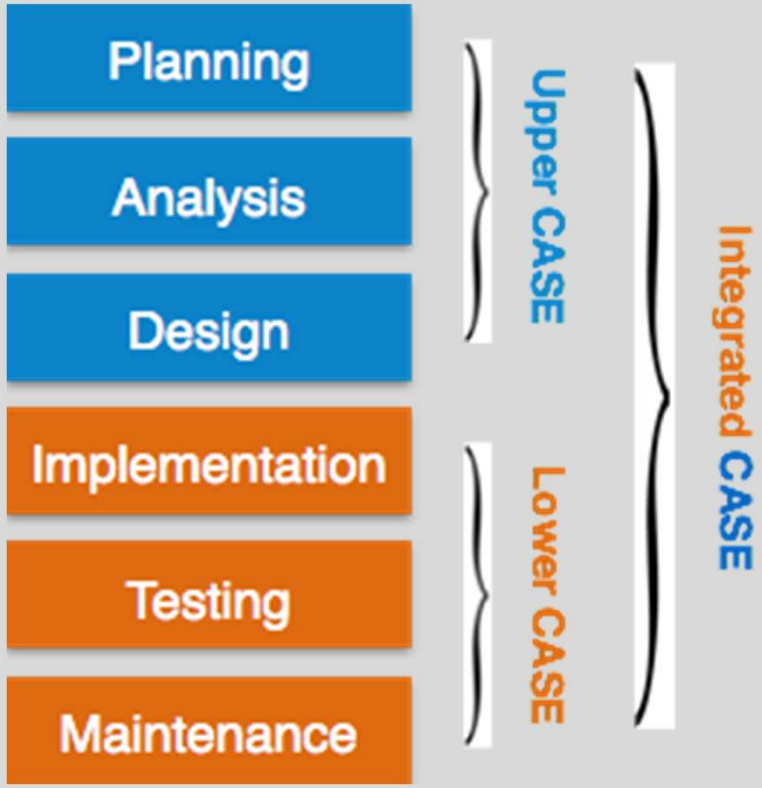
TEHNIČKO VELEUČILIŠTE U ZAGREBU
POLYTECHNICUM ZAGRABIENSE

Objektno orijentirani razvoj programa

ISVU: 130938/19881

Dr. sc. Danko Ivošević, dipl. ing.
Predavač

Akademska godina 2021./2022.
Ljetni semestar



11. CASE-ALATI

Uvod (1)

- **CASE-alati** (engl. *Computer-Aided Software Engineering Tools*) su posebni **računalni alati** koji se koriste u procesu razvoja programske potpore **kako bi automatizirali dio aktivnosti**, **ubrzali izradu**, **pribavili korisne informacije o proizvodu koji se razvija** te **olakšali kasnije održavanje proizvoda**.

Uvod (2)

- Formalna definicija:

- **CASE-alati** su programski proizvodi koji **podupiru proces programskog inženjerstva**, a posebice **aktivnosti specifikacije**, **oblikovanja (modeliranja)**, **implementacije** i **evolucije**.

Prednosti CASE-alata

- Glavne prednosti korištenja ovih alata su:
  1. Mogu se koristiti u svakoj fazi životnog ciklusa programskog proizvoda.
  2. Povećanje kvalitete konačnog proizvoda kroz normiranje načina prikaza i dijeljenja informacija.
  3. Smanjenje vremena i napora potrebnog za izradu programske potpore.
     - Postiže se automatizacijom dijela procesnih aktivnosti (npr. izrada i organizacija dokumentacije) te povećanjem ponovne iskoristivosti (engl. *reusability*) modela i komponenti.

Automatizacija

- **Automatizacija je iznimno važna značajka CASE-alata.**
- CASE podupire automatizaciju oblikovanja raznim alatima kao što su:
  - grafički uređivači za razvoj modela sustava,
  - rječnici i zbirke za upravljanje entitetima u oblikovanju,
  - okruženja za oblikovanje i konstrukciju korisničkih sučelja,
  - alati za pronalaženje pogrešaka u programu,
  - automatizirani prevoditelji koji generiraju nove inačice programa itd.

Nedostaci CASE-alata

- Napredni CASE-alati mogu biti **složeni** do te mjere da **od korisnika zahtijevaju ulaganje značajnog napora u savladavanje korištenja alata**, a niti **cijena alata** (većina naprednih alata je komercijalna) nije zanemariva.
- Osim toga, **ograničavaju kreativnost**.
  - Nastojanja da se kroz alat ostvari što veće normiranje i stupanj automatizacije su često u sukobu s potrebom za kreativnosti i pronalaženjem inovativnih rješenja koje programsko inženjerstvo zahtijeva.

Klasifikacija CASE-alata

- Pri klasifikaciji CASE-alata koriste se tri različite perspektive:
  1. **Funkcijska perspektiva** – alati se klasificiraju prema specifičnoj funkciji koju obavljaju.
  2. **Procesna perspektiva** – alati se klasificiraju prema aktivnostima koje podupiru u procesu.
  3. **Integracijska perspektiva** – alati se klasificiraju prema njihovoj organizaciji u integrirane cjeline.

Funkcijska perspektiva (1)

- Vrste CASE-alata prema funkcijskom pogledu:
  - Vrsta: Alati za planiranje (engl. *planning tools*)
  - Primjer: PERT alati, tablični kalkulatori (npr. Excel)

  - Vrsta : Alati za uređivanje (engl. *editing tools*)
  - Primjer: Uređivači teksta, dijagrama (npr. Notepad, Word, Visio)

  - Vrsta : Alati za upravljanje promjenama (engl. *change management tools*)
  - Primjer: Sustavi za praćenje promjena u zahtjevima i proizvodu (npr. IBM Rational DOORS, Borland Caliber)

Funkcijska perspektiva (2)

- Vrste CASE-alata prema funkcijskom pogledu:
  - Vrsta: Alati za upravljanje konfiguracijom (engl. *configuration management tools*)
  - Primjer: Sustavi za kontrolu i upravljanje inačicama (npr. Subversion, Git)

  - Vrsta: Alati za izradu prototipa (engl. *prototyping tools*)
  - Primjer: Jezici vrlo visoke razine apstrakcije (npr. UML)

  - Vrsta: Alati za potporu metodama (engl. *method-support tools*)
  - Primjer: Različiti podatkovni rječnici, generatori koda

Funkcijska perspektiva (3)

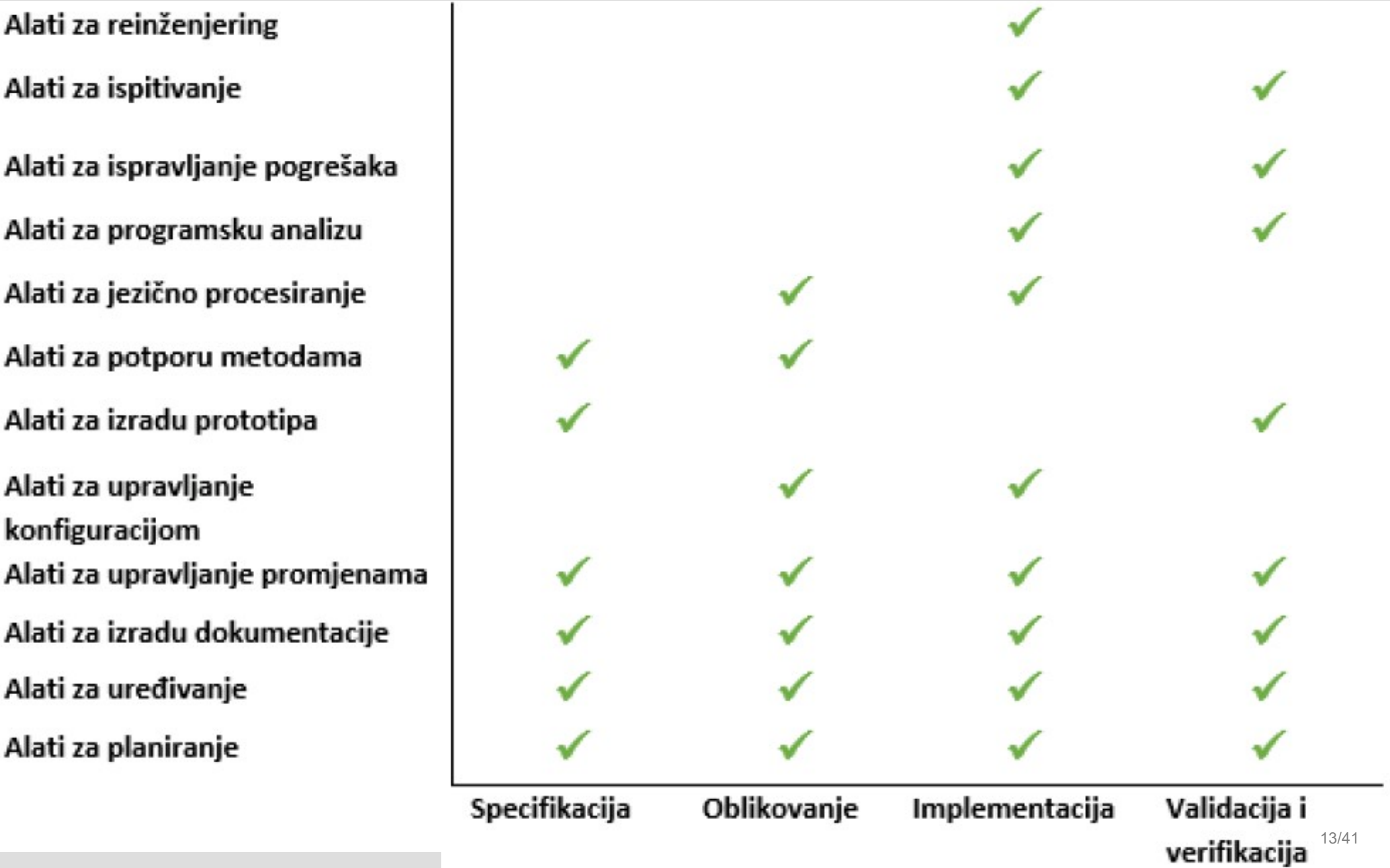
- Vrste CASE-alata prema funkcijskom pogledu:
  - Vrsta: Alati za jezično procesiranje (engl. *language-processing tools*)
  - Primjer: Kompajleri, interpreteri
  - Vrsta: Alati za programsku analizu (engl. *program analysis tools*)
  - Primjer: Različiti alati za analizu statičkih i dinamičkih performansi programa
  - Vrsta: Alati za ispitivanje (engl. *testing tools*)
  - Primjer: Generatori ispitnih skupova

Funkcijska perspektiva (4)

- Vrste CASE-alata prema funkcijskom pogledu:
  - Vrsta: Alati za ispravljanje pogrešaka (engl. *debugging tools*)
  - Primjer: Alati ugrađeni u razvojne okoline (npr. Visual Studio, Eclipse...)
  - Vrsta: Alati za izradu dokumentacije (engl. *documentation tools*)
  - Primjer: Različiti alati za prijelom i uređivanje slika
  - Vrsta: Alati za reinženjering (engl. *re-engineering tools*)
  - Primjer: Posebni alati za unakrsnu usporedbu dijelova sustava i restrukturiranje programa

Procesna perspektiva

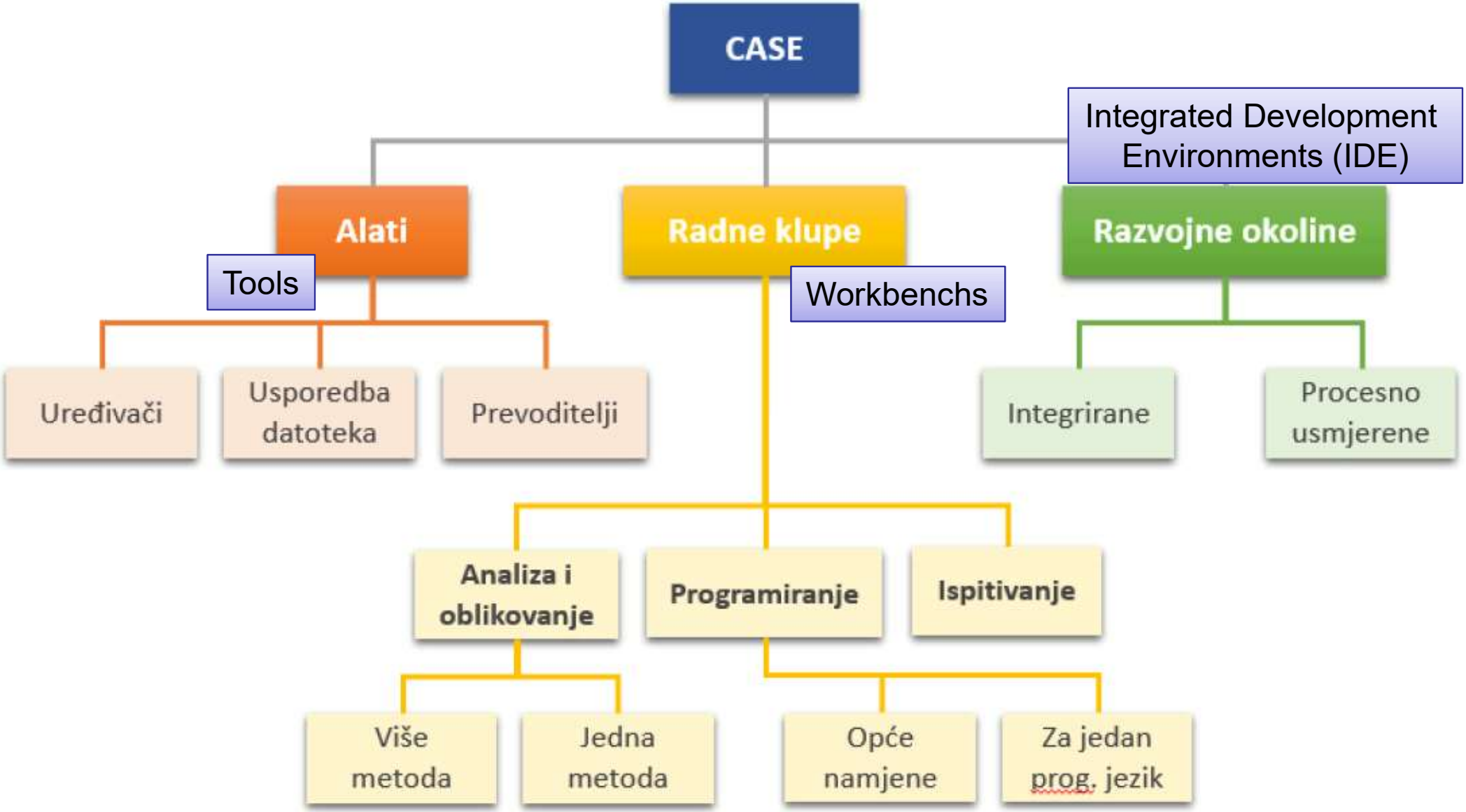
S obzirom na generičke aktivnosti procesa programskog inženjerstva



Integracijska perspektiva

- **Ova perspektiva promatra alate s obzirom na stupanj integracije alata u cjelinu.**
- Alati (u užem smislu) podupiru individualne zadatke u procesu (npr. oblikovanje, provjeru dosljednosti zahtjeva, uređivanje teksta, …).
- **Dvije kategorije alata prema ovoj perspektivi:**
  1. **Viši CASE, upper-CASE alati** (*front-end*), koji se koriste u raniji fazama kao što su izlučivanje zahtjeva, analize i modeliranja
  2. **Niži CASE, lower-CASE alati** (*back-end*), koji se koriste u kasnijim fazama implementacije, ispitivanja i održavanja.

Integracijska klasifikacija CASE-alata



Radne klupe

- **Radne klupe (engl. *workbenches*)** podupiru pojedine aktivnosti (faze) procesa (npr. specifikaciju).
- Radne klupe objedinjuju više različitih alata za potporu u nekoj fazi procesa programskog inženjerstva.
- **Najčešće podržavaju jednu od tri aktivnosti: 1) analizu i oblikovanje, 2) programiranje te 3) ispitivanje.**

Razvojne okoline

- **Razvojne okoline (engl. *environments*) podupiru cijeli ili značajan dio procesa programskog inženjerstva.**
- **Uključuju nekoliko integriranih radnih klupa.**

- Primjeri razvojnih okolina:
  - Visual Studio, Eclipse, IntelliJ, NetBeans, …

Primjeri CASE-alata i radnih klupa

- CASE-alati i radne klupe:
  - Subversion, Git – upravljanje konfiguracijama programske potpore.
  - ArgoUML, Astah Community Edition, Enterprise Architect – oblikovanje zahtjeva, oblikovanje modela objektno usmjerene arhitekture.
  - Microsoft Visio - oblikovanje zahtjeva, oblikovanje modela objektno usmjerene arhitekture.
  - Microsoft Visual Studio – implementacija programske potpore vezana uz Microsoftove tehnologije
  - Eclipse – implementacija programske potpore vezana uz Java tehnologije (i druge).

Upravljanje izvornim kodom (1)

- Prilikom razvoja programske potpore nezaobilazna je međusobna suradnja većeg broja osoba i timova koji se često nalaze na razdvojenim lokacijama (npr. različitim državama i vremenskim zonama), no rade nad istim datotekama.
- Kontrola inačica datoteka i vođenje evidencije o promjenama koje su nastale je potpuna nužnost!

Upravljanje izvornim kodom (2)

- Formalna definicija:

- **U programskom inženjerstvu, kontrola inačica programske potpore je svaki postupak koji prati i omogućava upravljanje promjenama nastalima u datotekama s izvornim kodom ili dokumentacijom.**

Važni pojmovi

- **Revizija** je osnovni pojam kojim se opisuje slijed razvoja.
- Jedinstveni slijed razvoja u kojem nema grananja naziva se **osnovna razvojna linija** (engl. *trunk*).
- Ako postoji potreba za razvojem dodatnih značajki programske potpore mimo osnovne linije, tada dolazi do razdvajanja razvoja u dvije ili više **grana** pri čemu svaku granu još nazivamo **pomoćnom razvojnom linijom** (engl. *branch*).

Važni pojmovi – Vršna revizija

- Kad nema grananja svaka revizija temelji se isključivo na jednoj jedinoj reviziji koja joj neposredno prethodi te **sve revizije čine jednu liniju**.
- U takvom nizu postoji jedinstvena zadnja inačica koja se često naziva **vršna revizija** (engl. *head*).

Važni pojmovi – Graf revizija

- Ako postoji grananje, iz jedne revizije može nastati nekoliko novih, a također je moguće da se nova revizija ne temelji na neposrednoj prethodnici nego na nekoj ranijoj reviziji. U takvom slučaju **graf revizija** umjesto linijskog poprima stablasti oblik te se vršna revizija mora izričito zadati.

Važni pojmovi – Spajanje linija

- Proces koji dovodi do spajanja revizija iz dviju ili više razvojnih linija u jedinstvenu reviziju naziva se **spajanje linija** (engl. *merge*). U praksi je ovaj proces iznimno teško izvesti i predstavlja jedan od najsloženijih aspekata kontrole inačica.

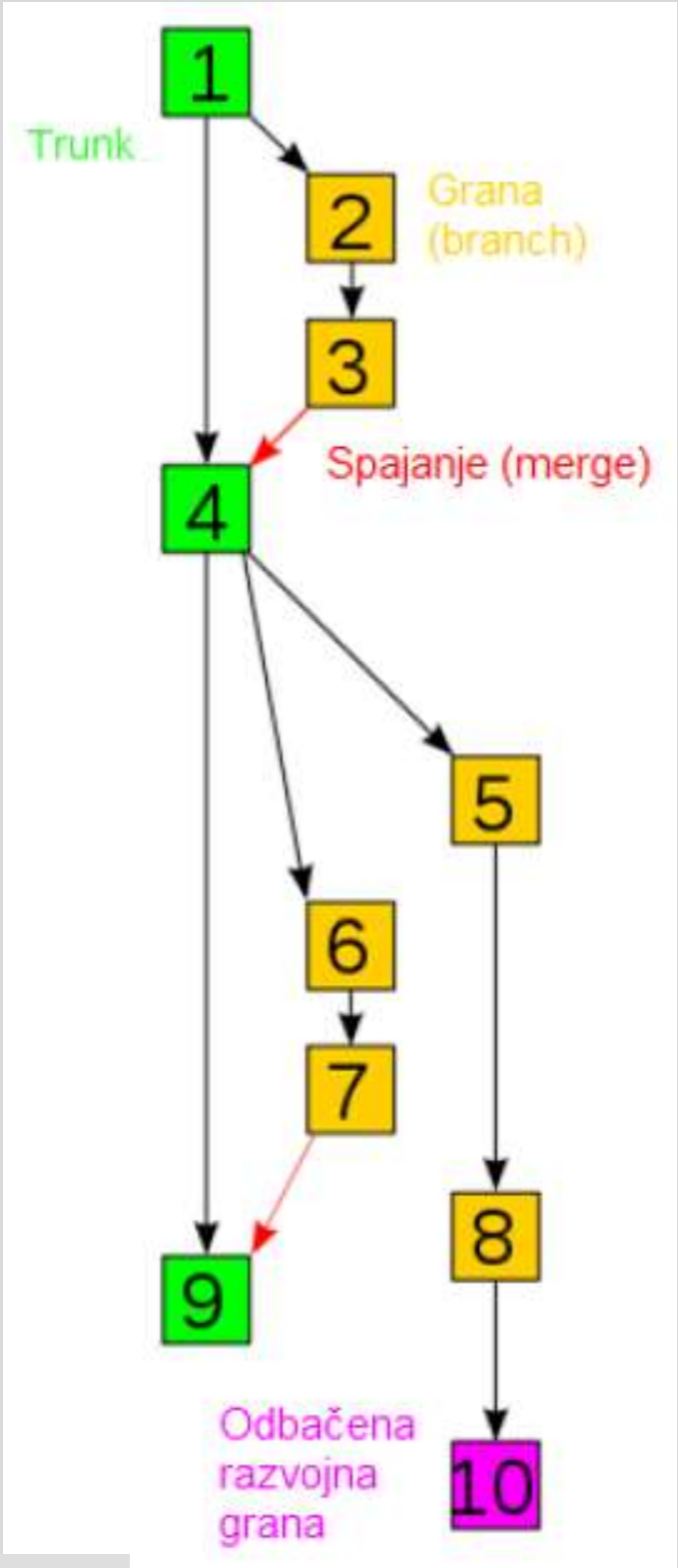
Najčešće korišteni pojmovi u sustavima za kontrolu inačica (1)

Pojam	Definicija
Branch (pomoćna razvojna linija)	Skup datoteka obuhvaćen sustavom kontrole inačica koji se u određenom trenutku odvaja od osnovne linije razvoja I dalje razvija zasebno i neovisno o ostalim pomoćnim linijama razvoja.
Checkout (dohvaćanje)	Proces stvaranja lokalne radne kopije repozitorija. Korisnik može dohvatiti vršnu reviziju ili može odabrati bilo koju dostupnu reviziju. Ovaj pojam se ponekad koristi i kao imenica koja označava samu radnu kopiju repozitorija.
Commit (provedba)	Proces zapisivanja promjena iz radne inačice natrag u repozitorij. Koristi se još i izraz *check-in*.
Conflict (sukob, spor)	Sukob koji nastaje kad dvoje ili više korisnika napravi promjene u istom dokumentu te sustav ne može automatski objediniti promjene u novu inačicu. Tada jedan od korisnika mora riješiti spor (*resolve*) tako što će sam uklopiti različite promjene ili odbaciti sve osim jedne inačice nove inačice.
Export (izvoz)	Proces sličan *checkout*-u s razlikom u tome da se u strukturi kazala ne nalaze i datoteke s meta-podacima potrebnim za kontrolu inačica.

Najčešće korišteni pojmovi u sustavima za kontrolu inačica (2)

Pojam	Definicija
Head (vršna inačica)	Najnovija inačica u svakoj grani. *Head* se upotrebljava za najnoviju glavnu inačicu.
Import (uvoz)	Proces uvoza lokalne strukture kazala u središnji repozitorij po prvi put (nije radna kopija).
Merge (spajanje)	Spajanje dviju ili više inačica skupa datoteka u jedinstvenu inačicu.
Repository (središnji repozitorij)	Mjesto (na poslužitelju) na kojem se nalaze sve datoteke i popratni meta-podaci.
Resolve (razrješavanje)	Razrješavanje sukoba koji nastaje kada više korisnika istovremeno pokuša unijeti promjene u isti dokument.
Trunk (osnovna razvojna linija)	Osnovna ili glavna linija razvoja.
Update (osvježavanje)	Osvježavanje lokalne radne kopije s promjenama koje su nastale u središnjem repozitoriju.

Primjer grafa kontrole inačica s osnovnom linijom i grananjima.



Vrste sustava za kontrolu inačica programske potpore

- Danas postoji mnogo različitih sustava za kontrolu inačica programske potpore.
- Najčešće korišteni:
  - Apache Subversion/Tortoise SVN
  - Git

- Nisu isti. Osim što se razlikuju po tipu licence – komercijalni i sustavi otvorenog koda, prije svega se razlikuju po način rada: lokalnosti pristupa te centraliziranosti, tj. postojanju središnjeg repozitorija.

Lokalizirani model

- **Lokalizirani model** (engl. *Local data model*) osnovni je tip sustava za kontrolu inačica.
- Pohrana i pristup podacima ograničeni su na jedno računalo na kojem su također sadržani podaci o ranijim promjenama.
- Promjene se prate za svaku datoteku zasebno, što znači da nije moguće odjednom raditi s cijelim skupom datoteka (npr. projektom). Kao dio GNU projekta razvijen je RCS sustav (*Revision Control System*) koji je kasnije poslužio kao okosnica za razvoj naprednijih sustava poput CVS-a i SVN-a.

Model klijent-poslužitelj

- **Model klijent-poslužitelj** podrazumijeva postojanje središnjeg repozitorija na poslužiteljskom računalu čije radne inačice korisnici pohranjuju na svojim računalima i u određenom trenutku sinkroniziraju sa središnjim repozitorijem.
- Jedan od prvih sustava otvorenog koda koji je implementirao ovaj model bio je CVS (*Concurrent Versions System*). Na temelju njega se kasnije razvio Apache SVN koji se i danas koristi.

Apache Subversion

- Apache Subversion (SVN) je jedan od najpoznatnijih sustava za kontrolu inačica programske potpore.
- Nasljednik je ranijeg CVS sustava, a najznačajnija unaprjeđenja su u vidu potpune atomarnosti operacije *commit*, zaključavanja datoteka u slučaju da više korisnika pokuša istovremeno raditi izmjene, dohvaćanje *readonly* inačice središnjeg repozitorija i dr.

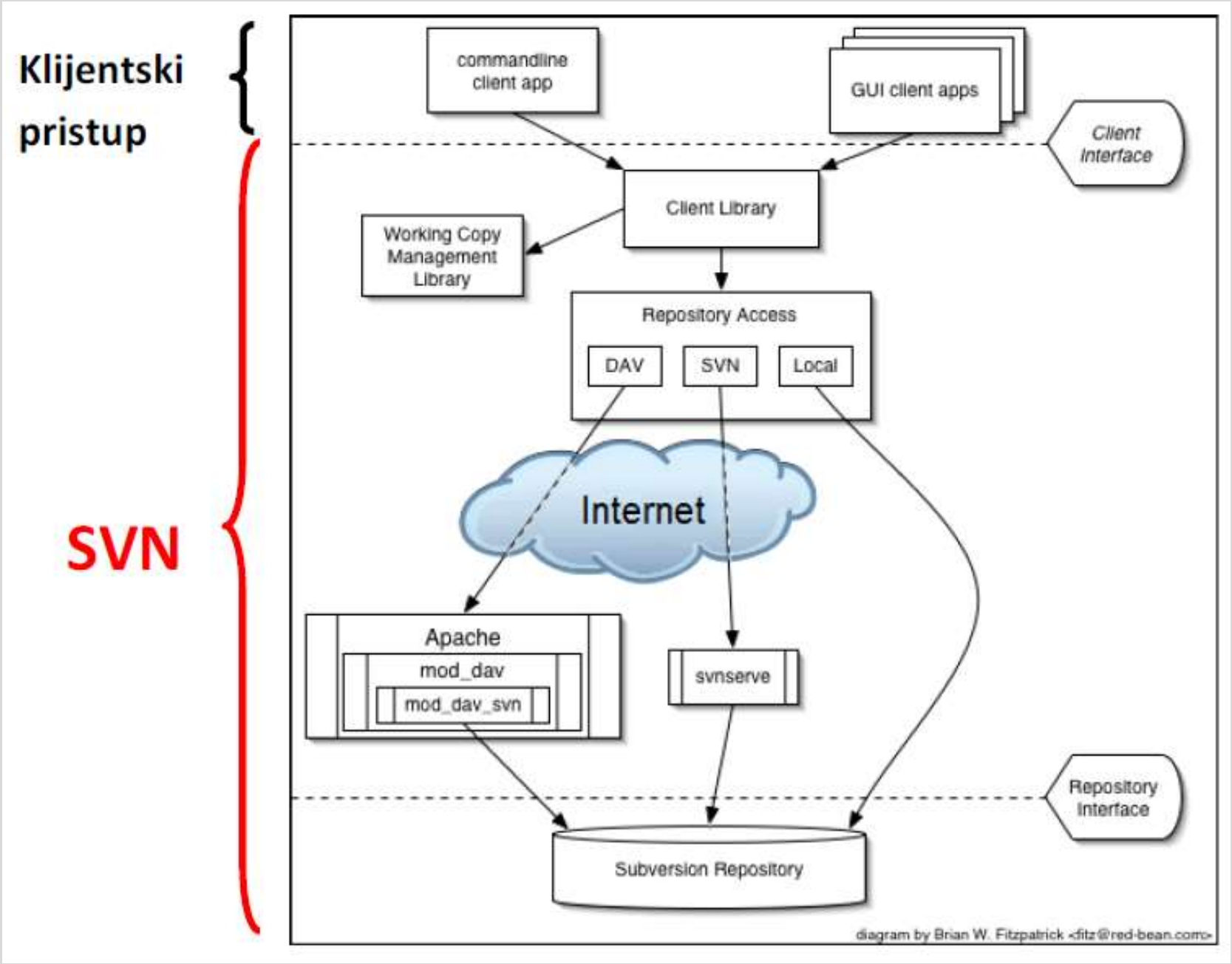
Najčešće korištene operacije

- Prilikom korištenja SVN-sustava, dvije najčešće korištene operacije su ***Update***, ***Commit*** i ***Checkout***.
- ***Update*** **mijenja lokalnu radnu kopiju kako bi bila identična stanju središnjeg repozitorija**.
- ***Commit*** **izvozi u središnji repozitorij promjene nastale u lokalnoj radnoj kopiji.**
- Prilikom stvaranja lokalne radne kopije (**prvo dohvaćanje sadržaja središnjeg repozitorija**) koristi se operacija ***Checkout***.

Operacije *Revert* i *Resolve*

- U slučaju sukoba inačica na kojima je istovremeno radilo više korisnika, sukob se može riješiti **odbacivanjem vlastitih promjena** – operacija ***Revert*** – ili se ide u **razrješavanje spora** – operacija ***Resolve*** – u kojem se odabire koja će se inačica zadržati (base, mine, full, working).

Arhitektura SVN-a



SVN klijenti

- Budući da Apache SVN podržava isključivo naredbeno-linijski način rada, razvijeni su posebni klijenti poput Tortoise SVN-a koji imaju integrirano grafičko korisničko sučelje.
  - URL: http://tortoisesvn.net/

Raspodijeljeni način rada

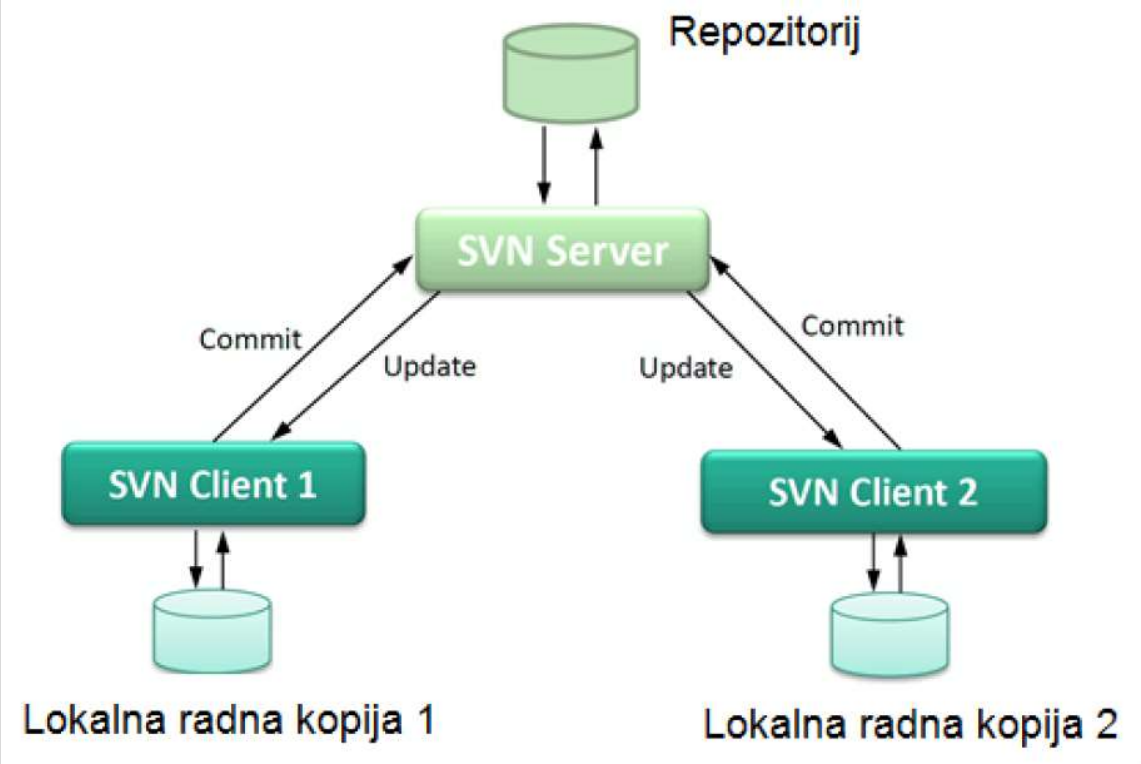
- U **raspodijeljenom modelu** (engl. *distributed model*) **ne postoji jedinstveni središnji repozitorij s kojim se klijenti sinkroniziraju. Svaka radna kopija repozitorija predstavlja repozitorij za sebe.**
- Tada su svi postojeći repozitoriji su ravnopravni, a korisnici mogu međusobno usklađivati repozitorije prema načelu ***peer-to-peer***.
- Glavni predstavnici ovakvog modela su sustavi otvorenog koda Git i Mercurial.

Git

- Git je danas vjerojatno najraširenij sustav otvorenog koda za kontrolu inačica programske potpore.
- U odnosu na SVN znatno je složeniji za savladavanje i korištenje, ali nudi dodatne funkcionalnosti.
- Git-ova raspodijeljena arhitektura omogućava istovremeno postojanje više smjerova razvoja programske potpore koji se mogu, ali i ne moraju objediniti.

Arhitektura Git-a

- Arhitektura sustava Git podrazumijeva rad s dva repozitorija – lokalnim i udaljenim.
- Rad s lokalnim repozitorijem nalikuje na rad sa središnjim repozitorijem u SVN-u.



Najvažnije operacije

- Najvažnije operacije Git-a:
  - operacija *Commit* zapisuje promjene u lokalni repozitorij.
  - operacija *Checkout* osvježava radnu kopiju lokalnog direktorija (operacija *Update* pod tim imenom ne postoji).
  - Lokalni repozitorij usklađuje se s udaljenim repozitorijem operacijama *Push* i *Fetch*
  - Ako se želi direktno osvježiti lokalna radna kopija umjesto *Fetch* + *Merge* može se direktno izvršiti operacija *Pull/Rebase*.
  - Prilikom prvog dohvaćanja sadržaja udaljenog direktorija koristi se operacija *Clone*.

Kvazi-repozitorij Index

- Dodatna mogućnost koju nudi Git je Index – lokalni kvazi-repozitorij u koji se pohranjuju nove i izmijenjene datoteke prije sinkronizacije s pravim lokalnim repozitorijem (operacije *Add* i *Checkout*).

Sustav *Git*

- Repozitorij – mjesto pohrane promjena datoteka
- Raspodijeljeni razvoj – na više mjesta
- Svaka mapa/kazalo – potpuni repozitorij
- → potpuni datotečni sustav
- Najviše korišteni alat za upravljanje inačicama datoteka
- Klijenti za standardna razvojna okruženja
- Različita ostvarenja *Git* poslužitelja:
  - → *GitLab* – web hosting usluga za *Git* temeljene projekte

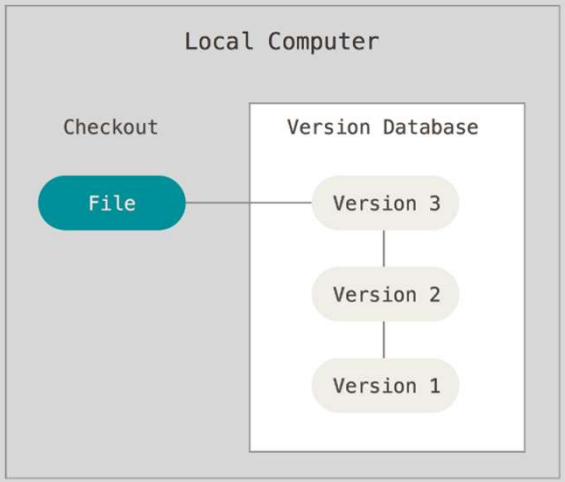
Sustav *Git* + *GitLab*

- Instalacija i dokumentacija *Git*-a dostupne na:
  - https://git-scm.com
- *GitLab* pristup na:
  - https://gitlab.com

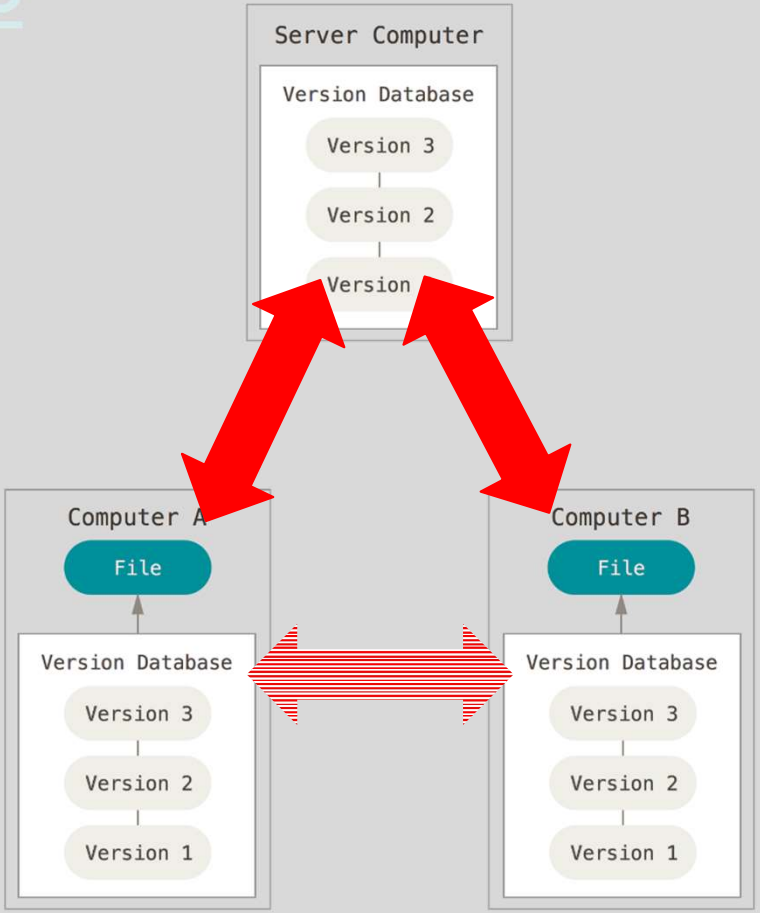
Sustav *Git* - Ideja

- https://git-scm.com/doc

Lokalno upravljanje:

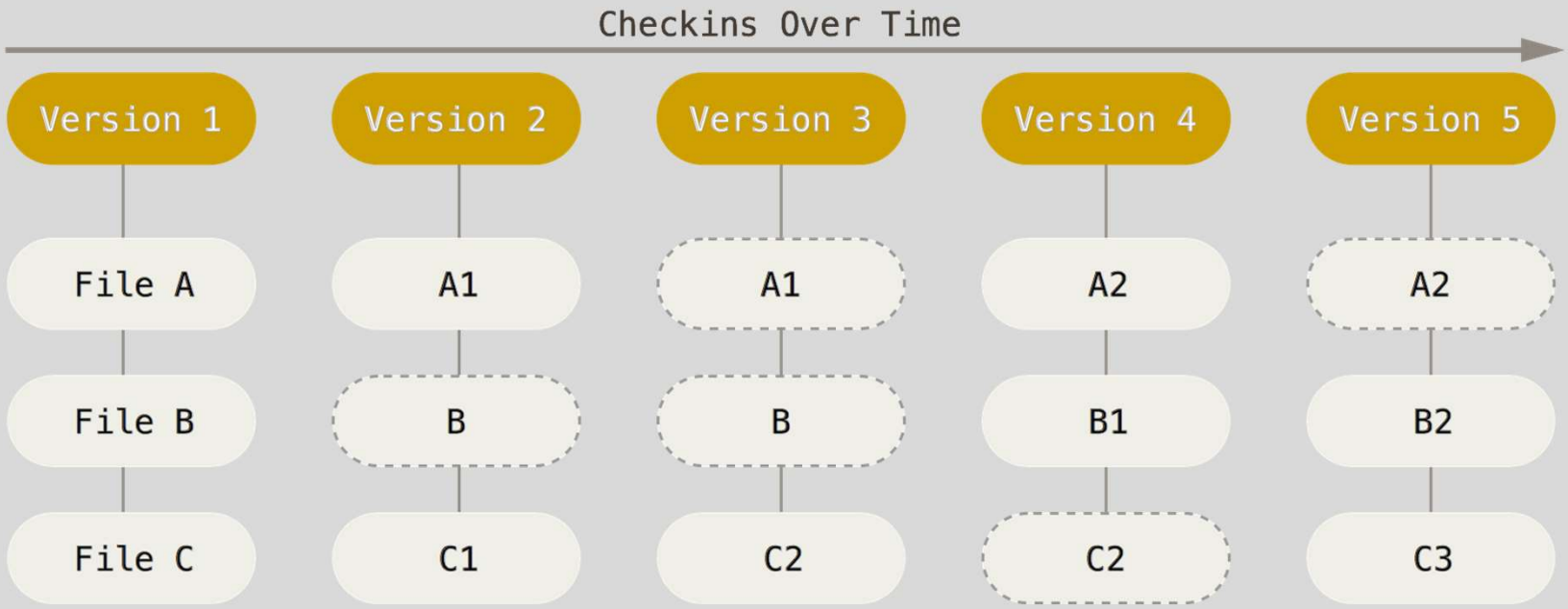


Raspodijeljeno upravljanje:



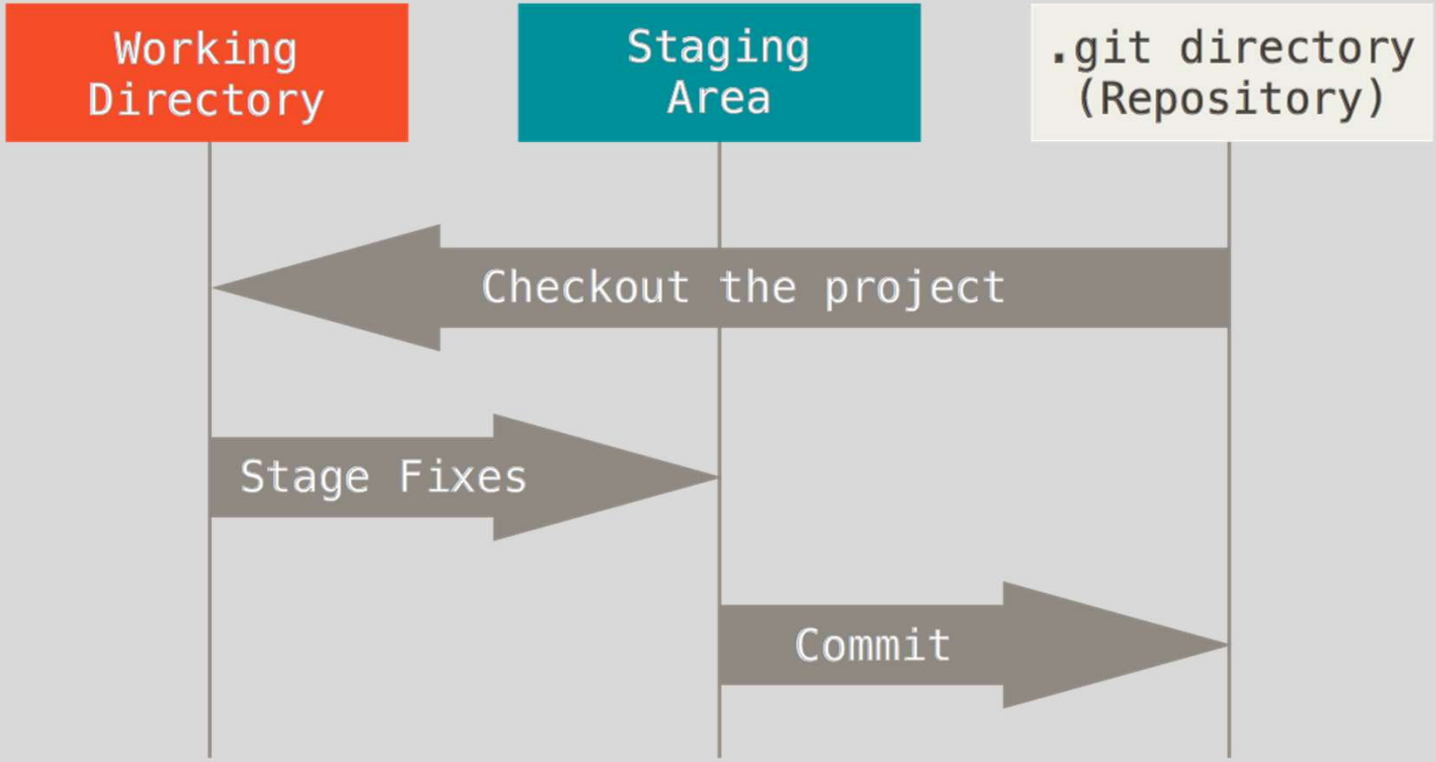
Sustav *Git* - Stanja

- https://git-scm.com/doc

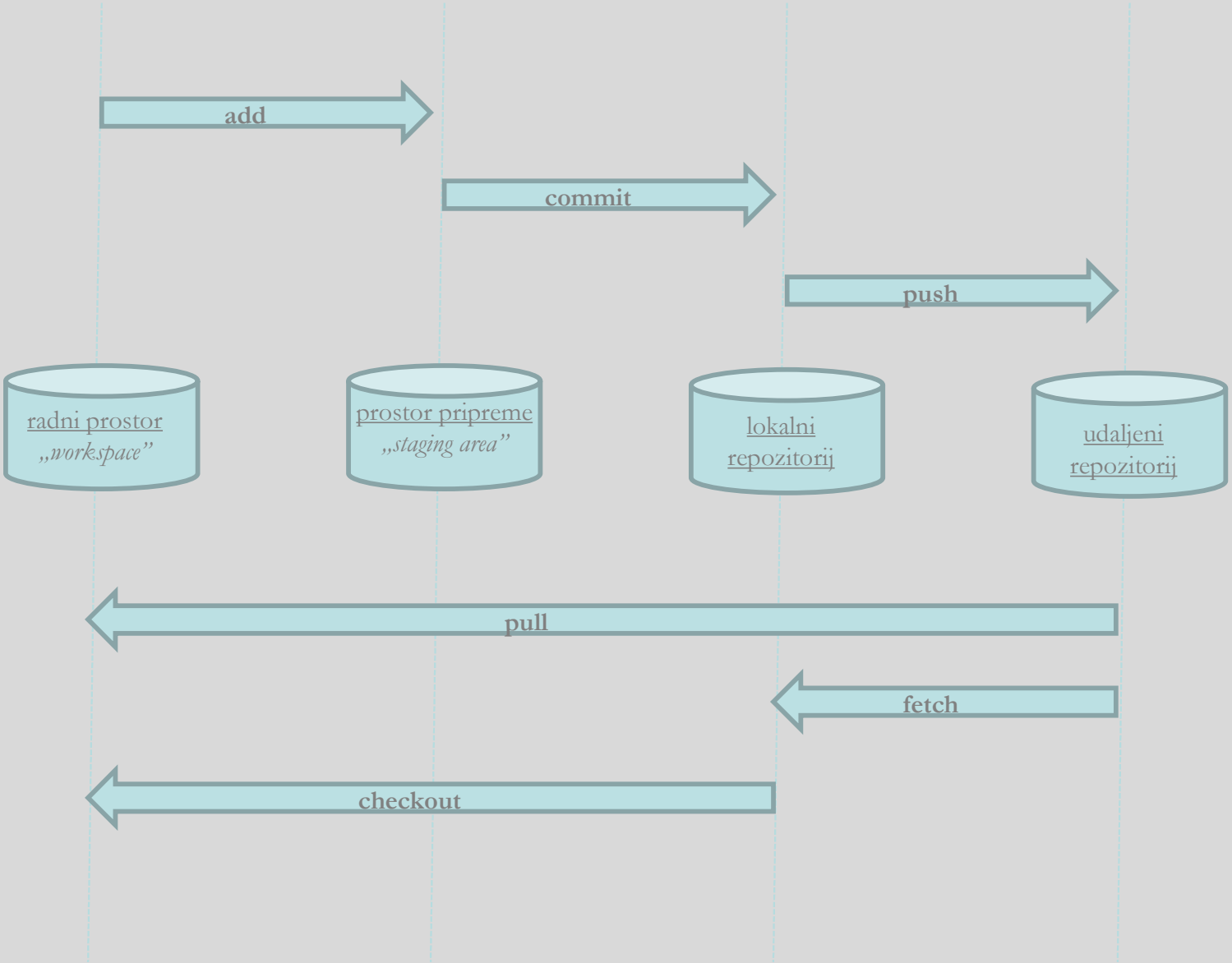


Sustav *Git* – Lokalni repozitorij

- https://git-scm.com/doc

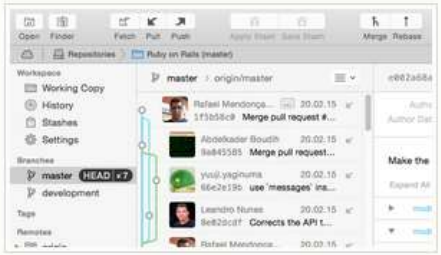


Sustav *Git* – Središnji repozitorij

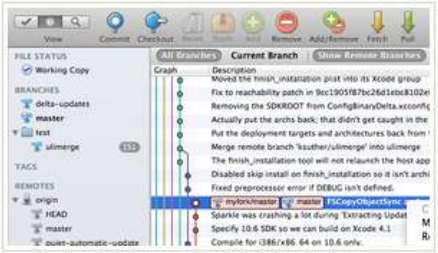


Git GUI klijenti

GitHub for Mac
Platforms: Mac
Price: Free

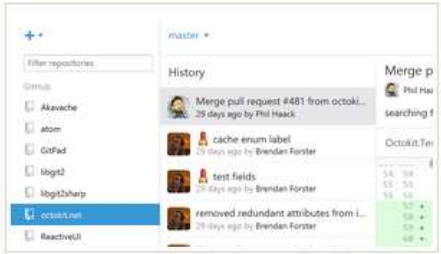


Tower
Platforms: Mac
Price: $69/user (Free 30 day trial)

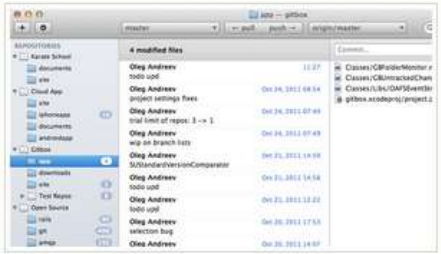


SourceTree
Platforms: Mac, Windows
Price: Free

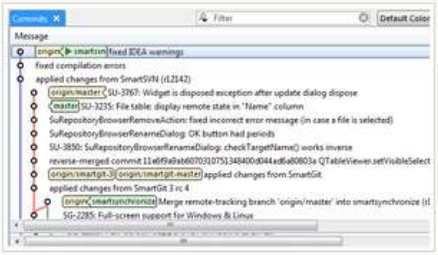
git-cola
Platforms: Windows, Mac, Linux
Price: Free



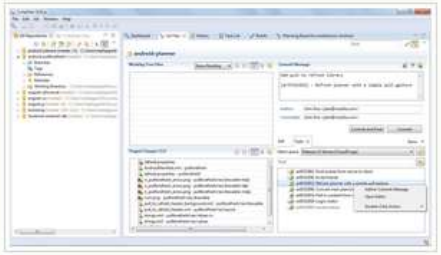
GitHub for Windows
Platforms: Windows
Price: Free



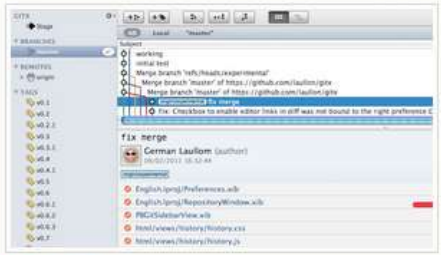
Gitbox
Platforms: Mac
Price: $14.99



SmartGit
Platforms: Windows, Mac, Linux
Price: $79/user / Free for non-commercial use

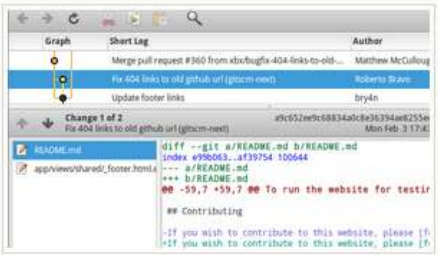


GitEye
Platforms: Windows, Mac, Linux
Price: Free

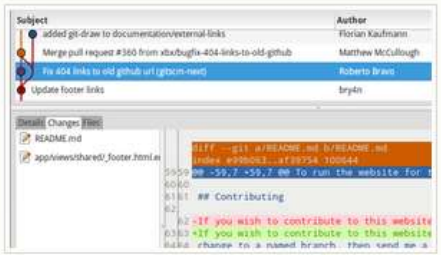


GitX-dev
Platforms: Mac
Price: Free

Git Extensions
Platforms: Windows
Price: Free

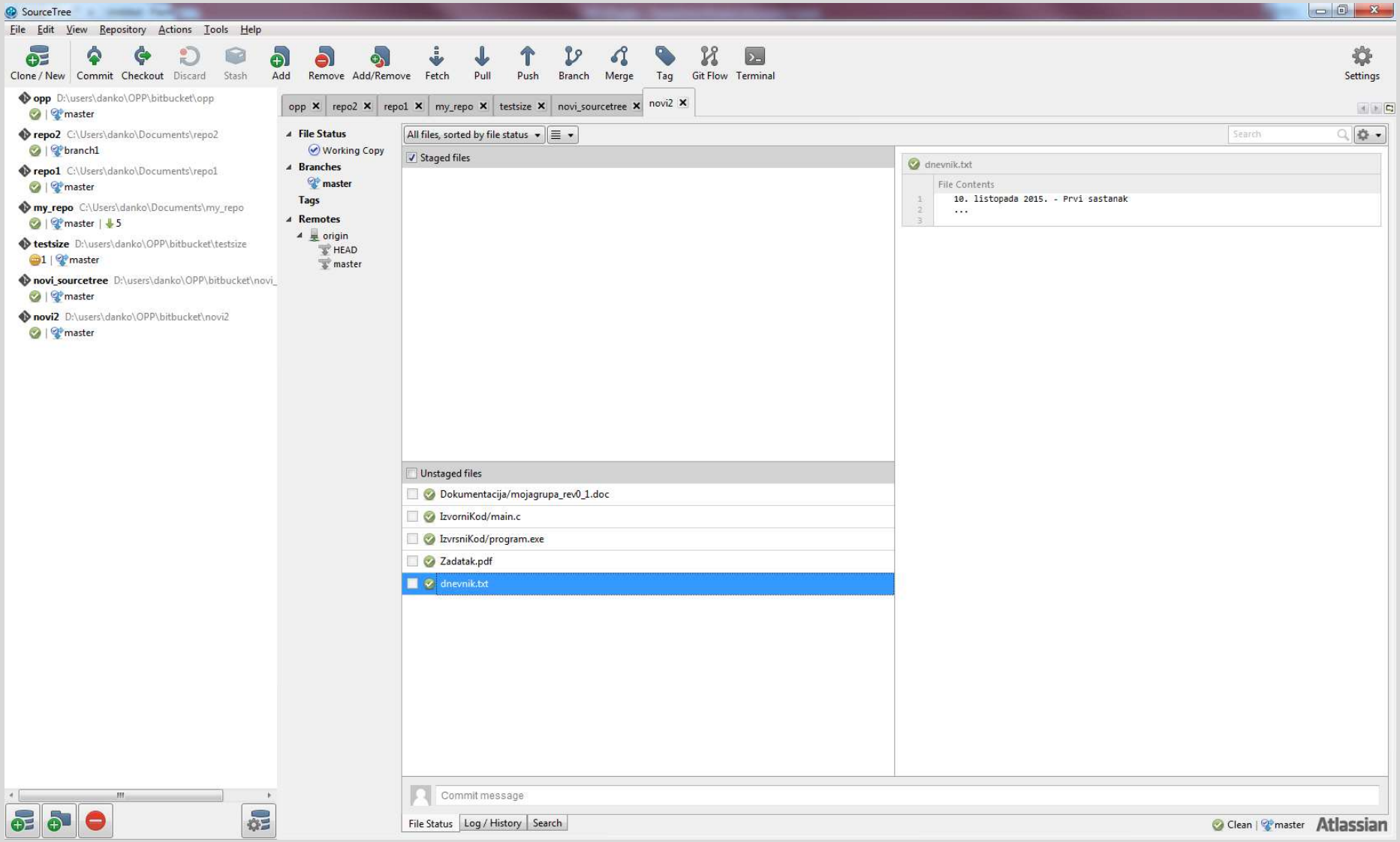


giggle
Platforms: Linux
Price: Free

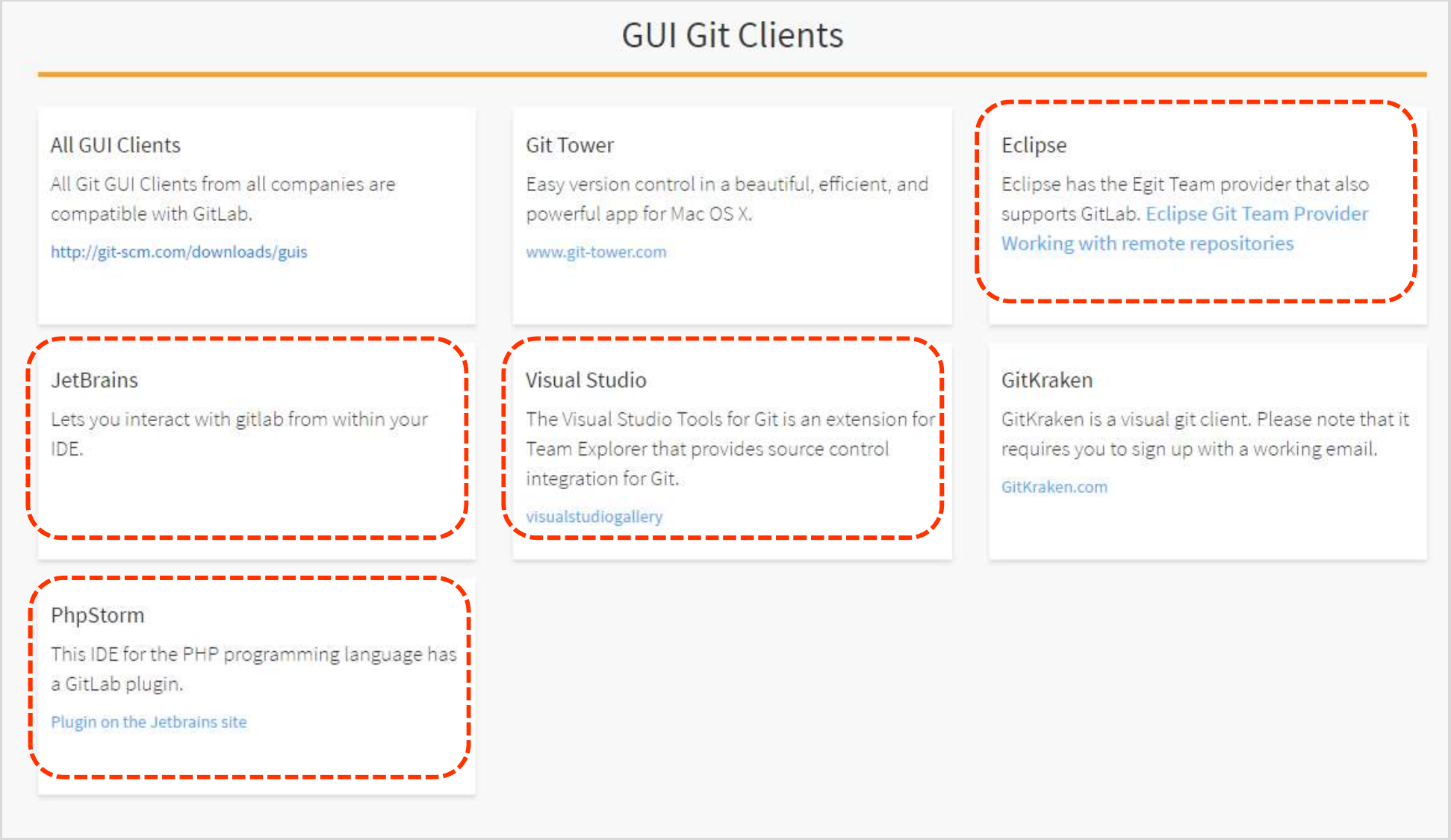


gitg
Platforms: Linux
Price: Free

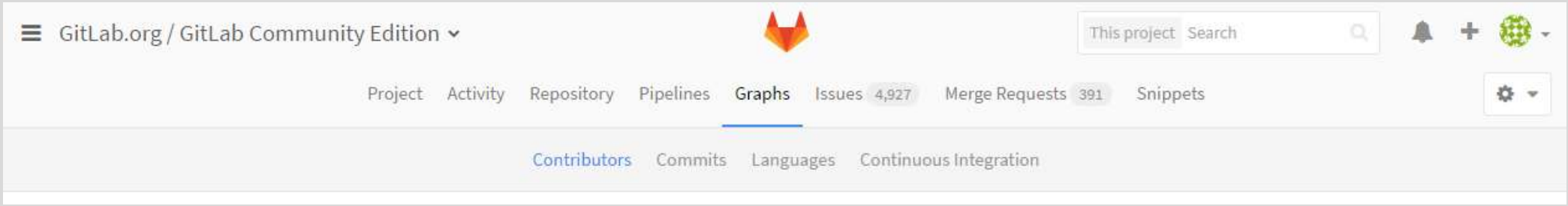
Git – SourceTree klijent

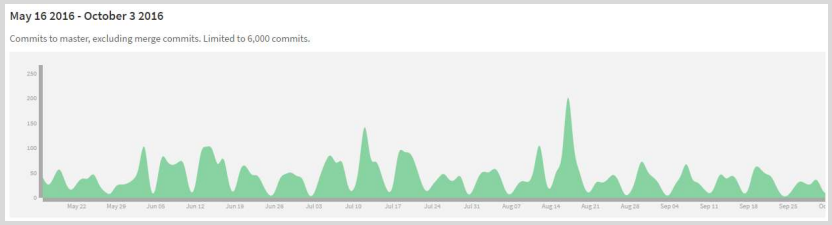


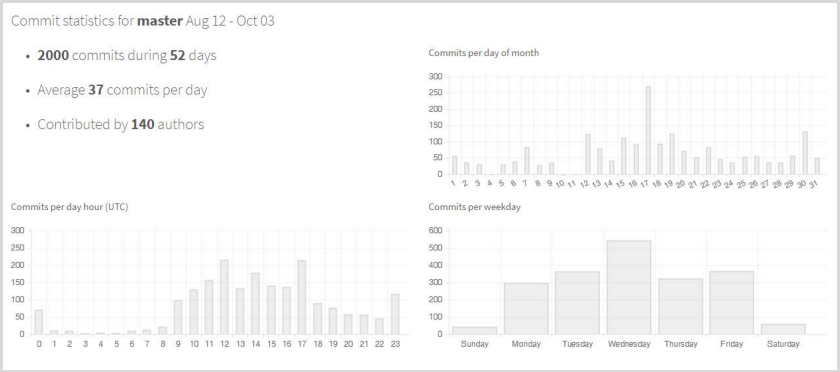
Integracija s razvojnim okruženjem

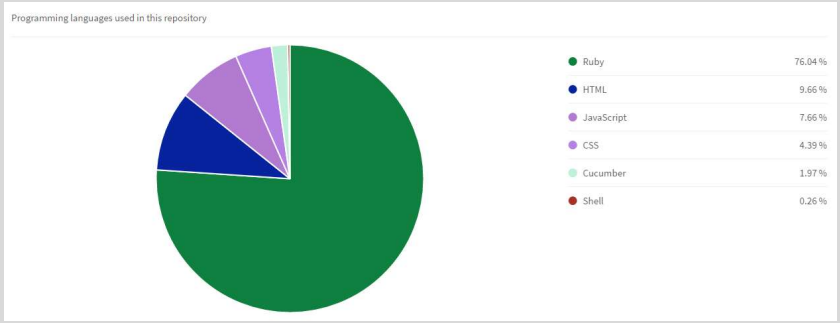


Pregled *GitLab* statistike









REFERENCE I LITERATURA

- Predavanja ovog predmeta
- GNU Project, RCS, https://www.gnu.org/software/rcs/rcs.html, pristupljeno 03/2015.
- Apache, Apache Subversion, http://subversion.apache.org/, pristupljeno 03/2015.
- Git, http://git-scm.com/, pristupljeno 03/2015.
- Nastavni materijali kolegija Oblikovanje programske potpore, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu.