

ThoughtWorks®

Sisteme de versionare software

George Popa
ThoughtWorks Romania

Sisteme de versionare - *version control systems (VCS)*

Definiție

- Instrument de dezvoltare software utilizat în gestionarea multiplelor versiuni ale fișierelor și dependențelor unei aplicații, înregistrând toate stările acestora, inclusiv modificări, autori și comentarii privind fiecare modificare.

Necesitatea unui sistem de versionare al proiectului

- Securitate - *repository* securizat, backup;
- Lucrul în echipă - permite colaborarea prin distribuirea modificărilor;
- Istoria proiectului - permite revizuirea stărilor anterioare ale proiectului;
- Integrare cu project trackers - modificările și comentariile vor apare în tracker.

Exemple de sisteme de versionare

- Apache Subversion (SVN) - <https://subversion.apache.org/>
- CVS - <http://www.nongnu.org/cvs/>
- Git - <https://git-scm.com/>
- Mercurial - <https://www.mercurial-scm.org/>
- Perforce - <http://www.perforce.com/>

Concepte VCS

- **Repository** - server de fişiere (bază de date) unde sunt stocate datele proiectului software
- **Commit** - modificări efectuate asupra unor fişiere, publicate în *repository*
- **Revision** - versiune a proiectului, ca urmare a aplicării unui commit
- **Working directory** - fişierele vizibile în sistemul de operare / IDE, ce corespund unui Revision curent, din Repository
- **Branch** - linie de dezvoltare a proiectului, ce conţine o istorie proprie
- **Merge** - operaţie de combinare a modificărilor din *branch*-uri separate
- **Checkout / fetch** - operaţiunea de descărcare modificări de pe *repository*
- **Push** - încărcarea modificărilor de pe un *repository* local pe un *repository* remote.

Repository

- O instanță a sistemului de versionare, ce poate funcționa independent sau în colaborare cu alte Repository - clone
- Interacțiunea dintre **repositories** se realizează prin operațiuni de **fetch** sau **push**
- Conține întreaga istorie a proiectului, pe fiecare linie de dezvoltare (branch), modificări (commit) și contributori.



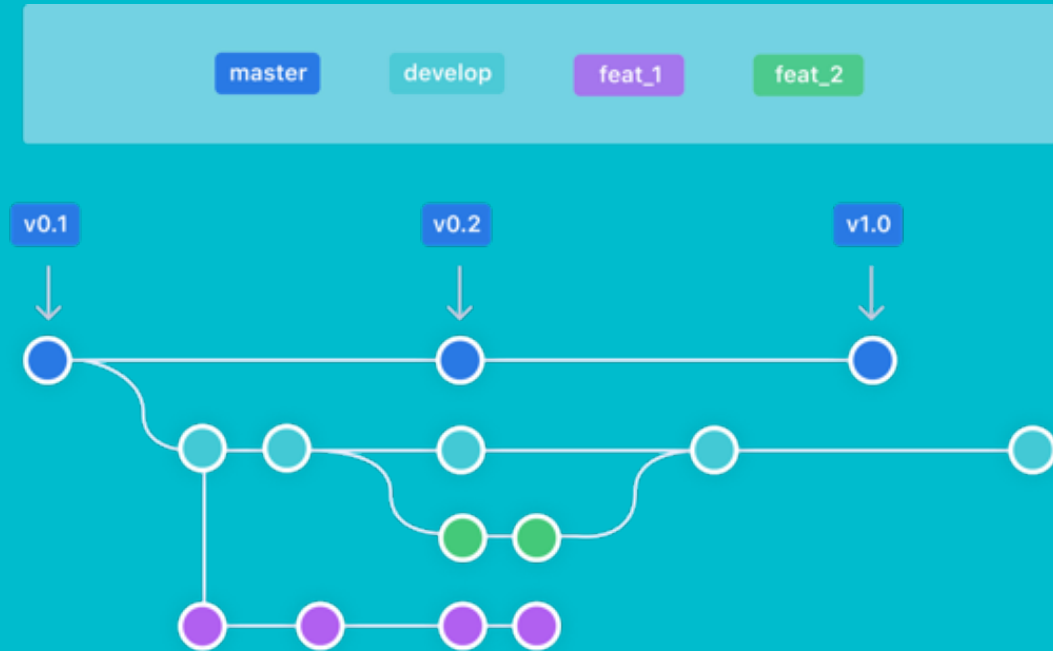
Commit

- Modificare unitară a fișierelor proiectului, ce dereglă păstrează proiectul într-o stare consistentă, și la care ne putem întoarce oricând.
- Commit-ul este atomic, reversibil și pe lângă modificările propriu-zise conține și informații despre autor, timestamp și comentarii.
- Ultimul commit se numeste HEAD sau *tip of the branch*

```
4 ■■■■ nodes/ba780c3a.39e07/func
...    ...    @@ -1,4 +1,4 @@
1      -if (true) {
2      -    console.log('Hello world.')
      1  +if (false) {
      2  +    console.log('Goodbye world.')
3      }
4      return msg; ↵
```

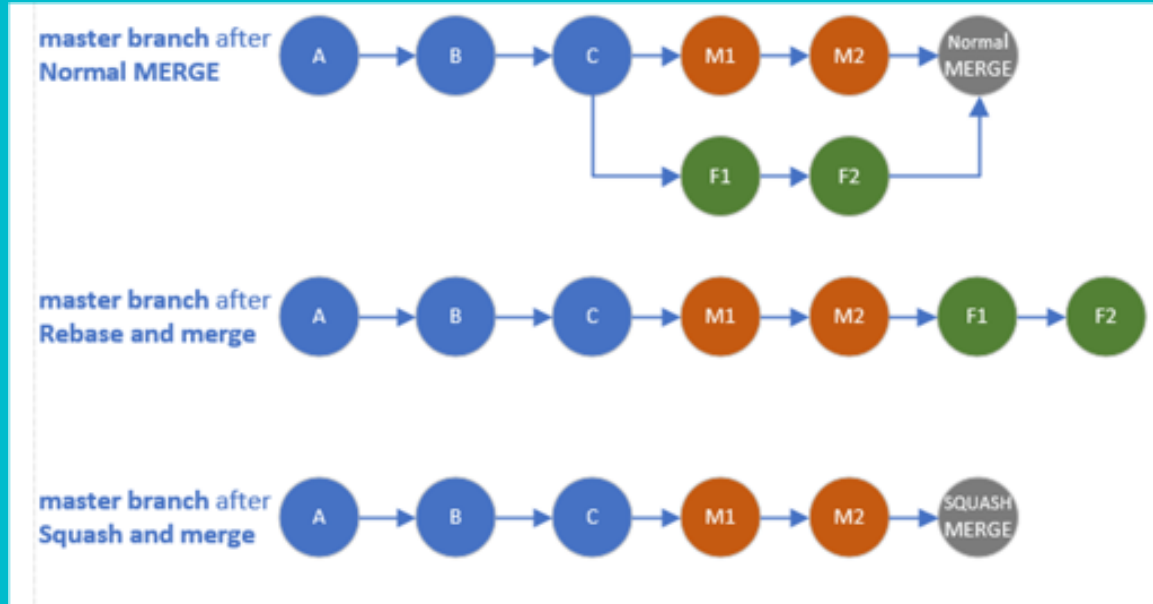
Branch

- Linii paralele de execuție a istoriei proiectului
- Fiecare branch reprezintă o suită de commits particulară
- Operațiunile de fork și merge reprezentând interacțiunile dintre branch-uri
- Trunk / master - conventional este branch-ul de bază al proiectului.

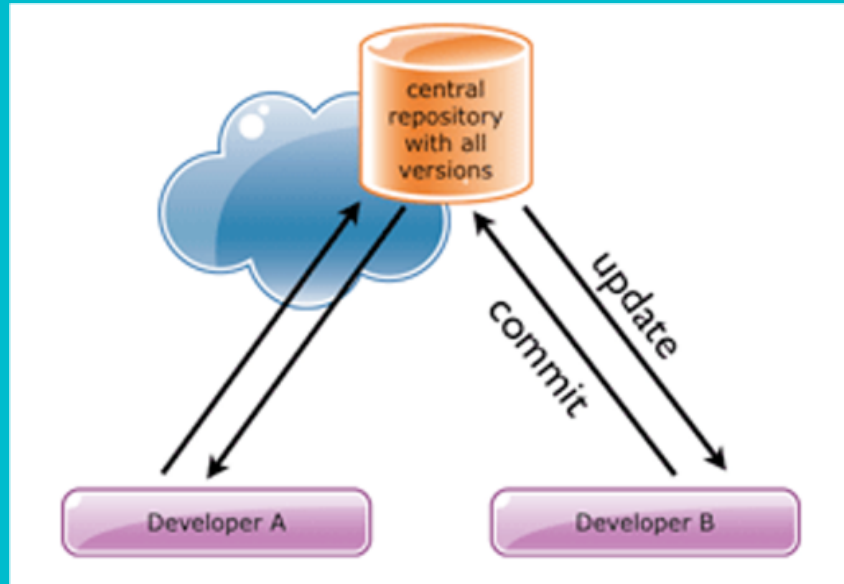


Merging branches

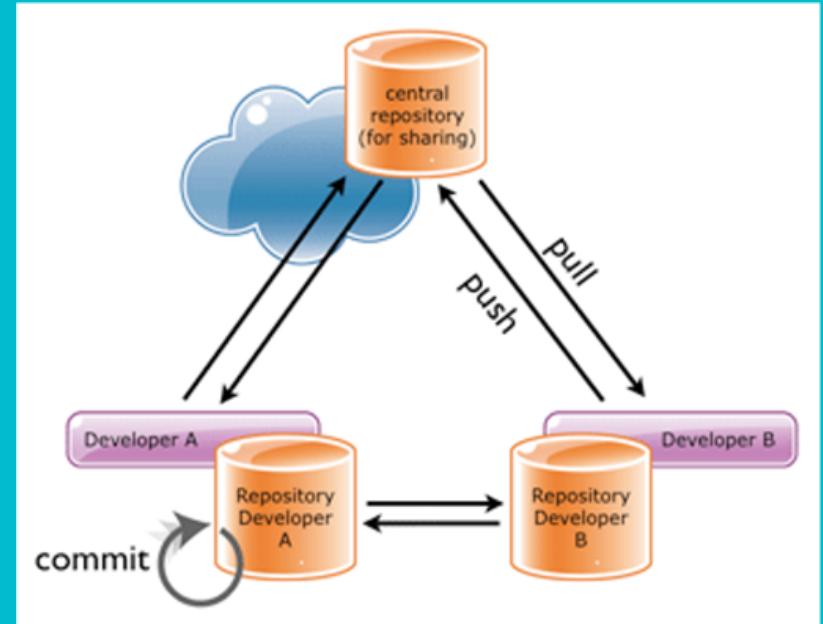
- Metoda de unificare a branch-urilor depinde de convențiile de dezvoltare:
 - Merge commit
 - Rebase work
 - Squash commit



Arhitecturi ale sistemelor de versionare



Arhitectura centralizată
(exemplu: SVN)



Arhitectura distribuită
(exemplu: Git)



git – Source Control Management

Sistem open source, descentralizat, dezvoltat de Linus Torvalds (autorul Linux kernel).

Particularități:

- Salveaza patches (diferente) pentru fiecare commit
- Local branches folosite intensiv
- Operațiuni de merge, rebase, fork mult mai facile decat in SVN
- Repository principal poate fi schimbat
- Fork & pull request

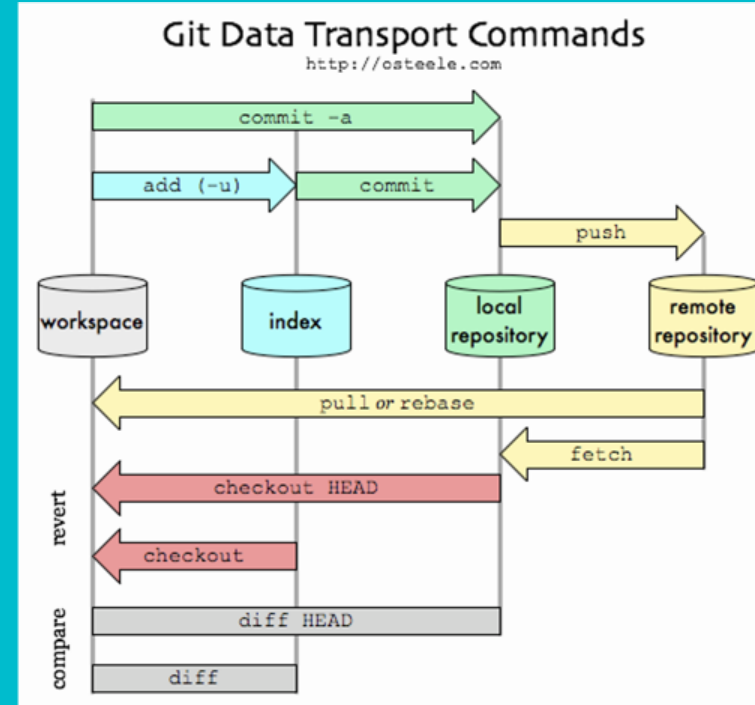
Documentatie: <https://git-scm.com/doc>

ThoughtWorks®

Git – Source Control Management

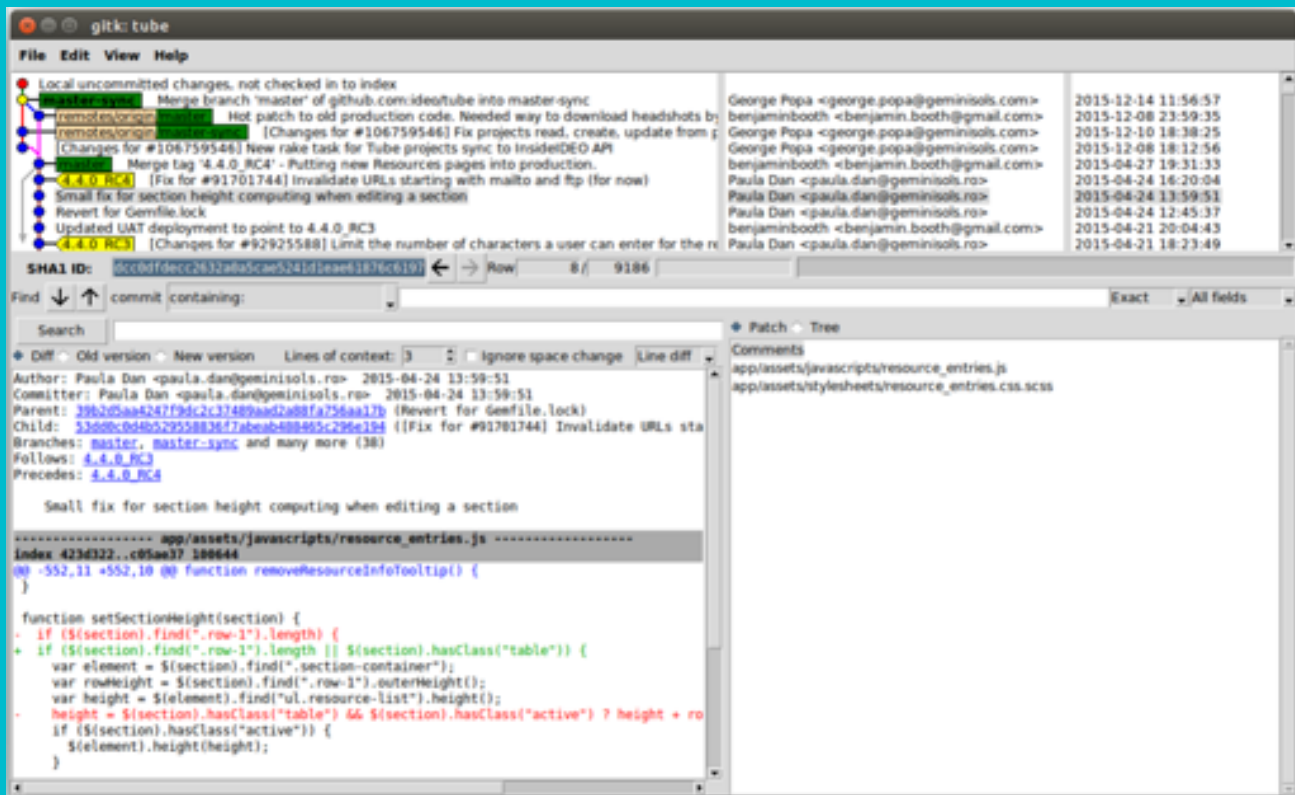
Comenzi uzuale:

- `git config user.name / user.email`
- `git init`
- `git add [file1, file2...]`
- `git commit -m "First commit"`
- `git remote add origin url.git`
- `git push origin master`
- `git clone url.git`
- `git pull origin master`
- `git rebase origin master`



Git GUI tools – gitk

<https://git-scm.com/downloads/guis>



Getting started!

(part 1)

Creați-vă un cont pe GitHub (<https://github.com>), iar un membru al echipei (Administrator) poate crea un repository nou.

Configurați git local:

- `git config --global user.name john-doe`
- `git config --global user.email johndoe@example.com`
- Comunicați user.name către repository Administrator pentru a vă acorda acces la proiect.

Descărcați proiectul de test:

- `git clone git@github.com:georgpopa/computer.git`
- `git status`

ThoughtWorks®



Getting started!

(part 2)

Creați un fișier nou în cadrul proiectului:

- touch nume_prenume.txt
- git status

Urcați fișierul nou creat pe *repository* github.com și descărcați fișierele colegilor:

- git add nume_prenume.txt
- git commit -m "Upload nume_prenume.txt"
- git pull origin master
- git push origin master
- git status

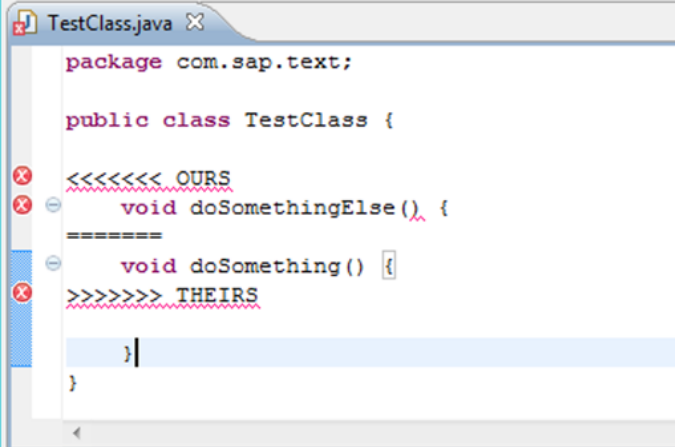
ThoughtWorks®



Git conflicts

- Inevitabile în munca în echipa!
- Apar de regulă când sunt modificări simultane în aceleași linii de cod.
- Nu există tool-uri automate care să rezolve toate conflictele.
- Rezolvarea conflictelor implică decizii luate de echipă, specifice logicii aplicației.

ThoughtWorks®

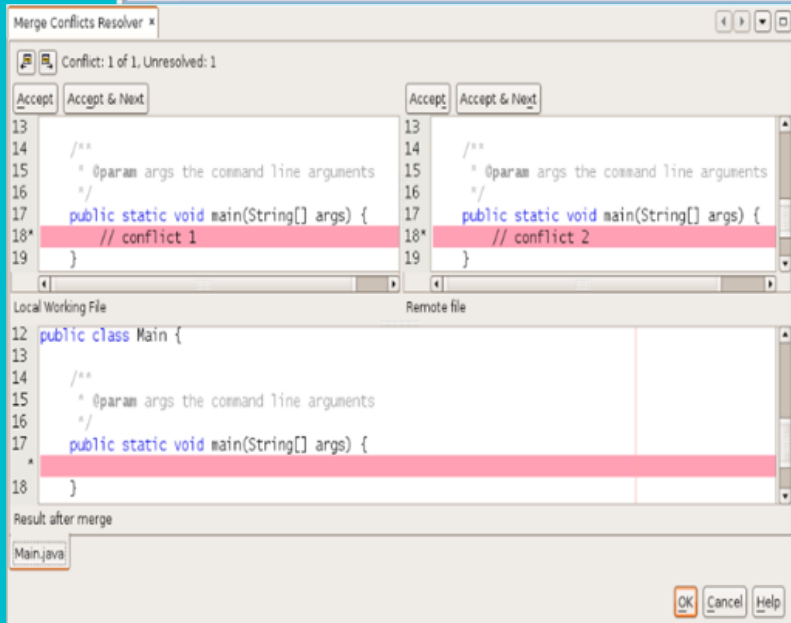


```
package com.sap.text;

public class TestClass {

    <<<<<<< OURS
    void doSomethingElse() {
        =====
    void doSomething() {
    >>>>>>> THEIRS

}
}
```



Getting started!

(part 3)

- În echipe de câte două persoane, modificați aceleași linii din același fișier:
 - `vi george.popa.txt`
- Comiteți modificările create și apoi efectuați *pull* de pe <https://github.com/> :
 - `git add george.popa.txt`
 - `git commit -m "Rename & repair greeting"`
 - `git pull origin master`
 - `### CONFLICT ### TODO: resolve!`
 - `git add george.popa.txt`
 - `git commit -m "Resolve conflict"`
 - `git push origin master`

ThoughtWorks®

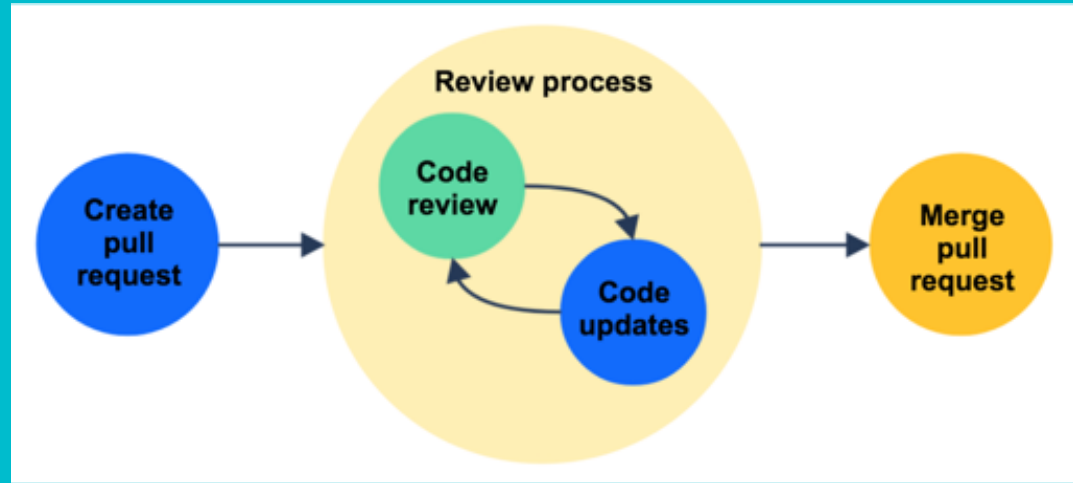


Git branches



- Utilizatorii git lucrează dereglă pe multiple branch-uri locale;
- Operatii cu branch-uri:
 - `git checkout -b new_feature`
 - `git rebase master`
 - `git checkout master`
 - `git merge new_feature`
 - `git push origin master`
- Pull requests

Pull request (specific GitHub)



- permite informarea echipei că există o nouă funcționalitate / bug fix, care așteaptă a fi integrată în *master*;
- acceptă feedback și permite aplicarea de modificări asupra commit-ului inițial, înainte de integrare;
- integrarea cu alte instrumente (CI builds, etc...), permite validarea unitară, funcțională sau a altor metrici înainte de integrare.