

1

Upravljanje verzijama

2018/19.03

- Kontrola verzija (Version control) = verzioniranje
 - kombinira procedure i alate radi upravljanja različitim verzijama objekata konfiguracije, koji nastaju softverskim procesima
- Mogućnosti sustava kontrole verzija
 - baza projekata (project database) ili riznica (repository)
 - pohranjuje sve relevantne objekte konfiguracije
 - verzioniranje
 - razlikovanje pohranjenih inačica objekata konfiguracije
 - pomagalo za izradu (make facility)
 - prikuplja relevantne objekte i proizvodi određenu verziju softvera
 - praćenje problema (issues tracking), praćenje pogreški (bug tracking)
 - bilježenje i praćenje statusa tema koje se odnose na pojedine objekte konfiguracije

Upravljanje konfiguracijom

3

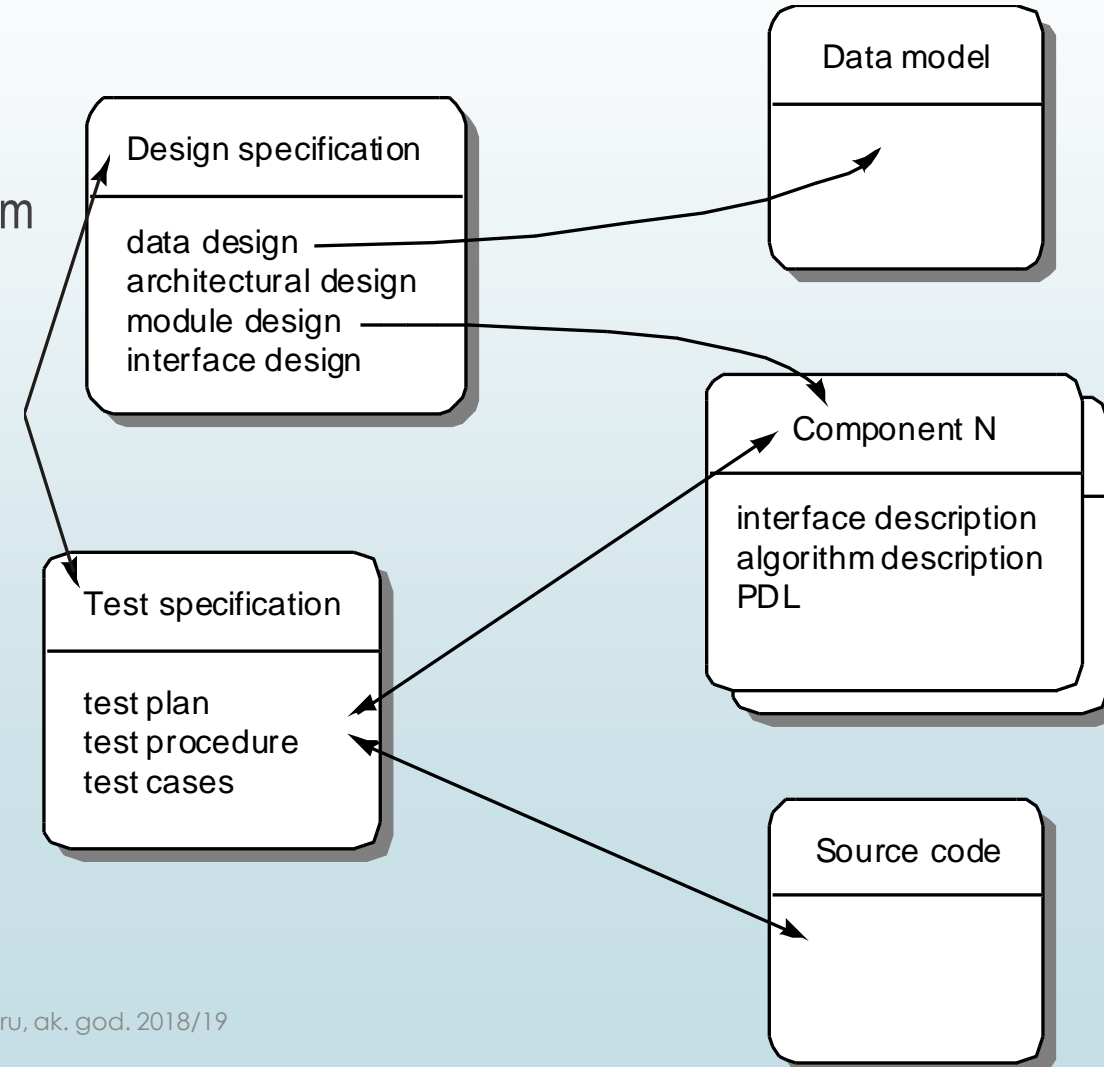
➤ Konfiguracija

- imenovani skup konfiguracijskih elemenata u određenoj točki životnog ciklusa

➤ Element konfiguracije (IEEE)

- agregacija hardvera i/ili softvera koja se tretira kao jedinka u procesu upravljanja konfiguracijom

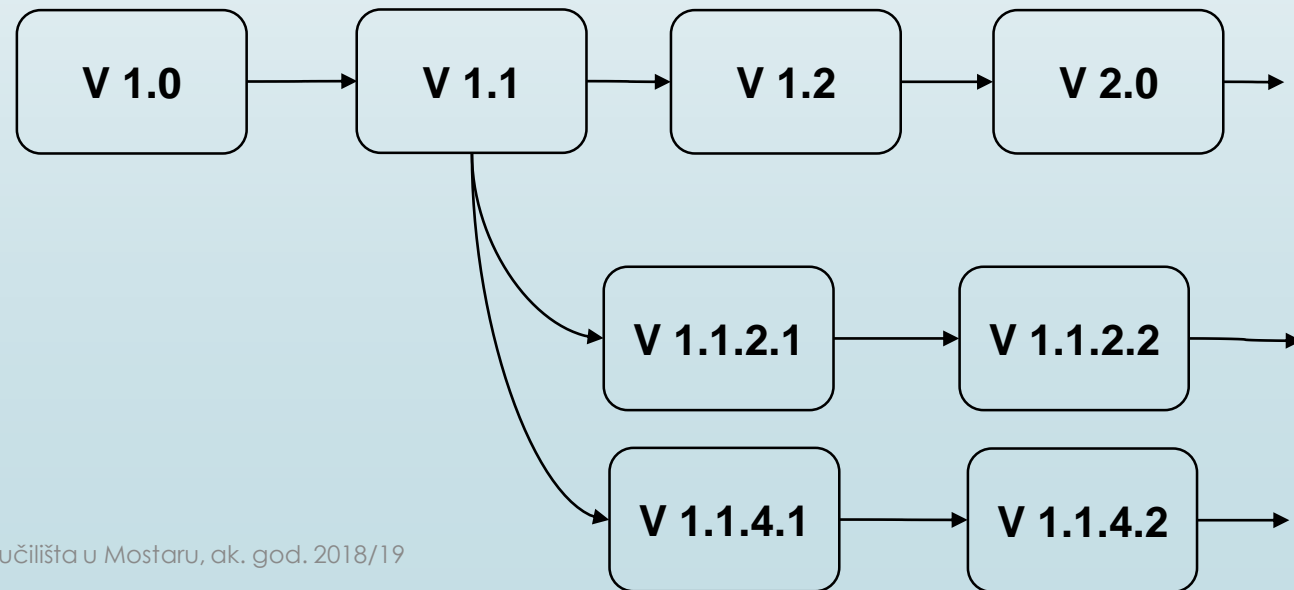
➤ Objekti konfiguracije



Verzije konfiguracije

4

- verzija, inačica (version) – određeno izdanje (issue, release) proizvoda
- objava, isporuka (release) – originalna verzija u primjeni, npr. zadnja v2.0
- revizija (revision) – ona koja se koristi umjesto originalne, podrazumijeva izmjene nastale kroz vrijeme (npr. zbog ispravljanja pogrešaka), npr. V1.2
- varijanta (variant) – alternativa originalu (hardverska platforma, različiti jezik), živi paralelno s njim, npr. v1.1.2.1
- osnovica (Baseline) – specifikacija proizvoda formalno provjerena i usvojena, koja služi kao temelj razvoja i koja se mijenja samo kroz formalnu proceduru kontrole promjena, IEEE (IEEE Std. No. 610.12-1990)



Označavanje verzija

5

- Verzija objektne datoteke u .NET Frameworku (assembly) određena je s četiri broja:

`<major version>.<minor version>.<build number>.<revision>`

- major version - mijenja se prilikom znatne promjene u (npr. kod redizajna koji prekida vertikalnu kompatibilnost sa starijim verzijama)
- minor version - mijenja se prilikom znatne promjene, ali uz zadržavanje kompatibilnosti s prethodnim verzijama
- build number - predstavlja ponovno prevođenje istog koda (npr. prilikom promjene platforme, procesora i slično)
- revision - primjenjuje se npr. prilikom izdavanja sigurnosnih zakrpa i sličnih manjih promjena

- Primjer: Properties \ AssemblyInfo

- major.minor.* (ili major.minor.build.*) automatski određuje build number i revision
 - build number: broj dana od 1.1.2000.
 - revision: broj sekundi proteklih od ponoći aktualnog dana podijeljen s 2

- .NET Core koristi Semantic Versioning: `major.minor.patch-suffix`

➤ Automatsko označavanje

➤ prednosti:

- eliminacija ručnog rada (npr. pisanja i izvedbe skripti)
- ne postoje dvije inačice s istom oznakom

➤ nedostaci:

- oznaka elementa ne podudara se s oznakom cijelog sustava
- novi brojevi ovise o danu i vremenu prevođenja
- verzija se mijenja pri svakom prevođenju, neovisno o tome jesu li se dogodile promjene ili ne

➤ Ručno verzioniranje

➤ prednosti:

- potpuna kontrola nad brojevima verzije
- moguća je sinkronizacija između verzije pojedinih komponenti i verzije cijelog sustava

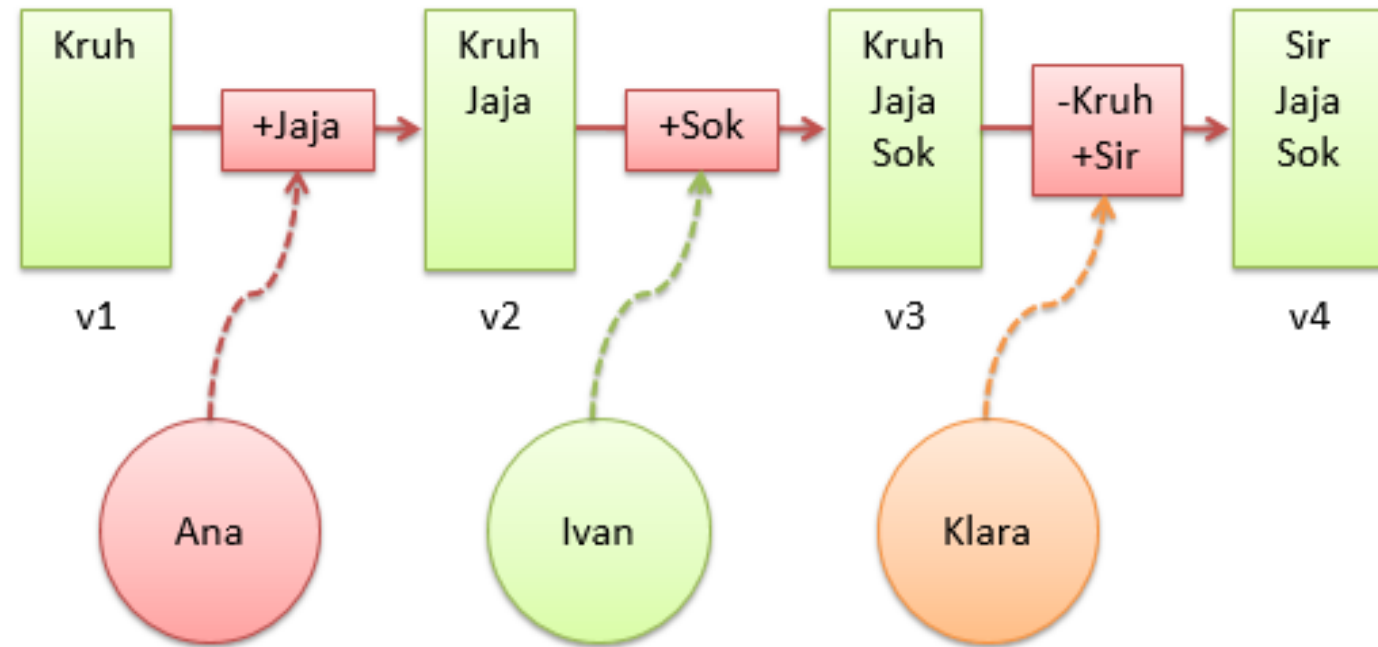
➤ nedostaci:

- verzioniranje se mora raditi ručno
- moguće je napraviti više različitih objektnih datoteka s istim oznakama

Sustavi za upravljanje verzijama izvornog koda

7

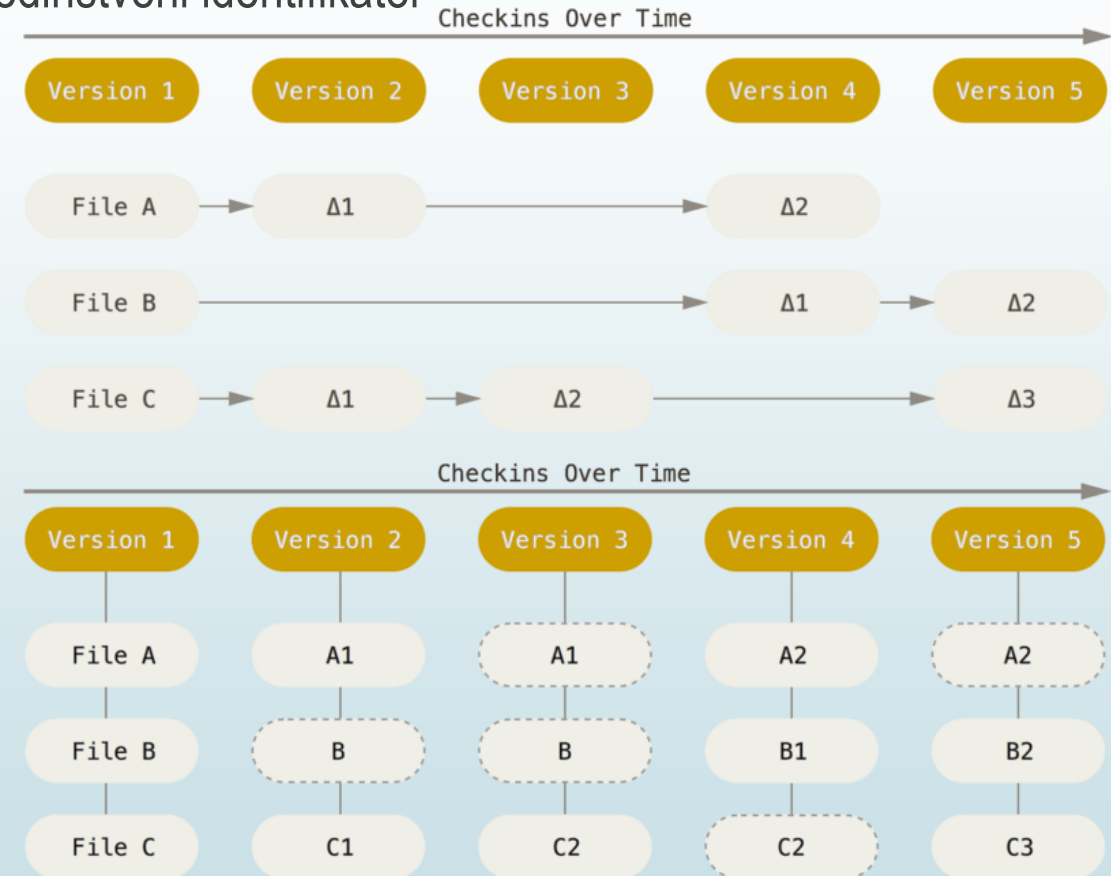
- Najčešće kontrola promjena izvornog koda i konfiguracijskih datoteka
 - ne evidentirati automatski generirane datoteke!
- Verzionirane datoteke
 - datoteke za koje se prati povijest promjena
- Repozitorij
 - sadrži zadnju verziju datoteka i povijest promjena (delte)
 - moguće rekonstruirati bilo koju prethodnu verziju ili usporediti prethodne verzije



Karakteristike sustava za upravljanje verzijama

8

- Identifikacija verzija i izdanja
 - Svaka verzija pohranjena u repozitoriju ima jedinstveni identifikator
- Jedan skup promjena može sadržati promjene na više datoteka
- Kompaktna pohrana
 - Umjesto kopije svake od verzija čuva se zadnja verzija te lista razlika između susjednih verzija



Chacon, Straub, ProGit 2nd Edition, Apress, 2014.

Centralizirani sustavi za upravljanje verzijama

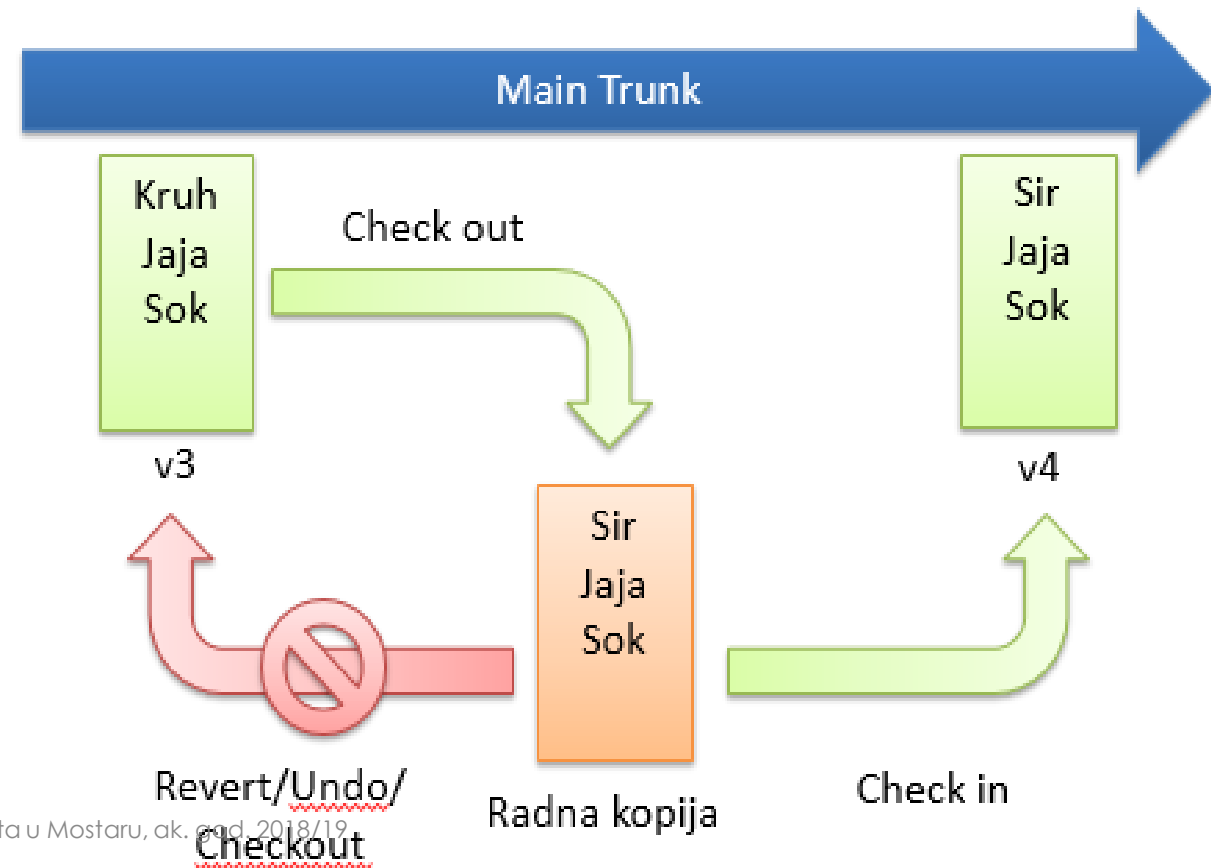
9

- Subversion, CVS, Team Foundation Server, ...
- Samo jedan centralni repozitorij
- Korisnik ima radnu kopiju (working copy, workspace)
 - Za svaku datoteku radna kopija sadrži određenu verziju iz centralnog repozitorija i informaciju o kojoj verziji se radi
 - Neki od alata (npr. TFS) dopuštaju da radna kopija sadrži samo dio centralnog repozitorija
- Promjene na radnoj kopiji započinju checkout-om
 - Najava izmjene datoteke
 - Može uključivati prethodni dohvat zadnje verzije datoteke i onemogućavanje ažuriranja te datoteke drugim korisnicima

Promjene na radnoj kopiji

10

- Korisnik promjene na radnoj kopiji može
 - odbaciti (undo, revert, checkout)
 - potvrditi slanjem na centralni repozitorij (check-in, commit)
- Skup poslanih promjena naziva se *changeset*
- Svaki skup promjena dobiva jedinstveni identifikator
 - Kod centraliziranih sustava identifikator je redni broj koji se povećava za 1



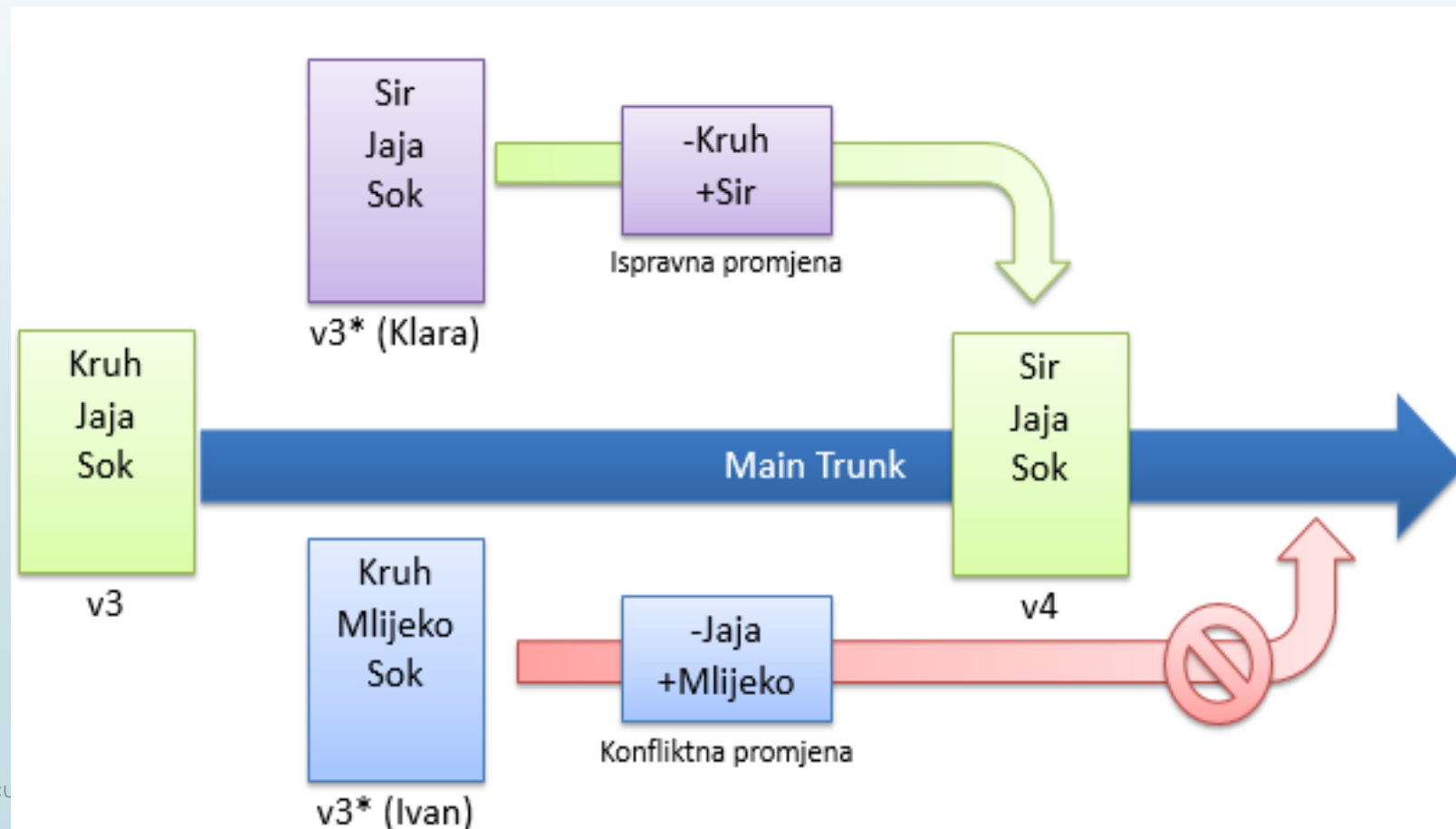
Konfliktne promjene

11

- Istovremene promjene mogu voditi do konflikta
 - Prvi korisnik će uspješno potvrdi promjene
 - Drugom korisniku će sustav dojaviti da su promjene napravljene u odnosu na verziju koja više nije najsvježija

➤ Rješenja

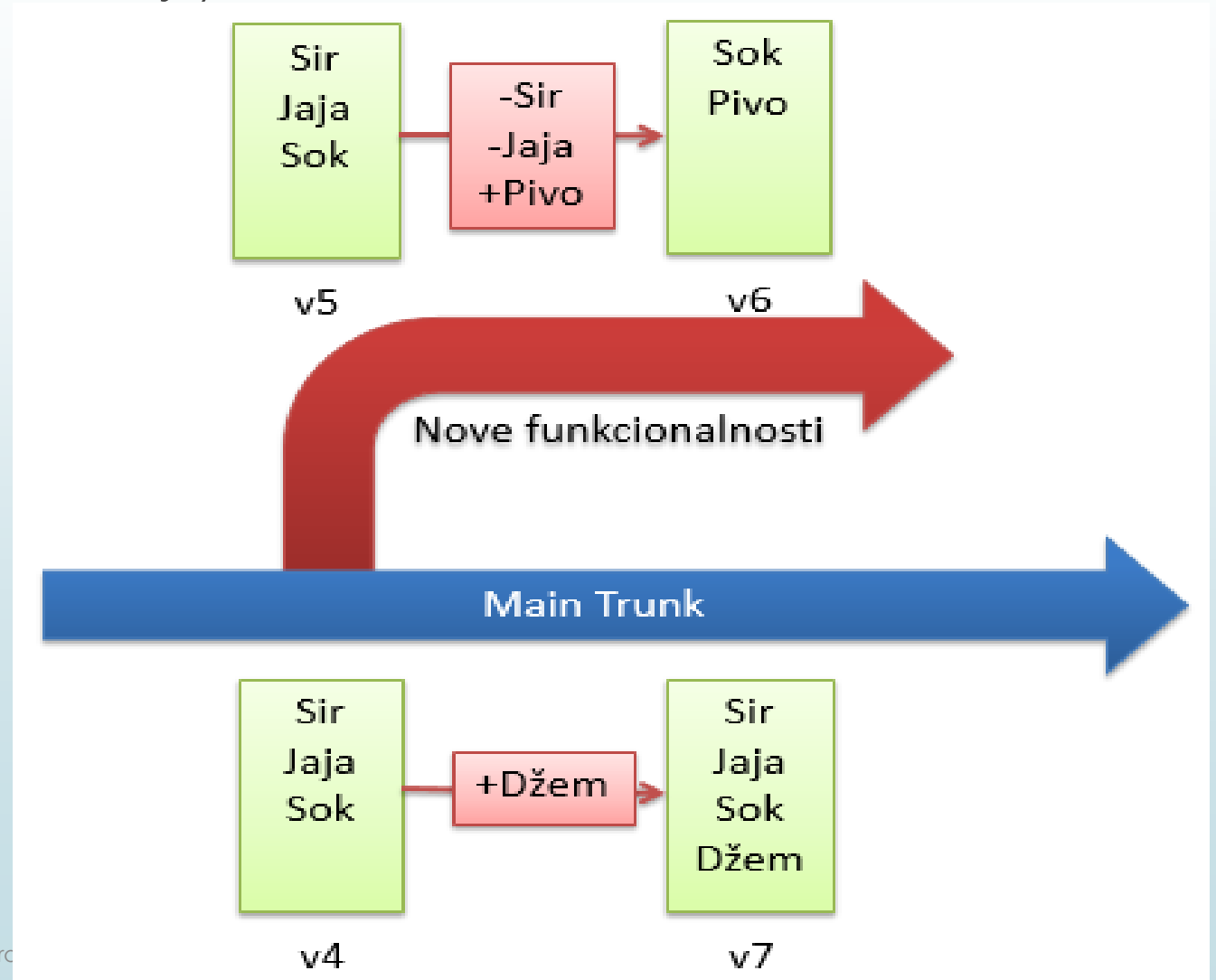
- Odbaciti vlastitu verziju
- Ignorirati verziju sa servera
- Automatsko ili ručno rješavanje preklapanja



Paralelne grane

12

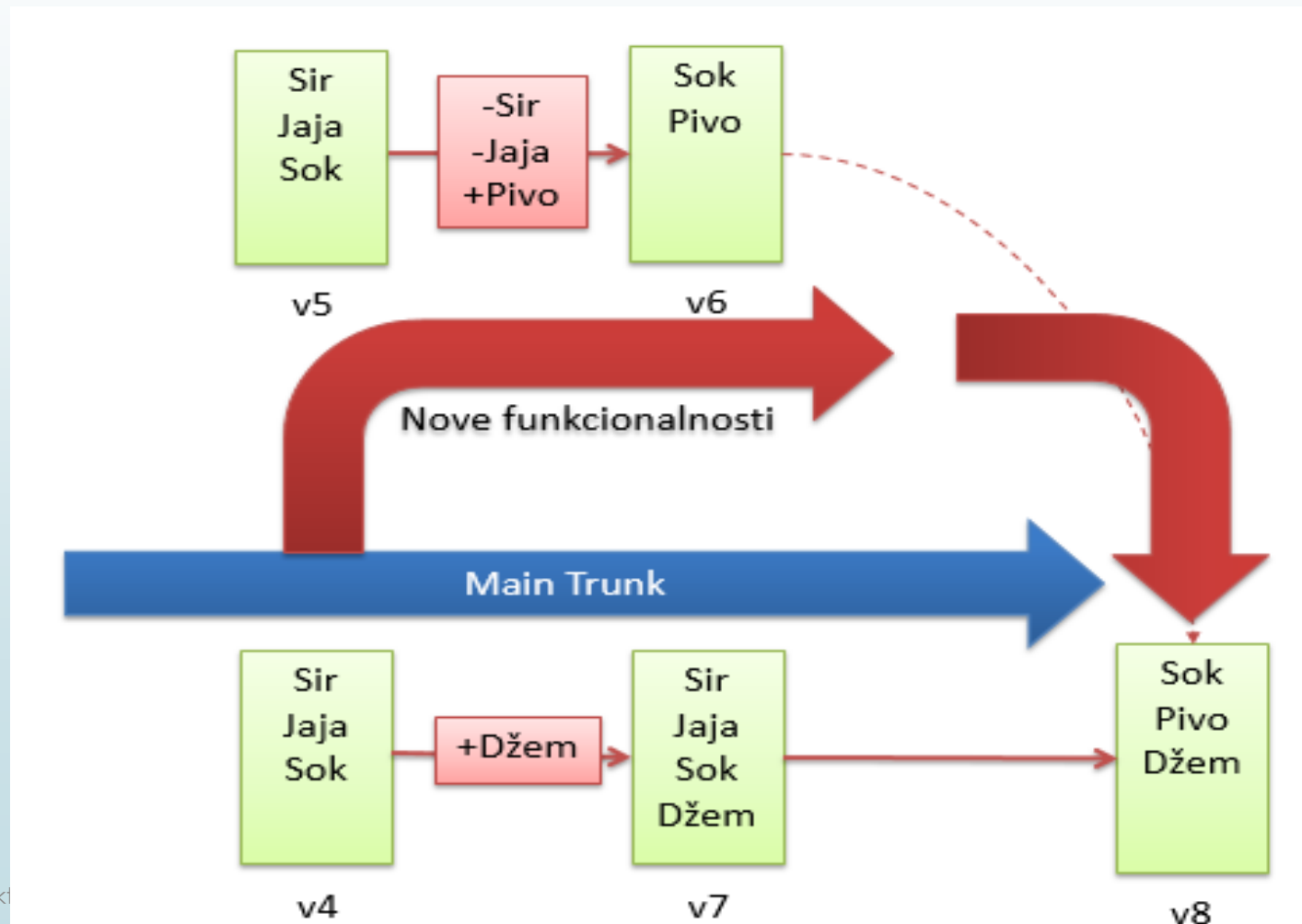
- Omogućava istovremeni rad na novim funkcionalnostima uz očuvanje ispravnosti i održavanje glavne (isporučene verzije).
- Ovisno o sustavu može se raditi o fizički odvojenim kopijama datoteka



Spajanje grana

13

- Nije nužno – paralelna grana ne mora se nikad spojiti natrag
- Ugradnja promjena iz paralelne grane s promjenama u međuvremenu obavljenim na glavnoj grani
 - paralelna grana se može, ali i ne mora obrisati
- Git dodatno omogućava reproduciranje samo dijela promjena iz paralelne grane (cherry-pick)

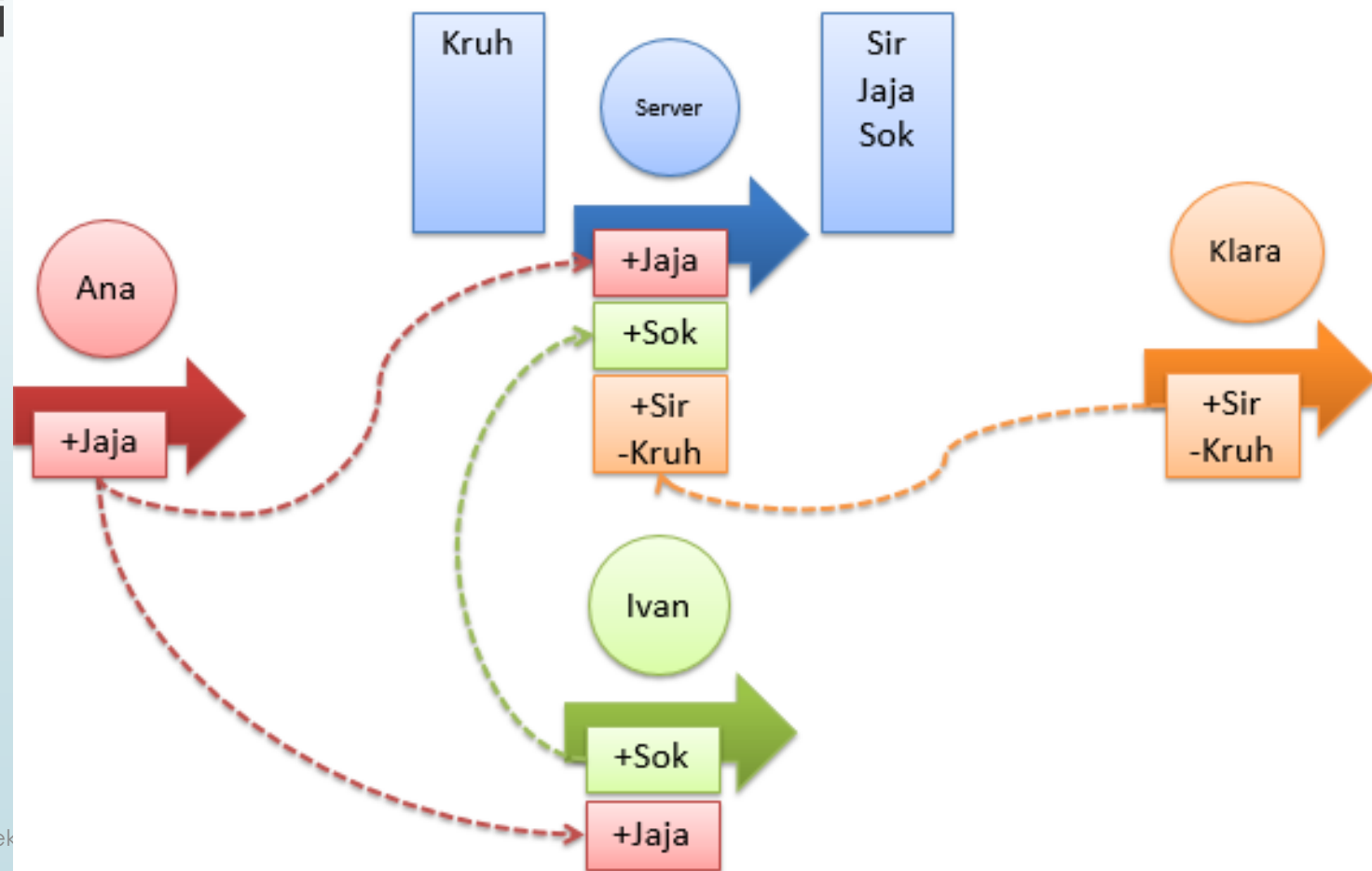


- Git, Mercurial
- Svaki korisnik ima kompletnu kopiju repozitorija
 - sadržaj verzioniranih datoteka, ali i cijela povijest promjena
 - lakše kopiranje repozitorija na neko drugo mjesto
- Korisnik može evidentirati promjene u izvanmrežnom načinu rada
 - naknadno prenosi svoju povijest na centralni server (push)
 - osvježava svoju verziju s verzijom na serveru (fetch i/ili pull)
 - pull = fetch + merge
- Inicijalno stvaranje kopije centralnog repozitorija naziva se kloniranje (engl. clone)
- Napomena: centralni repozitorij ne mora postojati!
- Ostali koncepti akcija i problema (konflikata) su isti ili slični kao kod centraliziranih sustava.

Pojava konflikta u distribuiranim sustavima

15

- Kod distribuirani sustav češće dolazi do konflikta
 - Različiti sadržaji repozitorija kod pojedinih korisnika + neovisne promjene
 - Potrebno spajanje promjena
- Svaka promjena mora imati jedinstveni identifikator
- Redni broj nije dovoljan
 - potrebna složenija oznaka (jedinstveni kod, npr. hash)



Neki od osnovnih pojmova

16

- Kloniranje centralnog repozitorij ili inicijalizacija samostalnog

- `git clone adresa_repozitorija`
- `git init`

- Dohvat zadnje verzije centralnog repozitorija

- `git pull origin naziv grane`
- origin označava udaljeni repozitorij, naziv glavne grane je master

- Datoteke čije se promjene žele evidentirati dodaju se u međuspremnik (Staging Area)

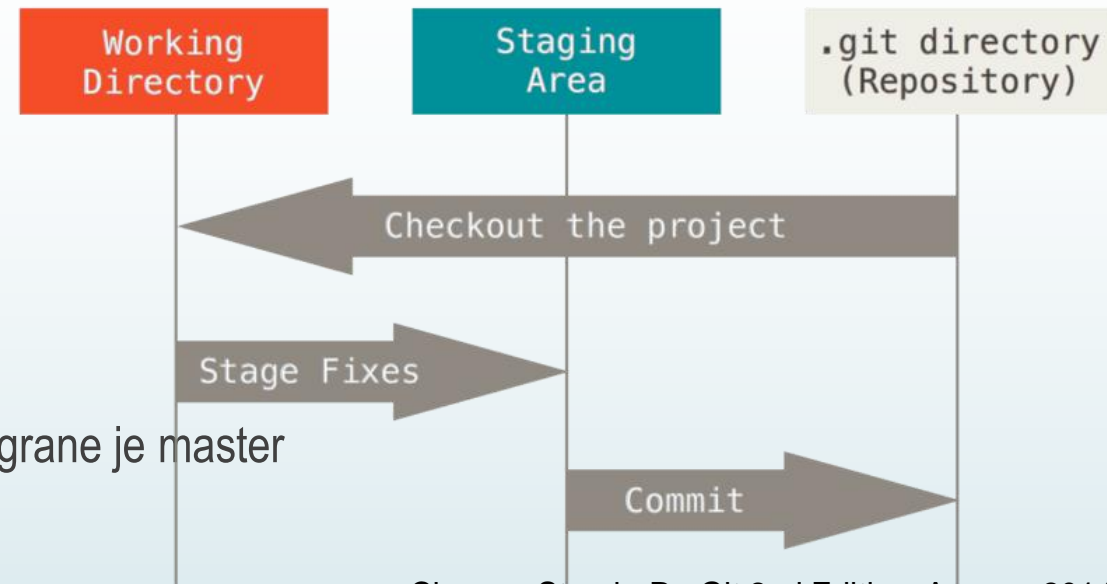
- `git add naziv datoteke ili putanja do više datoteka`

- Korisnik

- potvrđuje `git commit -m "opis promjena"`
- Ili odbacuje `git checkout naziv datoteke ili putanja`

- Korisnik šalje svoje promjene na server

- `git push origin naziv grane`



Chacon, Straub, ProGit 2nd Edition, Apress, 2014.

➤ Stash i Unstash

- Privremeno spremanje izmjena, ali bez potvrđivanja promjena (tj. bez commita)
- Slično kao Shelve/Unshelve kod TFS-a

Literatura za Git

18

- S. Chacon, B. Straub: Pro Git, 2nd Edition 2014. <https://git-scm.com/book/en/v2>
- T. Krajina: Uvod u Git: <https://tkrajina.github.io/uvod-u-git/git.pdf>