

# Tehnici Avansate de Programare

Infrastructură și unelte

---

Petru Rebeja, Marius Apetrii

18 Februarie 2021

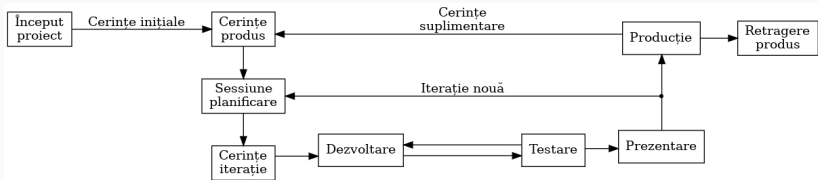
Facultatea de Matematică

Universitatea Alexandru Ioan Cuza, Iași

Pagina cursului: <https://rebeja.eu/dotnet-math>



# Ciclul de dezvoltare al aplicațiilor



# Etapa I: Identificarea unei nevoi

Un sistem software trebuie să satisfacă o anumită nevoie a utilizatorului:

- Automatizarea procesului manual,
- Performanță sporită
- Etc.

## Etapa II: Cerințe inițiale

- După identificarea nevoii se alcătuieste lista de cerințe inițiale.
- Opțional, se identifică attributele arhitecturale suplimentare ale sistemului.

## Etapa III: Minimum Viable Product

Din lista cerințelor inițiale se aleg cele care constituie un minim de viabilitate pentru proiect.

### **Definition (Minimum viable product)**

Versiune a produsului software care conține o cantitate suficientă de funcționalitate pentru a satisface utilizatorii timpurii.<sup>1</sup>

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Minimum\\_viable\\_product](https://en.wikipedia.org/wiki/Minimum_viable_product)

## Etapa IV: Dezvoltarea în iterații

- Cerințele sunt ordonate în funcție de importanță.
- Echipa alege  $N$  cele mai importante cerințe.

## Etapa IV: Dezvoltarea în iterații (cont.)

- Mai întâi se planifică abordarea (task breakdown).
- Fiecare membru ia câte o sarcină.



## Etapa IV: Dezvoltarea în iterații (cont.)

- Când o funcționalitate este gata, ea trece la testare.
- Dacă este conformă cu cerințele, aceasta este marcată ca finalizată.
- Altfel, revine la programator.

## Etapa IV: Dezvoltarea în iterații (cont.)

- La sfârșitul iterației se face o prezentare a funcționalităților implementate.
- Procesul se reia într-o iterație nouă.

## Etapa V: Lansarea în producție

- Lansarea în producție se face când:
  - Avem suficientă funcționalitate implementată.
  - Situația o impune (urgente, defecte etc.).

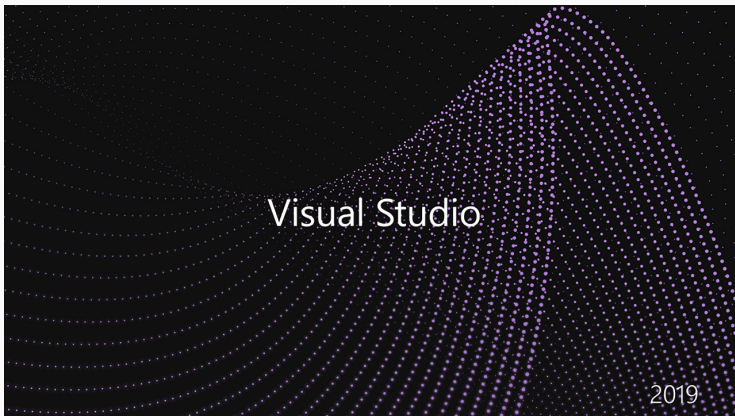
## Etapa VI: Întreținerea și dezvoltarea

- Utilizatorii semnalează erori și cer funcționalități noi.
- Acestea sunt adăugate la lista de cerințe.

## Etapa VII: Retragera produsului

Se întâmplă din mai multe motive:

- Nu aduce încasări suficiente,
- Perimare
- Etc.



---

<sup>2</sup><https://dev.infohub.cc/wp-content/uploads/2019/04/vs2019.jpg>

## Bune practici în scrierea codului-sursă

- Codul-sursă trebuie scris în limba engleză.
- Textul trebuie să fie preluat dintr-un set de resurse.
- Alegeți tipul de date potrivit.
- Rescrieți codul, nu dați copy/paste.
- Dacă nu există în sistemul de păstrare a istoricului atunci nu există.

- Produsele software mari pot ajunge și la milioane de linii de cod.
- Sistemele de control al reviziilor (version control systems) ne permit:
  - Să vedem toate modificările făcute asupra unui fișier,
  - Să revenim la o versiune anterioară,
  - Să lucrăm mai multe persoane simultan la același fișier.



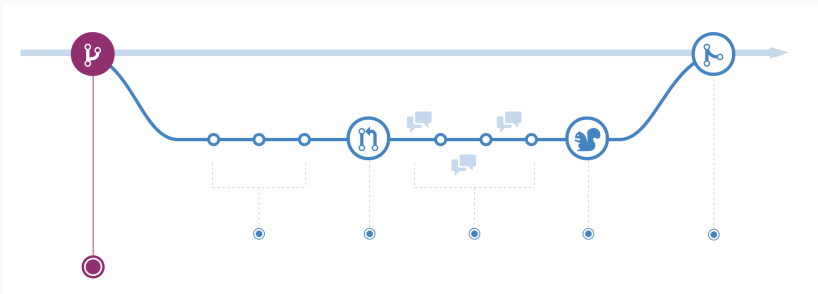
## Git SCM

Git este un sistem de control al reviziilor open-source, distribuit și gratuit. Sistemul a fost proiectat să gestioneze rapid și eficient proiecte de toate dimensiunile.<sup>3</sup>

---

<sup>3</sup><https://git-scm.com/>

# Fluxul de lucru *GitHub*<sup>4</sup>



<sup>4</sup><https://raw.githubusercontent.com/DRPrincess/BlogImages/master/qiniu/46e70c0bb557565b2fdcc1e5e4efefce.png>

## Bune practici pentru controlul reviziilor<sup>5</sup>

- Dacă nu există în sistemul de control al reviziilor atunci nu există deloc.
- Fiecare versiune trebuie să fie atomică.
- Preferați mai multe versiuni mici decât una mare.
- Descrieți modificările cât mai bine.
- Nu păstrați istoric pentru fișierele generate la compilare.

---

<sup>5</sup>[https:](https://www.troyhunt.com/10-commandments-of-good-source-control/)

[//www.troyhunt.com/10-commandments-of-good-source-control/](https://www.troyhunt.com/10-commandments-of-good-source-control/)

# Vă mulțumesc!

Mulțumesc pentru atenție!