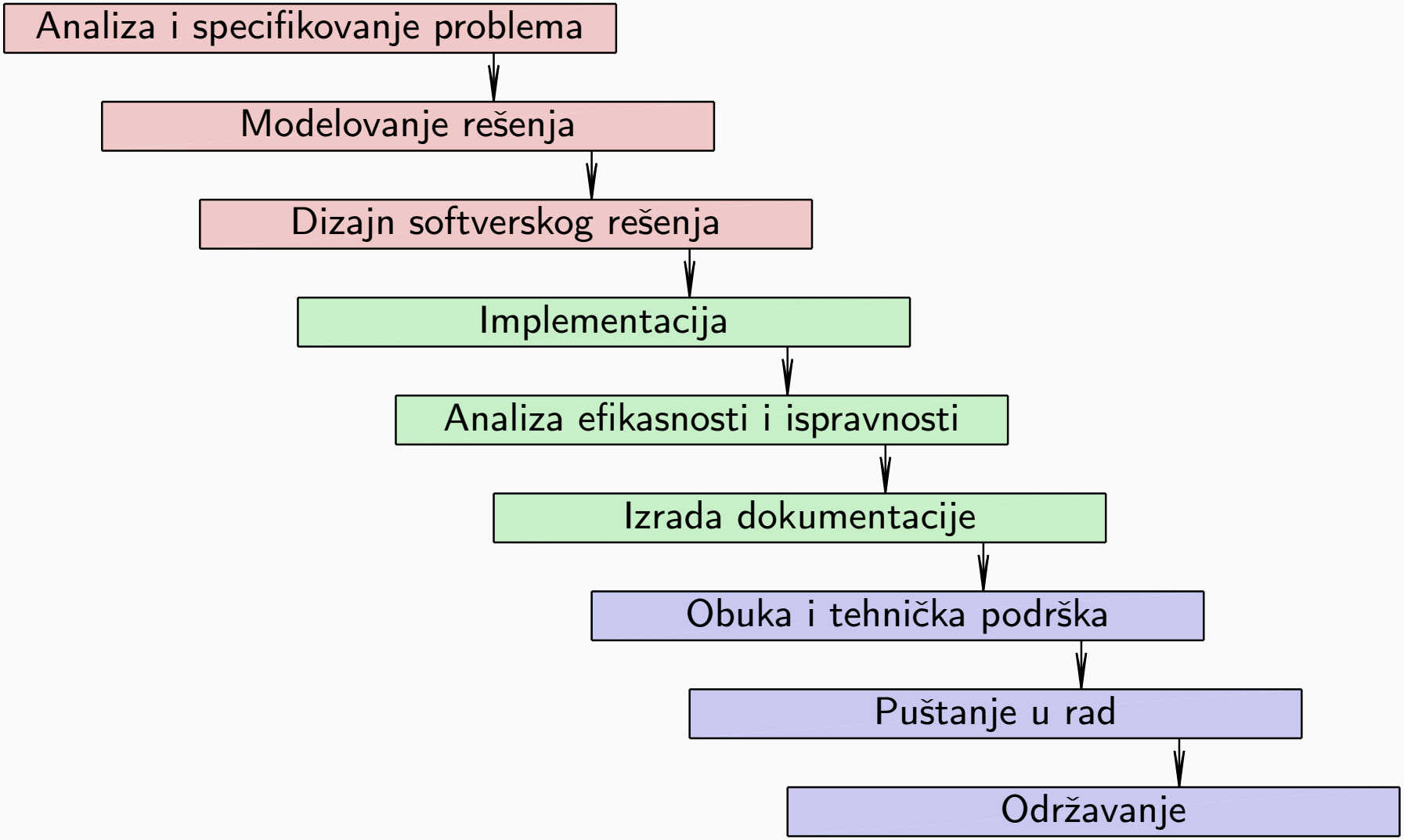
Stroga varijanta: faze redom, bez povratka

- zahtevi
- dizajn
- implementacija
- testiranje
- integracija

Metodologija vodopada (waterfall)

Stroga varijanta: faze redom, bez povratka

- zahtevi
- dizajn
- implementacija
- testiranje
- integracija
- održavanje


Analiza i specifikovanje problema
Modelovanje rešenja
Dizajn softverskog rešenja
Implementacija
Analiza efikasnosti i ispravnosti
Izrada dokumentacije
Obuka i tehnička podrška
Puštanje u rad
Održavanje

- **Kada je primenljiva:** zahtevi su poznati unapred, stabilni, bez rizičnih nepoznanica; arhitektura se može detaljno opisati; ima vremena za etape.

- **Kada je primenljiva:** zahtevi su poznati unapred, stabilni, bez rizičnih nepoznanica; arhitektura se može detaljno opisati; ima vremena za etape.
- **Prednosti:** jasna struktura + neophodna detaljna dokumentacija (često u velikim timovima).

- **Kada je primenljiva:** zahtevi su poznati unapred, stabilni, bez rizičnih nepoznanica; arhitektura se može detaljno opisati; ima vremena za etape.
- **Prednosti:** jasna struktura + neophodna detaljna dokumentacija (često u velikim timovima).
- **Kritike:** krutost (nema menjanja faza), zahtevi se menjaju, korisnici su često uključeni samo na početku i kraju, čekanja između zavisnih zadataka.

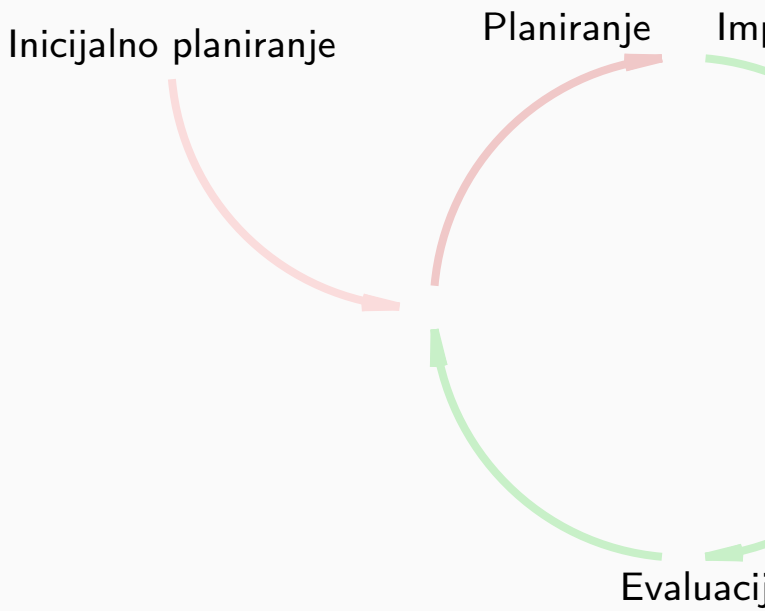
- **Kada je primenljiva:** zahtevi su poznati unapred, stabilni, bez rizičnih nepoznanica; arhitektura se može detaljno opisati; ima vremena za etape.
- **Prednosti:** jasna struktura + neophodna detaljna dokumentacija (često u velikim timovima).
- **Kritike:** krutost (nema menjanja faza), zahtevi se menjaju, korisnici su često uključeni samo na početku i kraju, čekanja između zavisnih zadataka.
- Danas se retko koristi; može biti korisna kod rigidnih sistema (npr. medicinski uređaji, avijacija); postoje varijante (npr. V-metodologija).

Metodologije razvoja softvera

Metodologija iterativnog i inkrementalnog razvoja

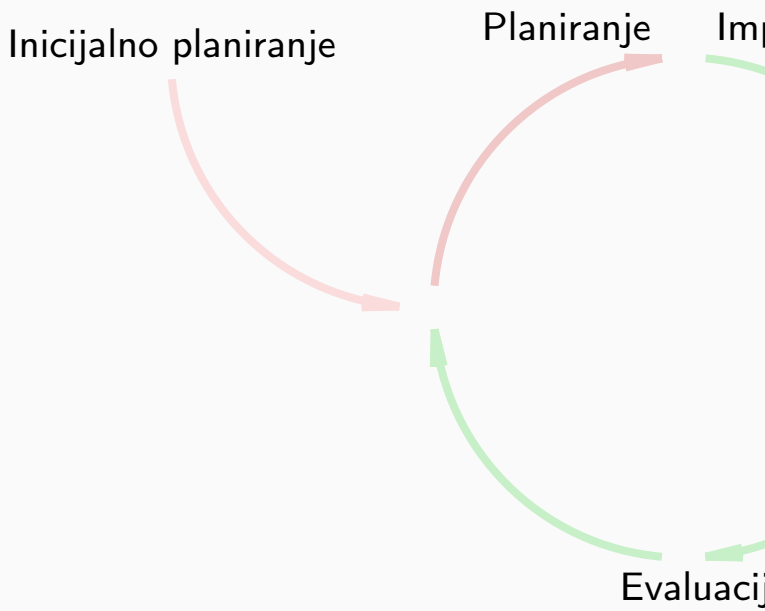
Iterativni i inkrementalni razvoj

- Razvoj se sprovodi u iteracijama.



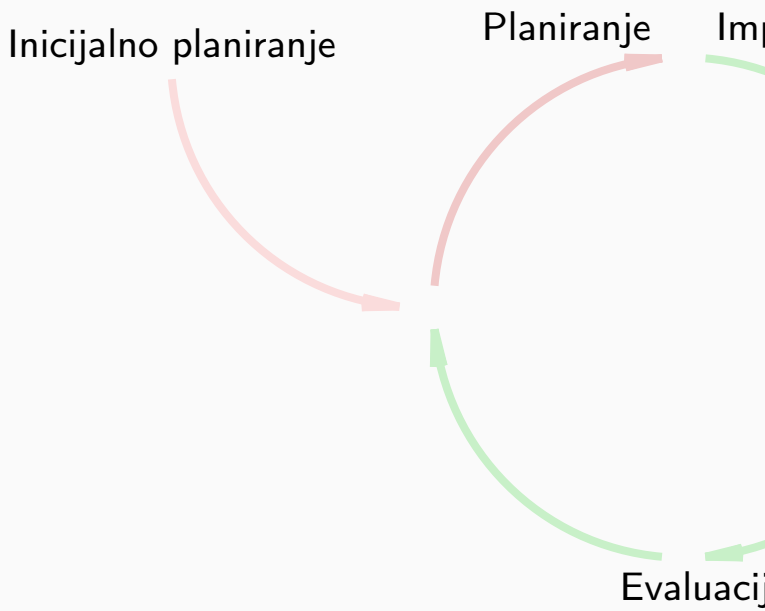
Iterativni i inkrementalni razvoj

- Razvoj se sprovodi u iteracijama.
- Sistem se gradi inkrementalno (dodavanje modula).



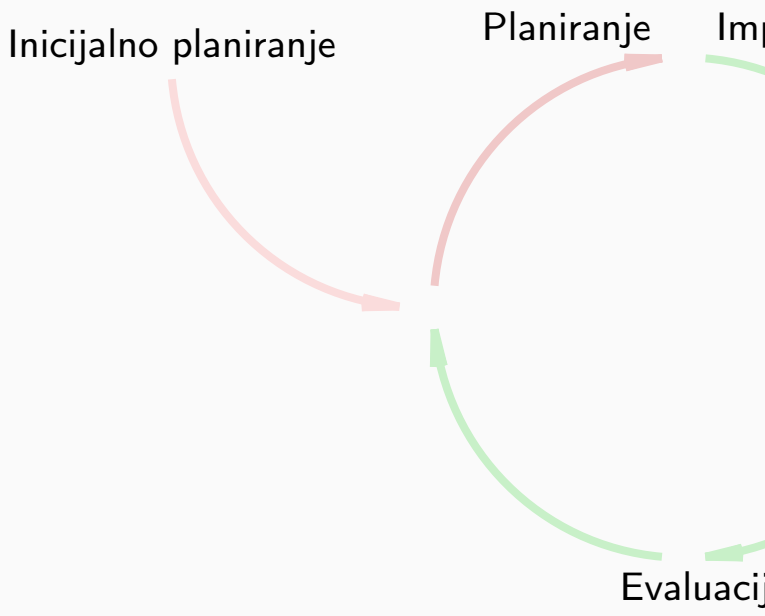
Iterativni i inkrementalni razvoj

- Razvoj se sprovodi u iteracijama.
- Sistem se gradi inkrementalno (dodavanje modula).
- Moduli se mogu modifikovati u budućim iteracijama.



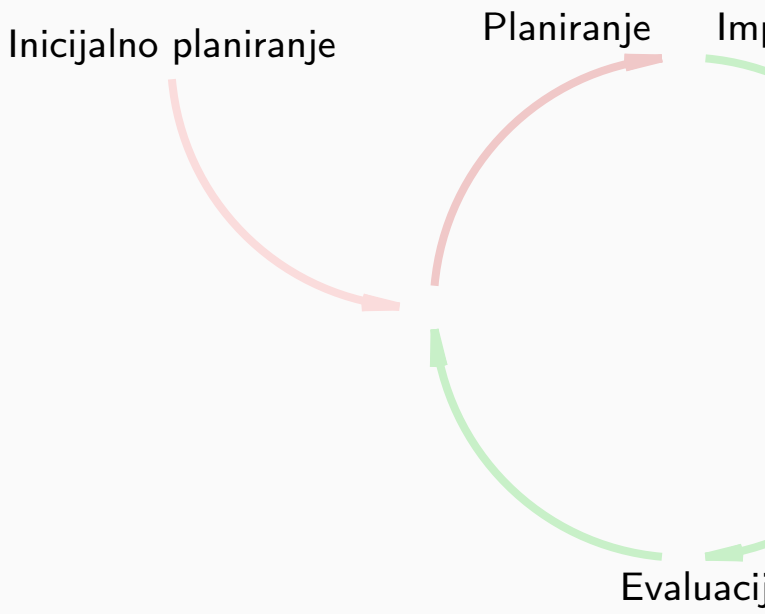
Iterativni i inkrementalni razvoj

- Razvoj se sprovodi u iteracijama.
- Sistem se gradi inkrementalno (dodavanje modula).
- Moduli se mogu modifikovati u budućim iteracijama.
- U jednom trenutku više faza životnog ciklusa može biti u toku.



Iterativni i inkrementalni razvoj

- Razvoj se sprovodi u iteracijama.
- Sistem se gradi inkrementalno (dodavanje modula).
- Moduli se mogu modifikovati u budućim iteracijama.
- U jednom trenutku više faza životnog ciklusa može biti u toku.
- Vraćanje unazad je moguće.



Metodologije razvoja softvera

Metodologija rapidnog razvoja

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.
- Prototipovi služe da se zahtevi definišu / preciziraju / potvrde.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.
- Prototipovi služe da se zahtevi definišu / preciziraju / potvrde.
- Dokumentacija je vrlo ograničena.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.
- Prototipovi služe da se zahtevi definišu / preciziraju / potvrde.
- Dokumentacija je vrlo ograničena.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.
- Prototipovi služe da se zahtevi definišu / preciziraju / potvrde.
- Dokumentacija je vrlo ograničena.

- Mogući problem: niz prototipova bez zadovoljavajuće finalne aplikacije, često zbog fokusiranja na GUI umesto na obradu/podatke.

Rapidni razvoj (RAD)

- Planiranje je svedeno na minimum radi brzih prototipova u iteracijama.
- Planiranje se preklapa sa implementacijom $\rightarrow$ lakše izmene zahteva u hodu.
- Kreće se od preliminarnog modela podataka i algoritama.
- Prototipovi služe da se zahtevi definišu / preciziraju / potvrde.
- Dokumentacija je vrlo ograničena.

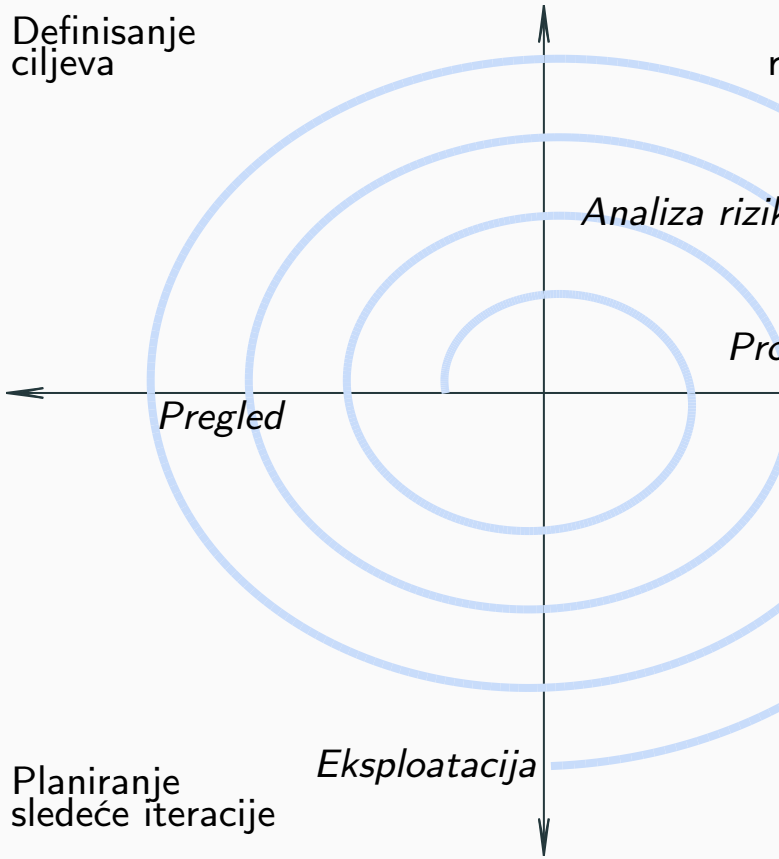
- Mogući problem: niz prototipova bez zadovoljavajuće finalne aplikacije, često zbog fokusiranja na GUI umesto na obradu/podatke.
- Pogodno: sopstvene potrebe ili ograničen broj korisnika.

Metodologije razvoja softvera

Spiralna metodologija

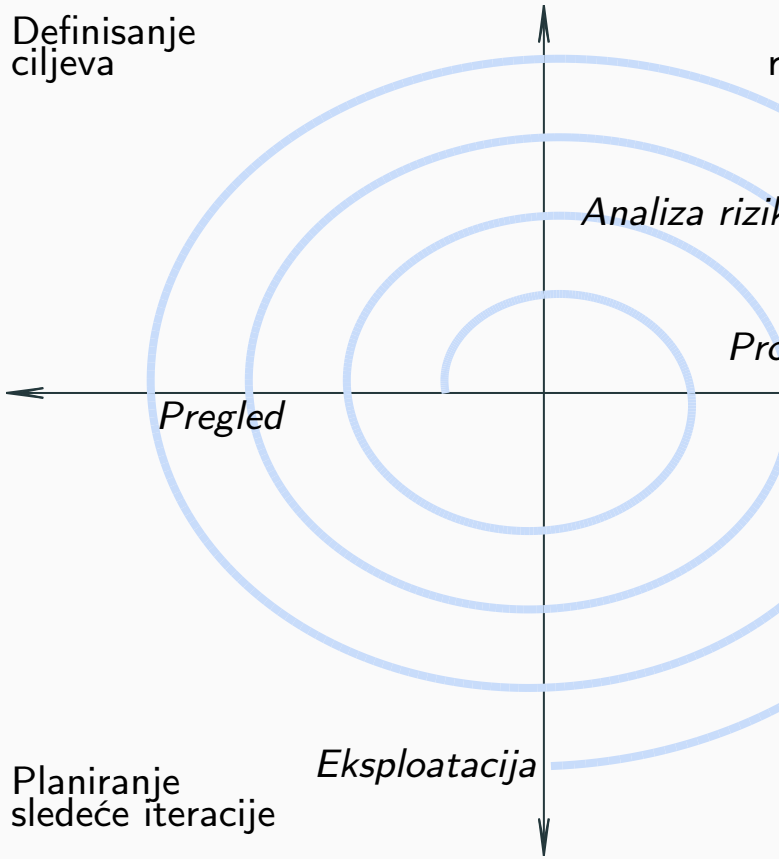
Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.



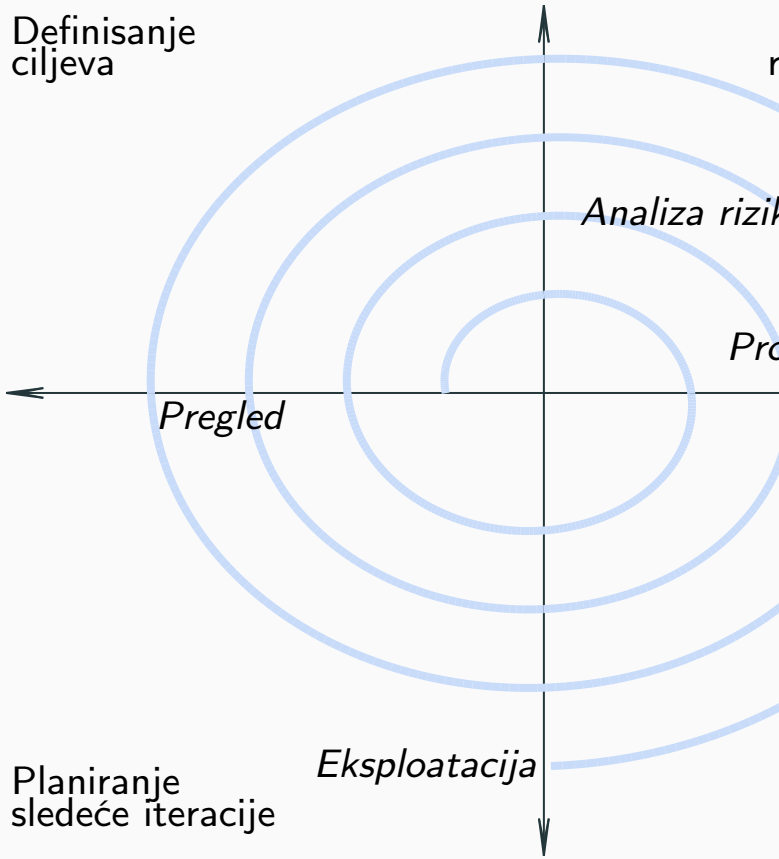
Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.
- Spirala prolazi više puta kroz: planiranje, implementaciju, evaluaciju tekuće verzije, analizu rizika.



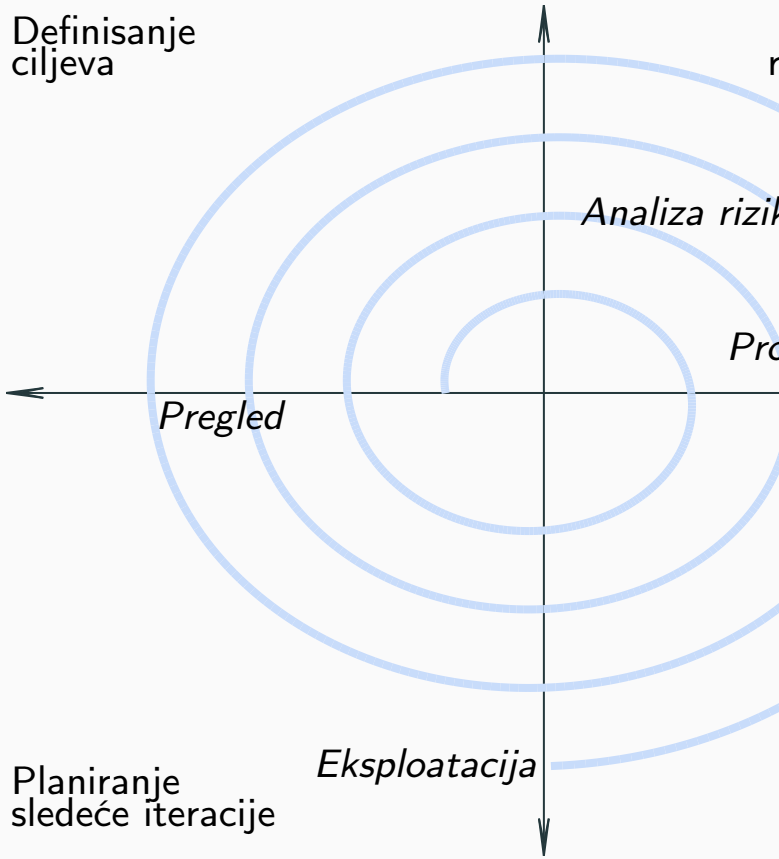
Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.
- Spirala prolazi više puta kroz: planiranje, implementaciju, evaluaciju tekuće verzije, analizu rizika.
- Faze se sprovode jedna za drugom (ne paralelno).



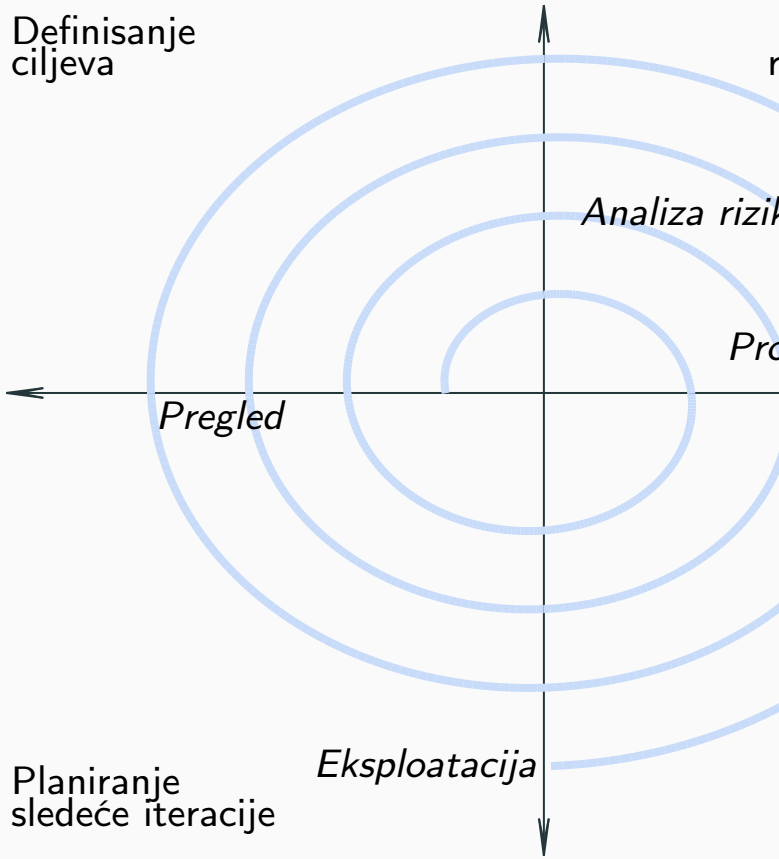
Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.
- Spirala prolazi više puta kroz: planiranje, implementaciju, evaluaciju tekuće verzije, analizu rizika.
- Faze se sprovode jedna za drugom (ne paralelno).
- Prvi prototip je aproksimacija finalnog proizvoda.



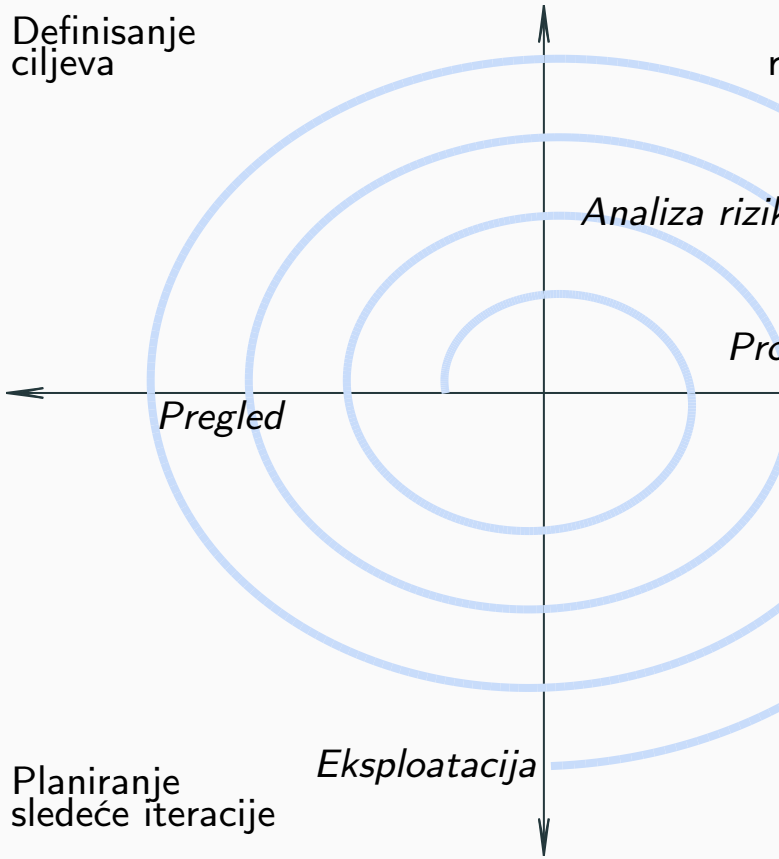
Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.
- Spirala prolazi više puta kroz: planiranje, implementaciju, evaluaciju tekuće verzije, analizu rizika.
- Faze se sprovode jedna za drugom (ne paralelno).
- Prvi prototip je aproksimacija finalnog proizvoda.
- Na kraju iteracije: evaluacija, profinjavanje specifikacije i analiza rizika (bagovi, cena, tempo, efikasnost, bezbednost, ...).



Spiralna metodologija

- Kombinuje analizu rizika sa vodopadom i iterativnim razvojem.
- Spirala prolazi više puta kroz: planiranje, implementaciju, evaluaciju tekuće verzije, analizu rizika.
- Faze se sprovode jedna za drugom (ne paralelno).
- Prvi prototip je aproksimacija finalnog proizvoda.
- Na kraju iteracije: evaluacija, profinjavanje specifikacije i analiza rizika (bagovi, cena, tempo, efikasnost, bezbednost, ...).



Metodologije razvoja softvera

Agilna metodologija razvoja

Agilna metodologija razvoja

Fokus na zadovoljstvo korisnika kroz ranu i inkrementalnu isporuku: iteracije sa minimalnim dodavanjem funkcionalnosti u kratkim intervalima (obično 1–4 nedelje).

Agilna metodologija razvoja

Fokus na zadovoljstvo korisnika kroz ranu i inkrementalnu isporuku: iteracije sa minimalnim dodavanjem funkcionalnosti u kratkim intervalima (obično 1–4 nedelje).

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.
- Nije primenljivo svuda: svet se menja, specifikacije često ne mogu unapred potpuno; agilnost pomaže adaptaciji i može smanjiti troškove promena.

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.
- Nije primenljivo svuda: svet se menja, specifikacije često ne mogu unapred potpuno; agilnost pomaže adaptaciji i može smanjiti troškove promena.
- **Agilni Manifest** (2001): 12 principa (npr. funkcionalan softver pre obimne dokumentacije, odgovor na promene pre praćenja plana, ...).

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.
- Nije primenljivo svuda: svet se menja, specifikacije često ne mogu unapred potpuno; agilnost pomaže adaptaciji i može smanjiti troškove promena.
- **Agilni Manifest** (2001): 12 principa (npr. funkcionalan softver pre obimne dokumentacije, odgovor na promene pre praćenja plana, ...).
- Iteracija = mali proizvod sa svim fazama (istovremeno); završava se na vreme i uz saglasnost naručioca.

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.
- Nije primenljivo svuda: svet se menja, specifikacije često ne mogu unapred potpuno; agilnost pomaže adaptaciji i može smanjiti troškove promena.
- **Agilni Manifest** (2001): 12 principa (npr. funkcionalan softver pre obimne dokumentacije, odgovor na promene pre praćenja plana, ...).
- Iteracija = mali proizvod sa svim fazama (istovremeno); završava se na vreme i uz saglasnost naručioca.

Agilna metodologija razvoja

- Cilj: minimizovanje rizika (bagovi, prekoračenje budžeta, izmena zahteva).
- Prioritet: isporuka ispred analize i dizajna (ali nisu obeshrabreni).
- Mali, visokomotivisani, samoorganizovani timovi; stalna komunikacija (često uživo) $\rightarrow$ manje pisanog traga i dokumentacije.
- Nije primenljivo svuda: svet se menja, specifikacije često ne mogu unapred potpuno; agilnost pomaže adaptaciji i može smanjiti troškove promena.
- **Agilni Manifest** (2001): 12 principa (npr. funkcionalan softver pre obimne dokumentacije, odgovor na promene pre praćenja plana, ...).
- Iteracija = mali proizvod sa svim fazama (istovremeno); završava se na vreme i uz saglasnost naručioca.

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

- Razvoj se odvija kroz **sprintove** (obično do mesec dana ili kraće).

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

- Razvoj se odvija kroz **sprintove** (obično do mesec dana ili kraće).
- Proizvod se održava u stanju koje se potencijalno može isporučiti.

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

- Razvoj se odvija kroz **sprintove** (obično do mesec dana ili kraće).
- Proizvod se održava u stanju koje se potencijalno može isporučiti.
- Na kraju svakog sprinta: sastanak aktera i tima radi pregleda stanja i planiranja.

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

- Razvoj se odvija kroz **sprintove** (obično do mesec dana ili kraće).
- Proizvod se održava u stanju koje se potencijalno može isporučiti.
- Na kraju svakog sprinta: sastanak aktera i tima radi pregleda stanja i planiranja.
- Skram ima jednostavan skup pravila/zaduženja/sastanaka koji se ne menja.

Skram (Scrum): ideja i ritam rada

Skram je vid agilne metodologije u kojem se neposredna, praktična iskustva koriste u upravljanju izazovima i rizicima.

- Razvoj se odvija kroz **sprintove** (obično do mesec dana ili kraće).
- Proizvod se održava u stanju koje se potencijalno može isporučiti.
- Na kraju svakog sprinta: sastanak aktera i tima radi pregleda stanja i planiranja.
- Skram ima jednostavan skup pravila/zaduženja/sastanaka koji se ne menja.
- Postoje sastanci na početku i kraju sprinta + **dnevni skram** (15 min).

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.
- Svaki Skram tim ima tri uloge: **vlasnik proizvoda**, **skram master**, **razvojni tim**.

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.
- Svaki Skram tim ima tri uloge: **vlasnik proizvoda**, **skram master**, **razvojni tim**.

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.
- Svaki Skram tim ima tri uloge: **vlasnik proizvoda**, **skram master**, **razvojni tim**.
- **Vlasnik proizvoda (product owner)**: određuje *šta* se razvija i *kojim redosledom*; kreira/redefiniše/procenjuje/prioritizuje **scrum backlog**; definiše kriterijume prihvatanja i proverava ih tokom sprinta.

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.
- Svaki Skram tim ima tri uloge: **vlasnik proizvoda**, **skram master**, **razvojni tim**.

- **Vlasnik proizvoda (product owner)**: određuje *šta* se razvija i *kojim redosledom*; kreira/redefiniše/procenjuje/prioritizuje **scrum backlog**; definiše kriterijume prihvatanja i proverava ih tokom sprinta.
- **Skram master**: olakšava komunikaciju, pomaže pridržavanje vrednosti/principa, uklanja prepreke i štiti tim od ometanja; nije tradicionalni menadžer.

Uloge u Skram timu

- Skram razvoj se sastoji od jednog ili više Skram timova.
- Svaki Skram tim ima tri uloge: **vlasnik proizvoda**, **skram master**, **razvojni tim**.

- **Vlasnik proizvoda (product owner)**: određuje *šta* se razvija i *kojim redosledom*; kreira/redefiniše/procenjuje/prioritizuje **scrum backlog**; definiše kriterijume prihvatanja i proverava ih tokom sprinta.
- **Skram master**: olakšava komunikaciju, pomaže pridržavanje vrednosti/principa, uklanja prepreke i štiti tim od ometanja; nije tradicionalni menadžer.
- **Razvojni tim**: samoorganizovan i multidisciplinaran (obično 5–9 članova); dizajnira/gradi/testira; pretvara stavke backloga u potencijalno isporučive funkcionalnosti; radi uz redovnu komunikaciju i zajedničku odgovornost.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.
- Sistem je integrisan i radi sve vreme (iako nema potpunu funkcionalnost).

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.
- Sistem je integrisan i radi sve vreme (iako nema potpunu funkcionalnost).
- Svi članovi tima poznaju ceo projekat; kôd je konzistentan i svako može da radi na svakom delu.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.
- Sistem je integrisan i radi sve vreme (iako nema potpunu funkcionalnost).
- Svi članovi tima poznaju ceo projekat; kôd je konzistentan i svako može da radi na svakom delu.
- Veoma mali koraci: prva iteracija može dati svesno nepotpunu, ali funkcionalnu celinu za **dan ili nedelju**.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.
- Sistem je integrisan i radi sve vreme (iako nema potpunu funkcionalnost).
- Svi članovi tima poznaju ceo projekat; kôd je konzistentan i svako može da radi na svakom delu.
- Veoma mali koraci: prva iteracija može dati svesno nepotpunu, ali funkcionalnu celinu za **dan ili nedelju**.
- Zahtevi se menjaju $\rightarrow$ naručilac je konstantno uključen; prikazuje se stalno funkcionalan (iako nekompletan) softver.

Ekstremno programiranje (XP)

- Vid agilne metodologije: posebno su važni **jednostavnost**, **motivacija** i **kvalitetni odnosi u timu**.
- **Pair programming**: jedan piše kôd, drugi traži greške i nedostatke (ili rad u većim grupama).
- Kôd jednostavnog dizajna koji se temeljno **testira** i **unapređuje** prema tekućim zahtevima.
- Sistem je integrisan i radi sve vreme (iako nema potpunu funkcionalnost).
- Svi članovi tima poznaju ceo projekat; kôd je konzistentan i svako može da radi na svakom delu.
- Veoma mali koraci: prva iteracija može dati svesno nepotpunu, ali funkcionalnu celinu za **dan ili nedelju**.
- Zahtevi se menjaju $\rightarrow$ naručilac je konstantno uključen; prikazuje se stalno funkcionalan (iako nekompletan) softver.
- Dokumentacija postoji, ali se izbegava preobimna

Eksploatacija

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.
- Deployment: isporuka nove verzije u ciljno okruženje (oblak ili lokalno), uz upravljanje **konfiguracijom** i **zavisnostima**.

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.
- Deployment: isporuka nove verzije u ciljno okruženje (oblak ili lokalno), uz upravljanje **konfiguracijom** i **zavisnostima**.
- Ključno: mogućnost brzog povratka na prethodnu verziju (rollback) ako se pojave problemi.

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.
- Deployment: isporuka nove verzije u ciljno okruženje (oblak ili lokalno), uz upravljanje **konfiguracijom** i **zavisnostima**.
- Ključno: mogućnost brzog povratka na prethodnu verziju (rollback) ako se pojave problemi.
- Automatizovani cevovodi izgradnje i isporuke (CI/CD):

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.
- Deployment: isporuka nove verzije u ciljno okruženje (oblak ili lokalno), uz upravljanje **konfiguracijom** i **zavisnostima**.
- Ključno: mogućnost brzog povratka na prethodnu verziju (rollback) ako se pojave problemi.
- Automatizovani cevovodi izgradnje i isporuke (CI/CD):
  - **Kontinuirana integracija (CI)**: automatizuje izgradnju i testiranje svake promene koda.

Eksploatacija: uvođenje u rad (deployment) i CI/CD

- Eksploatacija: softver se uvodi u rad i održava u stabilnom, bezbednom i predvidljivom režimu.
- Deployment: isporuka nove verzije u ciljno okruženje (oblak ili lokalno), uz upravljanje **konfiguracijom** i **zavisnostima**.
- Ključno: mogućnost brzog povratka na prethodnu verziju (rollback) ako se pojave problemi.
- Automatizovani cevovodi izgradnje i isporuke (CI/CD):
  - **Kontinuirana integracija (CI)**: automatizuje izgradnju i testiranje svake promene koda.
  - **Kontinuirana isporuka/puštanje u rad (CD)**: bezbedno i često objavljivanje novih verzija, uz manje rizika i kraće vreme do korisničke vrednosti.

Nakon puštanja: monitoring, oporavak i kraj životnog veka

- Post-deployment monitoring: prikupljanje **podataka** i **dnevnika rada (logs)** radi ranog otkrivanja degradacije performansi, regresija i incidenata.

Nakon puštanja: monitoring, oporavak i kraj životnog veka

- Post-deployment monitoring: prikupljanje **podataka** i **dnevnika rada (logs)** radi ranog otkrivanja degradacije performansi, regresija i incidenata.
- Pragovi, alarmi i jasno definisani ciljevi kvaliteta usluge.

Nakon puštanja: monitoring, oporavak i kraj životnog veka

- Post-deployment monitoring: prikupljanje **podataka** i **dnevnika rada (logs)** radi ranog otkrivanja degradacije performansi, regresija i incidenata.
- Pragovi, alarmi i jasno definisani ciljevi kvaliteta usluge.
- Strategije oporavka + kontrola pristupa + redovno ažuriranje + provera ranjivosti.

Nakon puštanja: monitoring, oporavak i kraj životnog veka

- Post-deployment monitoring: prikupljanje **podataka** i **dnevnika rada (logs)** radi ranog otkrivanja degradacije performansi, regresija i incidenata.
- Pragovi, alarmi i jasno definisani ciljevi kvaliteta usluge.
- Strategije oporavka + kontrola pristupa + redovno ažuriranje + provera ranjivosti.
- Upravljanje krajem životnog veka (end-of-life): planirano gašenje ili migracija sistema, komunikacija sa korisnicima, arhiviranje i zaštita podataka, usklađenost sa propisima i planiranje zamena (manji tehnički dug i operativni rizik).

Alati i tehnike korišćeni u razvoju softvera

Alati i tehnike korišćeni u razvoju softvera

Alati za upravljanje projektima

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:
  - **ljudima** (timovi i pojedinci),

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:
  - **ljudima** (timovi i pojedinci),
  - **procesima** (metodologije, radni tokovi, standardi),

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:
  - **ljudima** (timovi i pojedinci),
  - **procesima** (metodologije, radni tokovi, standardi),
  - **problemima** (zahtevi, ograničenja, rizici).

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:
  - **ljudima** (timovi i pojedinci),
  - **procesima** (metodologije, radni tokovi, standardi),
  - **problemima** (zahtevi, ograničenja, rizici).
- Cilj: balans **kvalitet** – **vreme** – **troškovi**.

Upravljanje softverskim projektima: šta obuhvata

- Upravljanje projektima: planiranje, organizacija, praćenje i kontrola procesa razvoja.
- Upravljanje:
  - **ljudima** (timovi i pojedinci),
  - **procesima** (metodologije, radni tokovi, standardi),
  - **problemima** (zahtevi, ograničenja, rizici).
- Cilj: balans **kvalitet** – **vreme** – **troškovi**.
- Ključni pojmovi: metrike, procena obima/složenosti, upravljanje rizicima, raspored aktivnosti, održavanje i reinženjering postojećih rešenja.

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,
  - pokrivenost testovima,

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,
  - pokrivenost testovima,
  - broj otvorenih bagova.

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,
  - pokrivenost testovima,
  - broj otvorenih bagova.
- Na osnovu metrika: procene troškova/trajanja i praćenje usklađenosti sa planom.

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,
  - pokrivenost testovima,
  - broj otvorenih bagova.
- Na osnovu metrika: procene troškova/trajanja i praćenje usklađenosti sa planom.
- Alati za rizike: identifikacija + rano uočavanje trendova/indikatora problema $\rightarrow$ pravovremene akcije (sprečiti/ublažiti).

Šta daju alati: praćenje, metrike, rizici, primeri

- Alati nisu samo „podsetnici“: podrška za odluke i evaluaciju napretka.
- Osnovno: zadaci, rokovi, organizacija timskog rada.
- Metrike (primeri iz teksta):
  - broj završenih zadataka po vremenu,
  - pokrivenost testovima,
  - broj otvorenih bagova.
- Na osnovu metrika: procene troškova/trajanja i praćenje usklađenosti sa planom.
- Alati za rizike: identifikacija + rano uočavanje trendova/indikatora problema $\rightarrow$ pravovremene akcije (sprečiti/ublažiti).
- Primeri sa tržišta: **Jira** (izveštavanje, integracije, skram/kanban), **Trello** (jednostavno vizuelno, manje timove i agilne procese).

Alati i tehnike korišćeni u razvoju softvera

Sistemi za kontrolu verzija

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.
- Korisno i za pouzdanu strategiju rezervnih kopija (kvar diska, greške u kodu).

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.
- Korisno i za pouzdanu strategiju rezervnih kopija (kvar diska, greške u kodu).

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.
- Korisno i za pouzdanu strategiju rezervnih kopija (kvar diska, greške u kodu).

Repozitorijum čuva izvorni kod i istoriju izmena:

- verzije datoteka (datum/vreme),

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.
- Korisno i za pouzdanu strategiju rezervnih kopija (kvar diska, greške u kodu).

Repozitorijum čuva izvorni kod i istoriju izmena:

- verzije datoteka (datum/vreme),
- istoriju izmena (ko/kada/šta + opis),

Sistemi za kontrolu verzija (VCS): svrha

- VCS: praćenje promena, saradnja i istorija projekta.
- Nazivi koji se sreću: SCM (Source Code Management), RCS (Revision Control System).
- Suština: čuvanje sadržaja + beleženje promena + pristup različitim verzijama.
- Korisno i za pouzdanu strategiju rezervnih kopija (kvar diska, greške u kodu).

Repozitorijum čuva izvorni kod i istoriju izmena:

- verzije datoteka (datum/vreme),
- istoriju izmena (ko/kada/šta + opis),
- grane (paralelan razvoj i kasnije spajanje).

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.
- Tok: preuzmi $\rightarrow$ izmeni $\rightarrow$ vrati izmene u repozitorijum.

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.
- Tok: preuzmi $\rightarrow$ izmeni $\rightarrow$ vrati izmene u repozitorijum.
- Ako više ljudi radi na istoj komponenti: sistem pomaže pri integraciji i upozorava na konflikte.

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.
- Tok: preuzmi $\rightarrow$ izmeni $\rightarrow$ vrati izmene u repozitorijum.
- Ako više ljudi radi na istoj komponenti: sistem pomaže pri integraciji i upozorava na konflikte.
- Opisi izmena su važni: formalna komunikacija u timu (često obavezno pravilo).

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.
- Tok: preuzmi $\rightarrow$ izmeni $\rightarrow$ vrati izmene u repozitorijum.
- Ako više ljudi radi na istoj komponenti: sistem pomaže pri integraciji i upozorava na konflikte.
- Opisi izmena su važni: formalna komunikacija u timu (često obavezno pravilo).

Timski rad: javni repozitorijum i privatni radni prostor

- Model rada: **javni repozitorijum** + **privatni radni prostor**.
- Tok: preuzmi $\rightarrow$ izmeni $\rightarrow$ vrati izmene u repozitorijum.
- Ako više ljudi radi na istoj komponenti: sistem pomaže pri integraciji i upozorava na konflikte.
- Opisi izmena su važni: formalna komunikacija u timu (često obavezno pravilo).

- Skladištenje verzija često koristi *delte* (čuvaju se razlike između verzija) radi uštede prostora.

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).
- **Mercurial**: takođe distribuiran, sličan Gitu.

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).
- **Mercurial**: takođe distribuiran, sličan Gitu.

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).
- **Mercurial**: takođe distribuiran, sličan Gitu.

- Kratko poreklo: Linux kernel je ranije koristio BitKeeper; 2005. dolazi do promene uslova, pa inicira razvoj Gita.

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).
- **Mercurial**: takođe distribuiran, sličan Gitu.

- Kratko poreklo: Linux kernel je ranije koristio BitKeeper; 2005. dolazi do promene uslova, pa inicira razvoj Gita.
- Ciljevi dizajna (sažeto): distribuiran razvoj, brzina/efikasnost (kompresija + delte), pouzdanost (SHA1 integritet), nepromenljivost istorije, atomske transakcije, grane, slobodan alat.

Git i drugi sistemi (SVN, Mercurial)

- **Git**: distribuirani VCS — svaki korisnik ima punu istoriju repozitorijuma (rad van mreže, veća otpornost).
- **SVN**: centralizovan — jedan centralni repozitorijum (lakši nadzor, manje fleksibilno).
- **Mercurial**: takođe distribuiran, sličan Gitu.

- Kratko poreklo: Linux kernel je ranije koristio BitKeeper; 2005. dolazi do promene uslova, pa inicira razvoj Gita.
- Ciljevi dizajna (sažeto): distribuiran razvoj, brzina/efikasnost (kompresija + delte), pouzdanost (SHA1 integritet), nepromenljivost istorije, atomske transakcije, grane, slobodan alat.
- U knjizi: Git preko komandne linije (GUI klijenti često nude samo deo funkcionalnosti).

Alati i tehnike korišćeni u razvoju softvera

Distribuirani softverski sistemi

Distribuirani softverski sistemi

- **Distribuiran sistem** = softver koji radi na **više računara/čvorova** i komunicira preko mreže.

Distribuirani softverski sistemi

- **Distribuiran sistem** = softver koji radi na **više računara/čvorova** i komunicira preko mreže.
- Ciljevi: **skaliranje**, **dostupnost** i **otpornost na kvarove**.

Distribuirani softverski sistemi

- **Distribuiran sistem** = softver koji radi na **više računara/čvorova** i komunicira preko mreže.
- Ciljevi: **skaliranje**, **dostupnost** i **otpornost na kvarove**.
- Cena: više složenosti u **komunikaciji, sinhronizaciji, testiranju i bezbednosti**.

Distribuirani softverski sistemi

- **Distribuiran sistem** = softver koji radi na **više računara/čvorova** i komunicira preko mreže.
- Ciljevi: **skaliranje**, **dostupnost** i **otpornost na kvarove**.
- Cena: više složenosti u **komunikaciji, sinhronizaciji, testiranju i bezbednosti**.
- Performanse zavise i od **mreže** (propusnost, latencija, opterećenje), ne samo CPU-a.

Osnovne osobine distribuiranih sistema

- **Deljenje resursa:** zajedničko korišćenje diskova, baza, servisa, itd.

Osnovne osobine distribuiranih sistema

- **Deljenje resursa:** zajedničko korišćenje diskova, baza, servisa, itd.
- **Konkurentnost:** delovi sistema rade paralelno (često uz asinhronu komunikaciju).

Osnovne osobine distribuiranih sistema

- **Deljenje resursa:** zajedničko korišćenje diskova, baza, servisa, itd.
- **Konkurentnost:** delovi sistema rade paralelno (često uz asinhronu komunikaciju).
- **Skalabilnost:** kapacitet raste dodavanjem ili jačanjem resursa.

Osnovne osobine distribuiranih sistema

- **Deljenje resursa:** zajedničko korišćenje diskova, baza, servisa, itd.
- **Konkurentnost:** delovi sistema rade paralelno (često uz asinhronu komunikaciju).
- **Skalabilnost:** kapacitet raste dodavanjem ili jačanjem resursa.
- **Otpornost na greške:** sistem nastavlja rad i kad neki čvor otkaže (npr. replikacija, preusmeravanje).

Skaliranje: *scale up* vs *scale out*

- **Scale up:** jači postojeći čvor (više RAM/CPU) $\rightarrow$ brže, ali često skuplje i ograničeno hardverom.

Mini-provera (30s): Kod kog pristupa je važnije dobro balansiranje opterećenja i zašto?

Skaliranje: *scale up* vs *scale out*

- **Scale up:** jači postojeći čvor (više RAM/CPU) $\rightarrow$ brže, ali često skuplje i ograničeno hardverom.
- **Scale out:** dodavanje novih čvorova $\rightarrow$ fleksibilnije i često isplativije, ali zahteva dobar dizajn.

Mini-provera (30s): Kod kog pristupa je važnije dobro balansiranje opterećenja i zašto?

Skaliranje: *scale up* vs *scale out*

- **Scale up:** jači postojeći čvor (više RAM/CPU) $\rightarrow$ brže, ali često skuplje i ograničeno hardverom.
- **Scale out:** dodavanje novih čvorova $\rightarrow$ fleksibilnije i često isplativije, ali zahteva dobar dizajn.
- Tipični problem: **mrežna uska grla** i **ravnomerna raspodela opterećenja**.

Mini-provera (30s): Kod kog pristupa je važnije dobro balansiranje opterećenja i zašto?

Arhitektura: klijent–server (osnova)

- Najčešći model za sisteme dostupne preko interneta.

Arhitektura: klijent–server (osnova)

- Najčešći model za sisteme dostupne preko interneta.
- **Klijent** (pregledač/aplikacija) prikazuje i šalje zahteve; **server** obavlja obradu i pristupa podacima.

Arhitektura: klijent–server (osnova)

- Najčešći model za sisteme dostupne preko interneta.
- **Klijent** (pregledač/aplikacija) prikazuje i šalje zahteve; **server** obavlja obradu i pristupa podacima.
- Često postoji **više serverskih instanci** (na više računara) radi većeg kapaciteta.

Arhitektura: klijent–server (osnova)

- Najčešći model za sisteme dostupne preko interneta.
- **Klijent** (pregledač/aplikacija) prikazuje i šalje zahteve; **server** obavlja obradu i pristupa podacima.
- Često postoji **više serverskih instanci** (na više računara) radi većeg kapaciteta.
- **Balansiranje opterećenja:** zahtevi se raspoređuju na više servera da bi sistem podneo veći broj korisnika.

Varijante arhitektura

- **Master–rob:** master dodeljuje zadatke robovima; korisno kad su bitna stroga vremenska ograničenja (npr. real-time).

Varijante arhitektura

- **Master–rob:** master dodeljuje zadatke robovima; korisno kad su bitna stroga vremenska ograničenja (npr. real-time).
- **Dvoslojna / višeslojna klijent–server:**

Varijante arhitektura

- **Master–rob:** master dodeljuje zadatke robovima; korisno kad su bitna stroga vremenska ograničenja (npr. real-time).
- **Dvoslojna / višeslojna klijent–server:**
  - dvoslojna: klijent direktno priča sa jednim serverom;

Varijante arhitektura

- **Master–rob:** master dodeljuje zadatke robovima; korisno kad su bitna stroga vremenska ograničenja (npr. real-time).
- **Dvoslojna / višeslojna klijent–server:**
  - dvoslojna: klijent direktno priča sa jednim serverom;
  - višeslojna: prezentacioni / aplikacioni / data sloj (mogu na odvojenim čvorovima) $\rightarrow$ lakše skaliranje i održavanje.

Varijante arhitektura

- **Master–rob:** master dodeljuje zadatke robovima; korisno kad su bitna stroga vremenska ograničenja (npr. real-time).
- **Dvoslojna / višeslojna klijent–server:**
  - dvoslojna: klijent direktno priča sa jednim serverom;
  - višeslojna: prezentacioni / aplikacioni / data sloj (mogu na odvojenim čvorovima) $\rightarrow$ lakše skaliranje i održavanje.
- **P2P (peer-to-peer):** nema stroge podele na klijent/server; čvorovi razmenjuju resurse direktno (prednost: otpornost; mana: više koordinacije/overheada).

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.
  - Pošiljalac obično **čeka odgovor**.

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.
  - Pošiljalac obično **čeka odgovor**.
  - Zahteva da su obe strane **istovremeno dostupne**.

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.
  - Pošiljalac obično **čeka odgovor**.
  - Zahteva da su obe strane **istovremeno dostupne**.
- **Poruke + redovi (message-based):** asinhrono slanje; poruka čeka dok primalac ne bude dostupan.

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.
  - Pošiljalac obično **čeka odgovor**.
  - Zahteva da su obe strane **istovremeno dostupne**.
- **Poruke + redovi (message-based):** asinhrono slanje; poruka čeka dok primalac ne bude dostupan.
  - Bolje podnosi **privremenu nedostupnost**.

Komunikacioni modeli: RPC vs poruke

- **RPC/RMI (proceduralno):** poziv udaljene usluge kao da je lokalna funkcija/metoda.
  - Pošiljalac obično **čeka odgovor**.
  - Zahteva da su obe strane **istovremeno dostupne**.
- **Poruke + redovi (message-based):** asinhrono slanje; poruka čeka dok primalac ne bude dostupan.
  - Bolje podnosi **privremenu nedostupnost**.
  - Često postoji **međusoftver** koji rutira poruke i brine o transformaciji/pouzdanosti.

Bezbednost i eksploatacija u distribuiranim sistemima

- Više čvorova i veza $\Rightarrow$ veća **površina napada** (presretanje, neovlašćen pristup, DoS, lažni podaci).

Bezbednost i eksploatacija u distribuiranim sistemima

- Više čvorova i veza $\Rightarrow$ veća **površina napada** (presretanje, neovlašćen pristup, DoS, lažni podaci).
- Osnove: **enkripcija**, **autentifikacija**, **kontrola pristupa**, redovno ažuriranje i provera ranjivosti.

Bezbednost i eksploatacija u distribuiranim sistemima

- Više čvorova i veza $\Rightarrow$ veća **površina napada** (presretanje, neovlašćen pristup, DoS, lažni podaci).
- Osnove: **enkripcija**, **autentifikacija**, **kontrola pristupa**, redovno ažuriranje i provera ranjivosti.
- Operativno: **monitoring** + **logovi** + **alarmi** radi ranog otkrivanja regresija i incidenata.

Bezbednost i eksploatacija u distribuiranim sistemima

- Više čvorova i veza $\Rightarrow$ veća **površina napada** (presretanje, neovlašćen pristup, DoS, lažni podaci).
- Osnove: **enkripcija**, **autentifikacija**, **kontrola pristupa**, redovno ažuriranje i provera ranjivosti.
- Operativno: **monitoring** + **logovi** + **alarmi** radi ranog otkrivanja regresija i incidenata.
- **Oblak** se često koristi kao "prirodan nastavak" distribuiranih sistema: resursi se lako povećavaju/smanjuju po potrebi.

Alati i tehnike korišćeni u razvoju softvera

Mikroservisno orijentisan razvoj softvera

Mikroservisi: ideja i poreklo

- **CBSE (1990-te):** sistemi se grade spajanjem *komponenti* (“crnih kutija”) sa jasnim interfejsima.

Mikroservisi: ideja i poreklo

- **CBSE (1990-te):** sistemi se grade spajanjem *komponenti* (“crnih kutija”) sa jasnim interfejsima.
- Problem CBSE prakse: teže je **nezavisno razvijati** delove i **skalirati samo ono što treba**.

Mikroservisi: ideja i poreklo

- **CBSE (1990-te):** sistemi se grade spajanjem *komponenti* ("crnih kutija") sa jasnim interfejsima.
- Problem CBSE prakse: teže je **nezavisno razvijati** delove i **skalirati samo ono što treba**.
- **Mikroservisi** proširuju ideju: sistem se ne gradi kao jedan monolit, već kao **skup manjih servisa**.

Mikroservisi: ideja i poreklo

- **CBSE (1990-te):** sistemi se grade spajanjem *komponenti* ("crnih kutija") sa jasnim interfejsima.
- Problem CBSE prakse: teže je **nezavisno razvijati** delove i **skalirati samo ono što treba**.
- **Mikroservisi** proširuju ideju: sistem se ne gradi kao jedan monolit, već kao **skup manjih servisa**.
- Svaki servis je **samostalno isporučiv** i ima **sopstveni ciklus razvoja** (razvoj/test/deploy nezavisno).

Mikroservisi: ideja i poreklo

- **CBSE (1990-te):** sistemi se grade spajanjem *komponenti* ("crnih kutija") sa jasnim interfejsima.
- Problem CBSE prakse: teže je **nezavisno razvijati** delove i **skalirati samo ono što treba**.
- **Mikroservisi** proširuju ideju: sistem se ne gradi kao jedan monolit, već kao **skup manjih servisa**.
- Svaki servis je **samostalno isporučiv** i ima **sopstveni ciklus razvoja** (razvoj/test/deploy nezavisno).
- Servisi komuniciraju **mrežnim protokolima** $\Rightarrow$ prirodno se uklapaju u distribuirane sisteme.

Ključni principi, prednosti i izazovi

- Tri principa:

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.
  - **Veličina:** čim servis preraste (previše funkcionalnosti) $\Rightarrow$ podela na manje.

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.
  - **Veličina:** čim servis preraste (previše funkcionalnosti) $\Rightarrow$ podela na manje.
  - **Nezavisnost:** razvijanje, testiranje, deploy i skaliranje bez "blokiranja" drugih servisa.

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.
  - **Veličina:** čim servis preraste (previše funkcionalnosti) $\Rightarrow$ podela na manje.
  - **Nezavisnost:** razvijanje, testiranje, deploy i skaliranje bez "blokiranja" drugih servisa.
- **Prednosti:** skaliraš samo opterećeni servis; timovi rade paralelno; različite tehnologije po servisu; veća robusnost (pad jednog servisa ne ruši ceo sistem).

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.
  - **Veličina:** čim servis preraste (previše funkcionalnosti) $\Rightarrow$ podela na manje.
  - **Nezavisnost:** razvijanje, testiranje, deploy i skaliranje bez "blokiranja" drugih servisa.
- **Prednosti:** skaliraš samo opterećeni servis; timovi rade paralelno; različite tehnologije po servisu; veća robusnost (pad jednog servisa ne ruši ceo sistem).
- **Tipična isporuka:** servisi često u **kontejnerima** (npr. Docker) sa svim zavisnostima i konfiguracijom.

Ključni principi, prednosti i izazovi

- **Tri principa:**
  - **Ograničeni kontekst (bounded context):** servis ima jasnu poslovnu odgovornost i granice.
  - **Veličina:** čim servis preraste (previše funkcionalnosti) $\Rightarrow$ podela na manje.
  - **Nezavisnost:** razvijanje, testiranje, deploy i skaliranje bez "blokiranja" drugih servisa.
- **Prednosti:** skaliraš samo opterećeni servis; timovi rade paralelno; različite tehnologije po servisu; veća robusnost (pad jednog servisa ne ruši ceo sistem).
- **Tipična isporuka:** servisi često u **kontejnerima** (npr. Docker) sa svim zavisnostima i konfiguracijom.
- **Izazovi:** sporija/skuplja mrežna komunikacija; rizik preopterećenja zajedničkog servisa; teže testiranje i debagovanje (interakcije); pitanja **podataka i konzistentnosti** (posebno ako se deli baza).

Alati i tehnike korišćeni u razvoju softvera

Ugrađeni softver

Ugrađeni softver: šta je i zašto je poseban

- **Ugrađeni (embedded) softver** je deo **hardversko-softverskog** sistema: upravlja uređajem i reaguje na događaje iz okruženja.

Ugrađeni softver: šta je i zašto je poseban

- **Ugrađeni (embedded) softver** je deo **hardversko-softverskog** sistema: upravlja uređajem i reaguje na događaje iz okruženja.
- Radi pod **ograničenim resursima**: memorija, procesorska snaga, energija $\Rightarrow$ optimizacije nisu luksuz nego obaveza.

Ugrađeni softver: šta je i zašto je poseban

- **Ugrađeni (embedded) softver** je deo **hardversko-softverskog** sistema: upravlja uređajem i reaguje na događaje iz okruženja.
- Radi pod **ograničenim resursima**: memorija, procesorska snaga, energija $\Rightarrow$ optimizacije nisu luksuz nego obaveza.
- Često je **real-time**: odgovor mora stići u **strogo definisanom roku**. Kašnjenje može imati ozbiljne posledice (npr. kočenje).

Ugrađeni softver: šta je i zašto je poseban

- **Ugrađeni (embedded) softver** je deo **hardversko-softverskog** sistema: upravlja uređajem i reaguje na događaje iz okruženja.
- Radi pod **ograničenim resursima**: memorija, procesorska snaga, energija $\Rightarrow$ optimizacije nisu luksuz nego obaveza.
- Često je **real-time**: odgovor mora stići u **strogo definisanom roku**. Kašnjenje može imati ozbiljne posledice (npr. kočenje).
- **Pouzdanost i sigurnost** su ključne (automobili, medicinski uređaji, saobraćaj, telekom).

Ugrađeni softver: šta je i zašto je poseban

- **Ugrađeni (embedded) softver** je deo **hardversko-softverskog** sistema: upravlja uređajem i reaguje na događaje iz okruženja.
- Radi pod **ograničenim resursima**: memorija, procesorska snaga, energija $\Rightarrow$ optimizacije nisu luksuz nego obaveza.
- Često je **real-time**: odgovor mora stići u **strogo definisanom roku**. Kašnjenje može imati ozbiljne posledice (npr. kočenje).
- **Pouzdanost i sigurnost** su ključne (automobili, medicinski uređaji, saobraćaj, telekom).
- Dizajn ograničavaju i **fizički uslovi** (prostor, temperatura, vibracije, energetska efikasnost).

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.
- U praksi se softver često realizuje kao **skup konkurentnih procesa/niti** koji komuniciraju.

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.
- U praksi se softver često realizuje kao **skup konkurentnih procesa/niti** koji komuniciraju.
- Zato se često koristi **RTOS** (Real-Time OS): scheduler upravlja vremenom i resursima da bi se poštovali rokovi.

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.
- U praksi se softver često realizuje kao **skup konkurentnih procesa/niti** koji komuniciraju.
- Zato se često koristi **RTOS** (Real-Time OS): scheduler upravlja vremenom i resursima da bi se poštovali rokovi.
- Opšti OS (Windows / standardni Linux) **ne garantuju** real-time; postoje real-time varijante Linux-a (npr. RTLinux / PREEMPT-RT).

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.
- U praksi se softver često realizuje kao **skup konkurentnih procesa/niti** koji komuniciraju.
- Zato se često koristi **RTOS** (Real-Time OS): scheduler upravlja vremenom i resursima da bi se poštovali rokovi.
- Opšti OS (Windows / standardni Linux) **ne garantuju** real-time; postoje real-time varijante Linux-a (npr. RTLinux / PREEMPT-RT).
- Ponašanje sistema se često modeluje **dijagramima stanja** (reakcije na periodične i aperiodične signale).

Razvoj: RTOS, konkurentnost i modelovanje ponašanja

- Razvoj je **interdisciplinaran**: hardver + softver, uz stalnu integraciju i sinhronizaciju.
- U praksi se softver često realizuje kao **skup konkurentnih procesa/niti** koji komuniciraju.
- Zato se često koristi **RTOS** (Real-Time OS): scheduler upravlja vremenom i resursima da bi se poštovali rokovi.
- Opšti OS (Windows / standardni Linux) **ne garantuju** real-time; postoje real-time varijante Linux-a (npr. RTLinux / PREEMPT-RT).
- Ponašanje sistema se često modeluje **dijagramima stanja** (reakcije na periodične i aperiodične signale).
- Često se koristi **C** zbog efikasnosti, ali bez ugrađene podrške za konkurentnost $\Rightarrow$ oslanjanje na RTOS mehanizme (semafori, međusobno isključivanje) i veći rizik grešaka ako se njima loše upravlja.

Alati i tehnike korišćeni u razvoju softvera

Veštačka inteligencija u razvoju softvera

Veštačka inteligencija u razvoju softvera: kada ima smisla

- VI **pomaže** kada postoji **obrazac** koji je teško eksplicitno isprogramirati (nelinearno, šumovito, kompleksno).

Kada VI nije dobar izbor: pravila su jasna i stabilna (npr. jednostavne formule), nema obrasca ili nema podataka, ili je greška preskupa.

Veštačka inteligencija u razvoju softvera: kada ima smisla

- VI **pomaže** kada postoji **obrazac** koji je teško eksplicitno isprogramirati (nelinearno, šumovito, kompleksno).
- Potrebni su **dostupni i kvalitetni podaci** (reprezentativni za realne uslove).

Kada VI nije dobar izbor: pravila su jasna i stabilna (npr. jednostavne formule), nema obrasca ili nema podataka, ili je greška preskupa.

Veštačka inteligencija u razvoju softvera: kada ima smisla

- VI **pomaže** kada postoji **obrazac** koji je teško eksplicitno isprogramirati (nelinearno, šumovito, kompleksno).
- Potrebni su **dostupni i kvalitetni podaci** (reprezentativni za realne uslove).
- Problem je često **prediktivan** (zaključivanje iz istorijskih podataka).

Kada VI nije dobar izbor: pravila su jasna i stabilna (npr. jednostavne formule), nema obrasca ili nema podataka, ili je greška preskupa.

Veštačka inteligencija u razvoju softvera: kada ima smisla

- VI **pomaže** kada postoji **obrazac** koji je teško eksplicitno isprogramirati (nelinearno, šumovito, kompleksno).
- Potrebni su **dostupni i kvalitetni podaci** (reprezentativni za realne uslove).
- Problem je često **prediktivan** (zaključivanje iz istorijskih podataka).
- **Cena greške** treba da bude prihvatljiva: VI nije egzaktna $\Rightarrow$ za visoko-rizične domene često ide **VI + ljudska ekspertiza** / dodatna provera.

Kada VI nije dobar izbor: pravila su jasna i stabilna (npr. jednostavne formule), nema obrasca ili nema podataka, ili je greška preskupa.

Veštačka inteligencija u razvoju softvera: kada ima smisla

- VI **pomaže** kada postoji **obrazac** koji je teško eksplicitno isprogramirati (nelinearno, šumovito, kompleksno).
- Potrebni su **dostupni i kvalitetni podaci** (reprezentativni za realne uslove).
- Problem je često **prediktivan** (zaključivanje iz istorijskih podataka).
- **Cena greške** treba da bude prihvatljiva: VI nije egzaktna $\Rightarrow$ za visoko-rizične domene često ide **VI + ljudska ekspertiza** / dodatna provera.
- Ako se **obrasci menjaju vremenom**, VI može da se re-trenira i prilagođava.

Kada VI nije dobar izbor: pravila su jasna i stabilna (npr. jednostavne formule), nema obrasca ili nema podataka, ili je greška preskupa.

Razvoj VI rešenja i VI kao alat programera

- VI se koristi i kao **pomoćni alat** u razvoju: asistenti (npr. GitHub Copilot, ChatGPT, Cline) za dopunu koda, testove, refaktorisanje, dokumentaciju.

Razvoj VI rešenja i VI kao alat programera

- VI se koristi i kao **pomoćni alat** u razvoju: asistenti (npr. GitHub Copilot, ChatGPT, Cline) za dopunu koda, testove, refaktorisanje, dokumentaciju.
- Ograničenja: alati mogu generisati **netačan/neefikasan kod**; postoje i **bezbednosne i pravne** brige (slanje vlasničkog koda, pitanja autorskih prava).

Razvoj VI rešenja i VI kao alat programera

- VI se koristi i kao **pomoćni alat** u razvoju: asistenti (npr. GitHub Copilot, ChatGPT, Cline) za dopunu koda, testove, refaktorisanje, dokumentaciju.
- Ograničenja: alati mogu generisati **netačan/neefikasan kod**; postoje i **bezbednosne i pravne** brige (slanje vlasničkog koda, pitanja autorskih prava).
- Praksa često koristi **hibridni pristup**: asistenti za nepoverljive zadatke, a lokalni modeli/klasični alati za osetljiv kod.