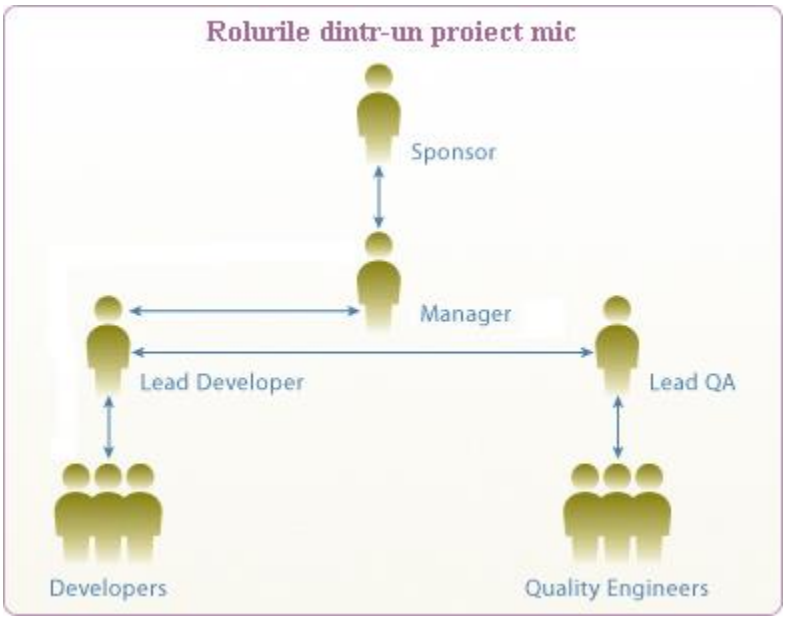
Rolurile în cadrul Echipei de Proiect

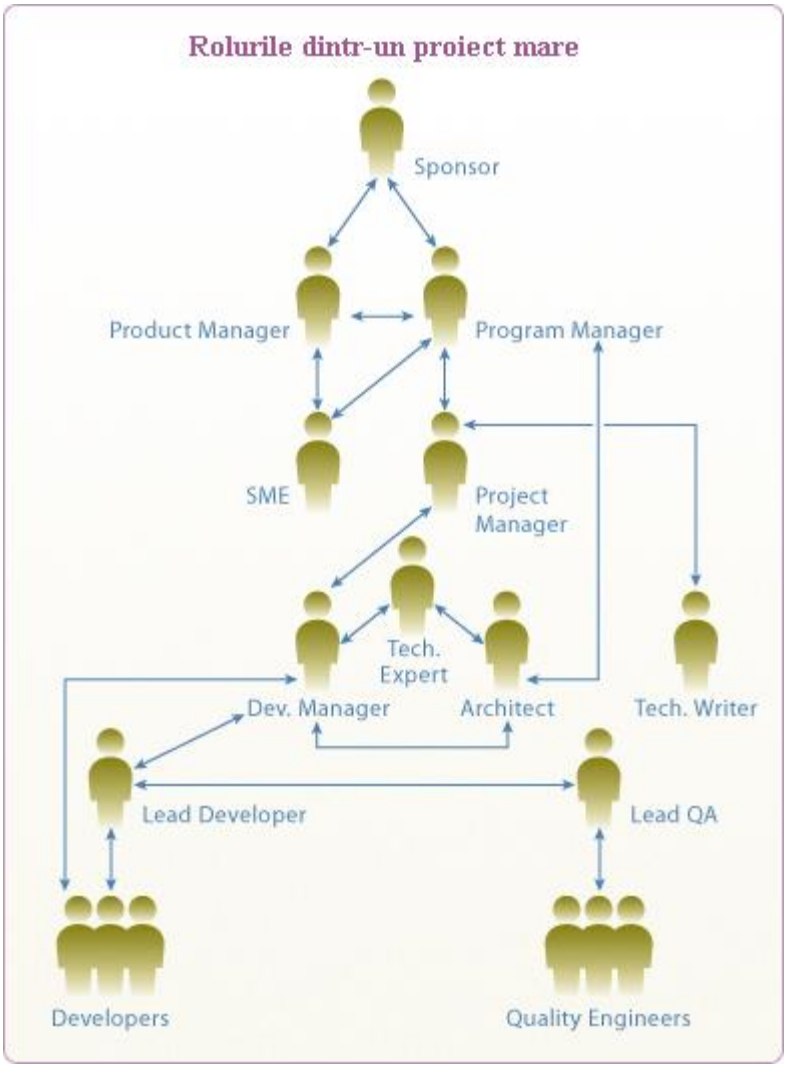
Cu cât proiectul este mai mare, cu atât rolurile din echipă sunt mai diversificate (mai specializate pe o anume arie de activități).



Într-un ***proiect mic***, principalele roluri din echipă sunt:

1. **Project Manager**
   - responsabil de succesul/eșecul proiectului
   - planifică & monitorizează evoluția proiectului
   - ia decizii
   - adesea, are background tehnic (pentru a înțelege ciclul de viață al proiectului)
   - întotdeauna, are abilități soft (de conducere, de motivare și stimulare, de conștientizare a cauzelor, de decizie, de negociere, de previziune, de inovație)
   - certificări PMI (Project Management Institute): PMP (Project Management Professional)
2. **Team Leader** (Lead Developer, Lead QA)
   - responsabil de execuția conform standardelor a activităților planificate
   - monitorizează activitatea echipei sale
   - intervine pentru a corecta, pentru a îndruma (coaching/support)
   - raportează managerului statusul real al proiectului
3. **Software Developer**
   - responsabil de implementarea software a cerințelor
   - în metode de dezvolare agile contribuie și la design, arhitectură, specificații
4. **Tester** (Quality Engineer)
   - responsabil de verificarea funcționalităților și a performanțelor produsului

- scrie scenarii de test, le execută, analizează rezultatele ⇒ rapoarte de testare



Într-un ***proiect mare***, pe lângă rolurile de mai sus, apar și altele, precum:

1. **System Architect**
   - responsabil de arhitectura produsului software
   - captează interesele partenerilor de business și le transpune alegând arhitectura potrivită
2. **Technical Writer**
   - responsabil de documentația tehnică a proiectului
   - primește documente tehnice de la ingineri, pe care le editează spre a fi corecte, clare și conforme cu standardele în vigoare

3. **Analyst** (Business Analyst, Business Systems Analyst, Systems Analyst, Requirements Analyst)
   - responsabil de înțelegerea corectă a cerințelor de business
   - studiază specificațiile clienților, clarifică toate detaliile proiectului cu clientul și cu partenerii (stakeholders)
   - redactează specificațiile aplicației software
4. **Consultant**
   - analizează fezabilitatea ofertelor de proiecte
   - propune soluții pentru diverse probleme legate de ciclul de viață al proiectelor
   - evaluează gradul de atingere a obiectivelor proiectelor
5. **Experți business**: SME (Subject Matters Expert), Product Manager, Program Manager

Software Design Document (SDD)

- document de **specificație** a soluției pentru sistemul software descris în **SRS**
- răspunde la întrebarea: *cum va fi construit sistemul software pentru a avea comportamentul descris în SRS?*
- prezinta metodologii, tehnologii, participanți și resurse implicate în proiect
- se poate redacta numai după finalizarea SRS-ului fiind un răspuns la cerințele prezentate în SRS
- este redactat de o echipă de software designers (proiectanți de sistem) și analiști business, pe baza SRS-ului și a experienței acestora
- reprezintă ghidul de construire a soluției folosit de echipa de dezvoltare a proiectului
- reprezintă un tool de analiză a întregului proiect în faza de început cât și de monitorizare pe parcurs

Secțiuni SDD

Un SDD are următoarele secțiuni:

1. **Modelul datelor** (*Data Design*)
   - prezintă *structurile de date* importante, *formatele fișierelor* folosite în cadrul soluției și*schema bazei de date*.
   - Structurile de date pot fi:
     - **globale** (structuri de date disponibile tuturor componentelor arhitecturii)

        - **de legătură** (structuri de date trimise ca argumente între componente pentru a asigura fluxul informației la nivel de aplicație)
        - **temporare** (structuri de date folosite temporar).
    - Schema bazei de date este descrisă prin:
        - **diagrama schemei** bazei de date
        - **descrierea semnificației tabelelor** și, pentru fiecare tabelă, descrierea semnificației coloanelor și indicarea cheilor primare și referențiale.

2. **Modelul arhitectural** (*Architectural Design*)
    - este o structură ierarhică alcătuită din componente interconectate
    - prezintă arhitectura sistemului – atât descriptiv, cât și sub forma unei diagrame de arhitectură
    - descrie:
        - fiecare componentă a arhitecturii,
        - restricțiile de implementare ale componentelor,
        - interacțiunea dintre componentele sistemului.
    - poate fi reprezentat prin :
        - **diagrame de componente** (pentru proiecte mari) - le-am numit "generic" în laboratorul 2: diagrame de arhitectură
        - **diagrame de clase**, în care relațiile ierarhice se bazează pe generalizare și specializare (pentru proiecte mici).

3. **Modelul interfeței cu utilizatorul** (*User Interface Design*)
    - prezintă *ferestrele* aplicației și *succesiunea* lor în cadrul sistemului.

4. **Elementele de testare** (*Testing Issues*)
    - identifică *componentele critice* ale aplicației (componente a căror performanță influențează decisiv performanța globală a aplicației)
    - propune *alternative* de proiectare a componentelor critice (pentru a fi folosite în cazul insuccesului soluției propuse inițial).

Conținutul unui SDD cuprinde:

1. Scopul documentului (*Document purpose*)
2. Obiective (*Objectives*)
3. Conținutul documentului (*Document overview*)
4. Modelul datelor (*Data Design*)
    - Structuri de date globale (*Global Data Structure*)

- Structuri de date de legătură (*Linking Data Structure*)
- Structuri de date temporare (*Temporary data structure*)
- Formatul fișierelor utilizate (*File Formats*)
- Descrierea bazei de date (*Database description*)
  - Diagrama schemei bazei de date (*Database Structure Diagram*)
  - Descrierea tabelelor (*Tables Description*)

5. Modelul arhitectural și modelul componentelor (*Architectural and component-level design*)
   - Arhitectura sistemului (*System Architecture*)
     - Șabloane arhitecturale folosite (*Architectural Patterns*)
     - Diagrama de arhitectură (*Architecture Diagram*)
   - Descrierea componentelor (*Component Description*)
   - Restricțiile de implementare (*Implementation Requirements*)
   - Interacțiunea dintre componente (*Component Interaction*)
6. Modelul interfeței cu utilizatorul (*User Interface Design*)
   - Succesiunea interfețelor (*Flowchart*)
   - Ferestrele aplicației (*Screen Images*)
7. Elemente de testare (*Testing Issues*)
   - Componente critice (*Critical components*)
   - Alternative (*Alternatives*)

Exemplu de Document SDD

Exemplu de Document SDD

Controlul versiunii - Git

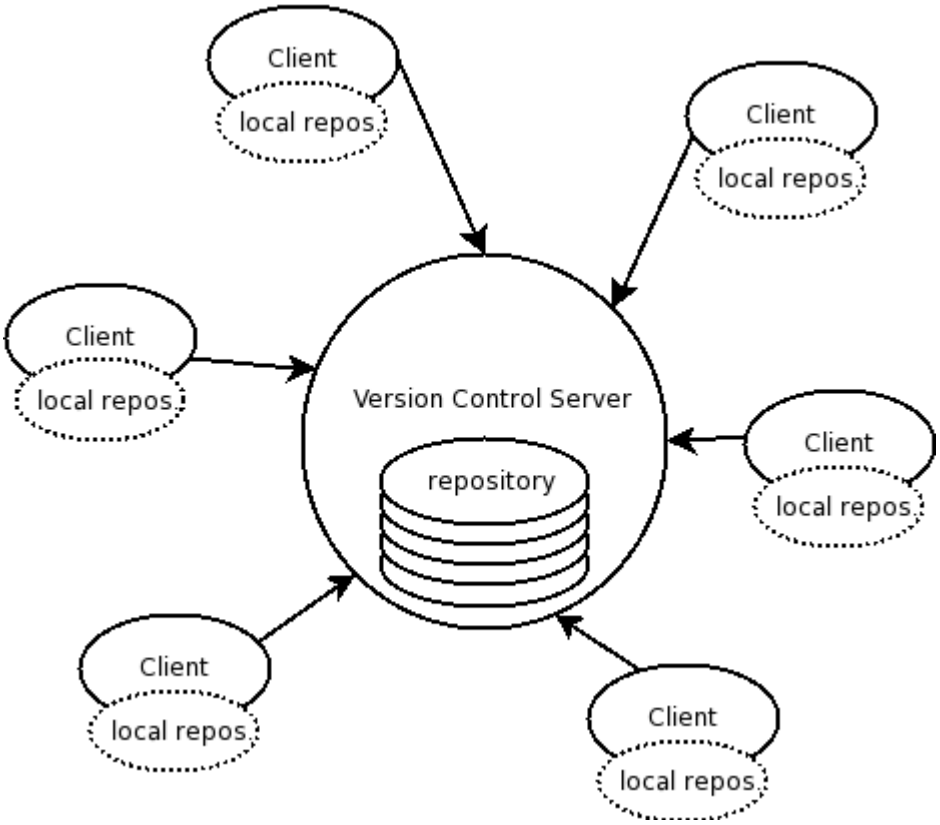
Sisteme pentru contolul versiunii (*Version Control Systems* - **VCS** - sau *Source Code Management* -**SCM**) sunt aplicații care permit lucrul colaborativ pe diverse fișiere, în special fișiere cod sursă. Sistemele pentru controlul versiunii sunt practic obligatorii în cadrul unui proiect cu dezvoltatori multipli. Astfel de sisteme reține istoricul modificărilor efectuate de fiecare dezvoltator și folosesc comenzi specializate care să faciliteze transmiterea acestor modificări între dezvoltatori.
Exceptând sistemele de gestiune a surselor, prezentate mai detaliat în continuare, și alte aplicații folosesc versiuni:

- Wiki Engines - pentru fiecare modificare se reține doar schimbările de la versiunea anterioară; orice modificare poate fi anulată;
- Google Docs - versiuni pentru fiecare modificare;

- MS Office, OpenOffice - permit atașarea unor numere de identificare pe documentele editate.

Principiul de funcționare a sistemelor de gestiune a codului este comun:

- Sursele sunt păstrate, de obicei, într-un **repository** (**depozit**) aflat pe un server accesibil tuturor dezvoltatorilor. Acest repository constituie mecanismul principal de sincronizare a surselor dezvoltatorilor.

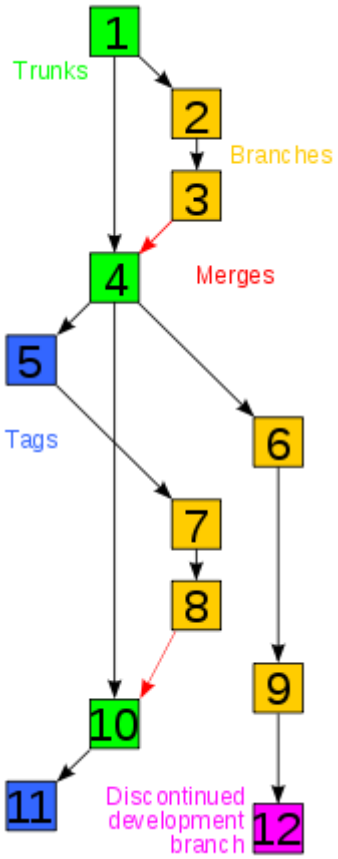


- Fiecare dezvoltator obține o copie a repository-ului, denumită **copie locală**. Copia rezidă pe sistemul dezvoltatorului. Dezvoltarea se va realiza în acest director.
- Există două operații care permit comunicația cu repository-ul:
  - **update** sau **pull** înseamnă actualizarea copiei locale cu informațiile din repository; este posibil ca alți dezvoltatori să fi comis schimbări în repository;
  - **commit** sau **push** înseamnă transmiterea/comiterea modificărilor locale în repository; ceilalți dezvoltatori vor folosi comanda **update** pentru actualizarea copiei locale cu acele informații.

Câteva cuvinte cheie utile în lucrul cu sisteme de control al versiunii sunt (denumirile sunt în engleză pentru că aceasta este forma uzuală de utilizare):

- **repository** locul (centralizat) în care se găsesc sursele și informațiile de modificare;

- **working copy** copia locală a repository-ului folosită de un dezvoltator;
- **check-out** operația de creare a unei copii locale pornind de la repository (clonarea repository-ului);
- **commit** (vezi mai sus)
- **update** (vezi mai sus)
- **branch** o ramură de dezvoltare (un fork) care poate fi dezvoltată în paralel față de ramura principală de dezvoltare;
- **trunk** ramura principală de dezvoltare;
- **tag** o referință la un snapshot al surselor la un moment dat în timp;
- **merge** o operație de unificare a două seturi de schimbări petrecute distinct;
- **conflict** situație în care două modificări sunt efectuate din surse diferite și sistemul nu poate găsi o soluție de unificare (merge) a acestor schimbări; dezvoltatorul va trebui să rezolve conflictul fie prin unificarea schimbărilor, fie prin selectarea unei singure variante



Există două tipuri de sisteme pentru gestiunea codului sursă:

- sisteme centralizate (Subversion, CVS, perforce);

- sisteme distribuite (Git, Darcs, Mercurial, Bazaar).

Detalii despre diferențele dintre acestea (mai degrabă între doi dintre cei mai cunoscuți reprezentanți, Subversion și Git) găsiți aici.