



# *Sisteme cu MicroProcesoare*

## **Curs 10**

### **Proiectarea şi dezvoltarea aplicaţiilor cu MC**

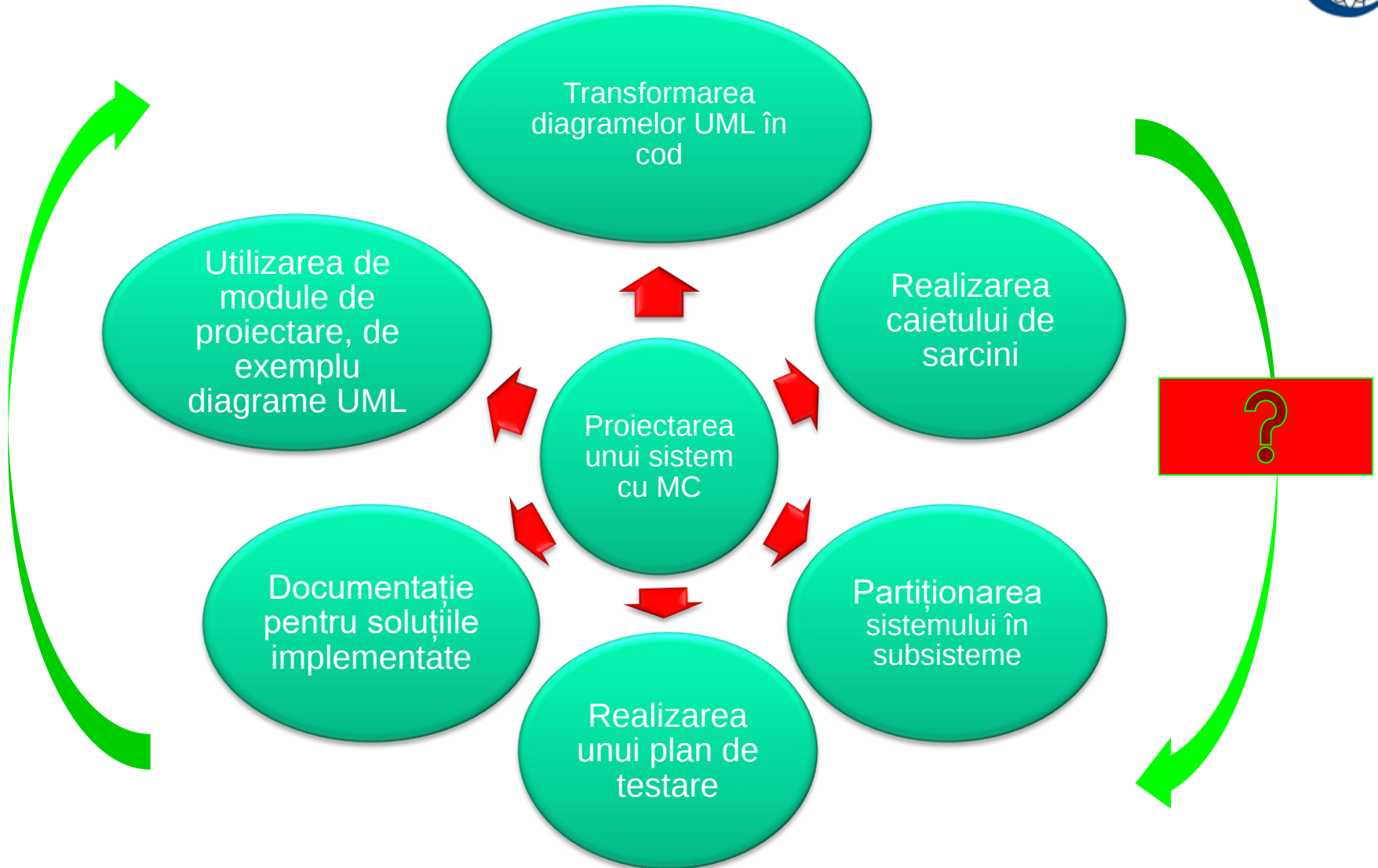
**Tiberiu COCIAS**



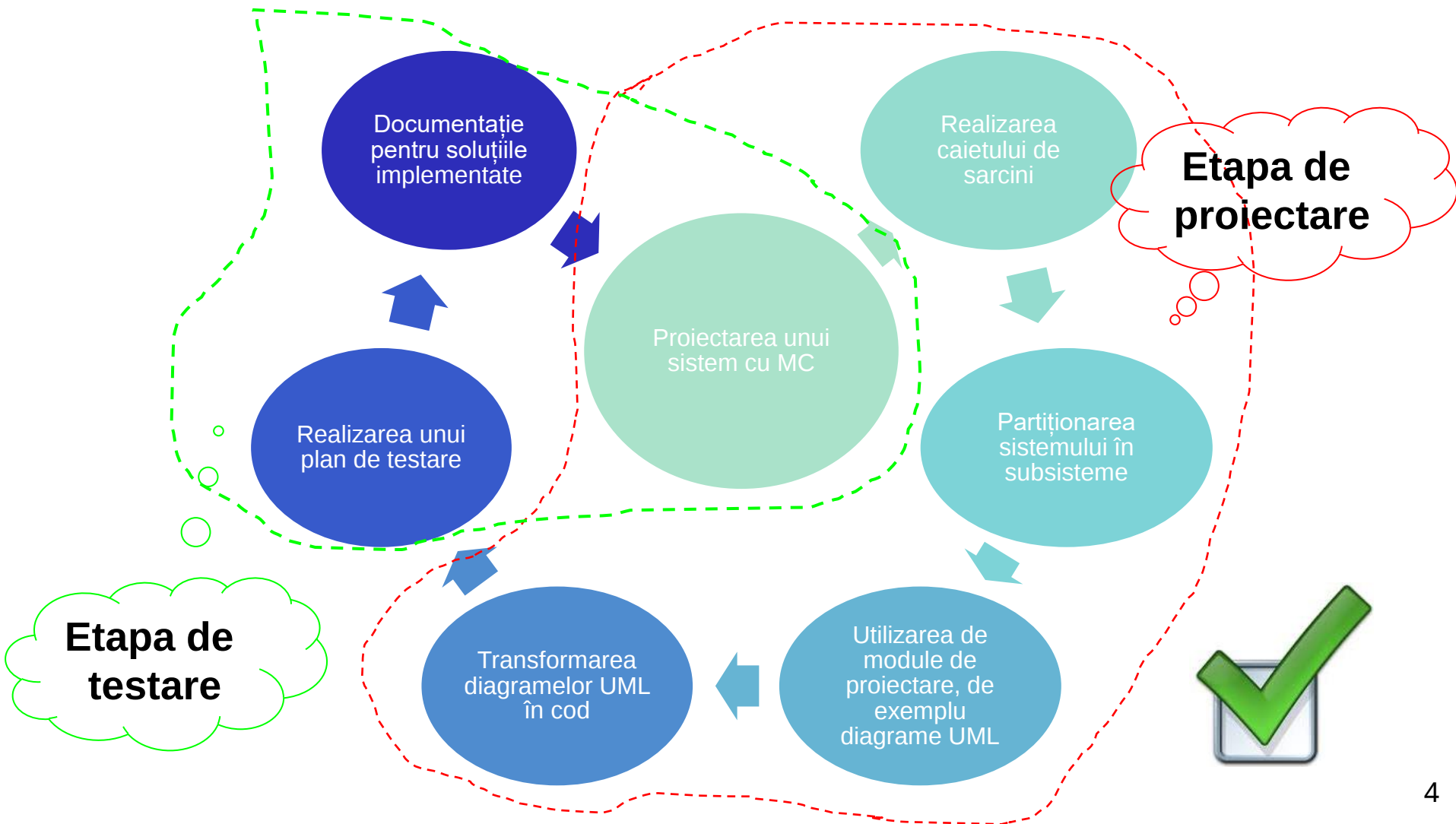
# Cuprins

- Obiective
- Proiectare și testare
- Caietul de sarcini
- Partiționarea sistemului
- Detalierea subsistemelor
- Scrierea și testarea primară a codului
- Planul de testare
- Documentare

# Obiective



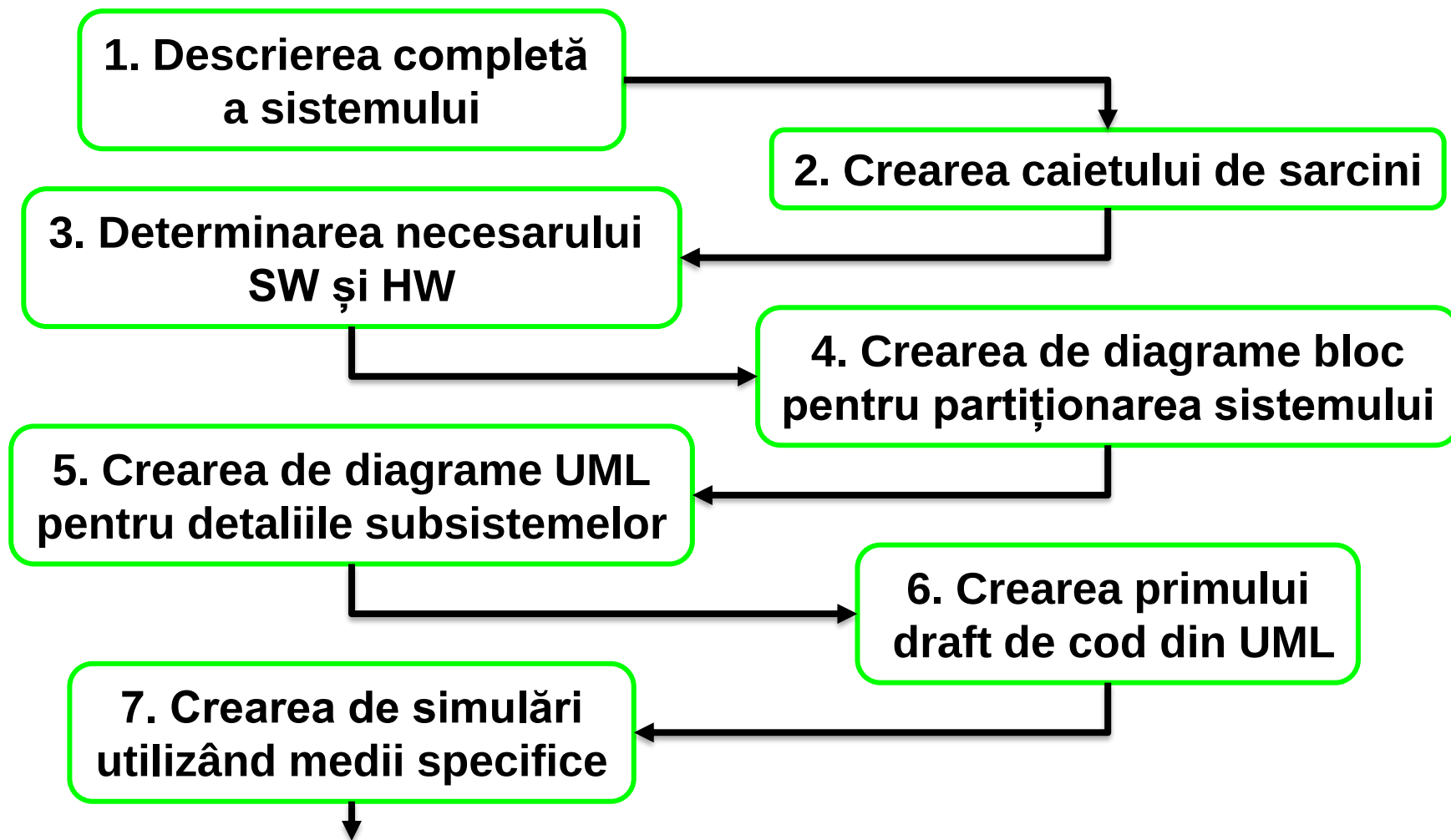
# Obiective





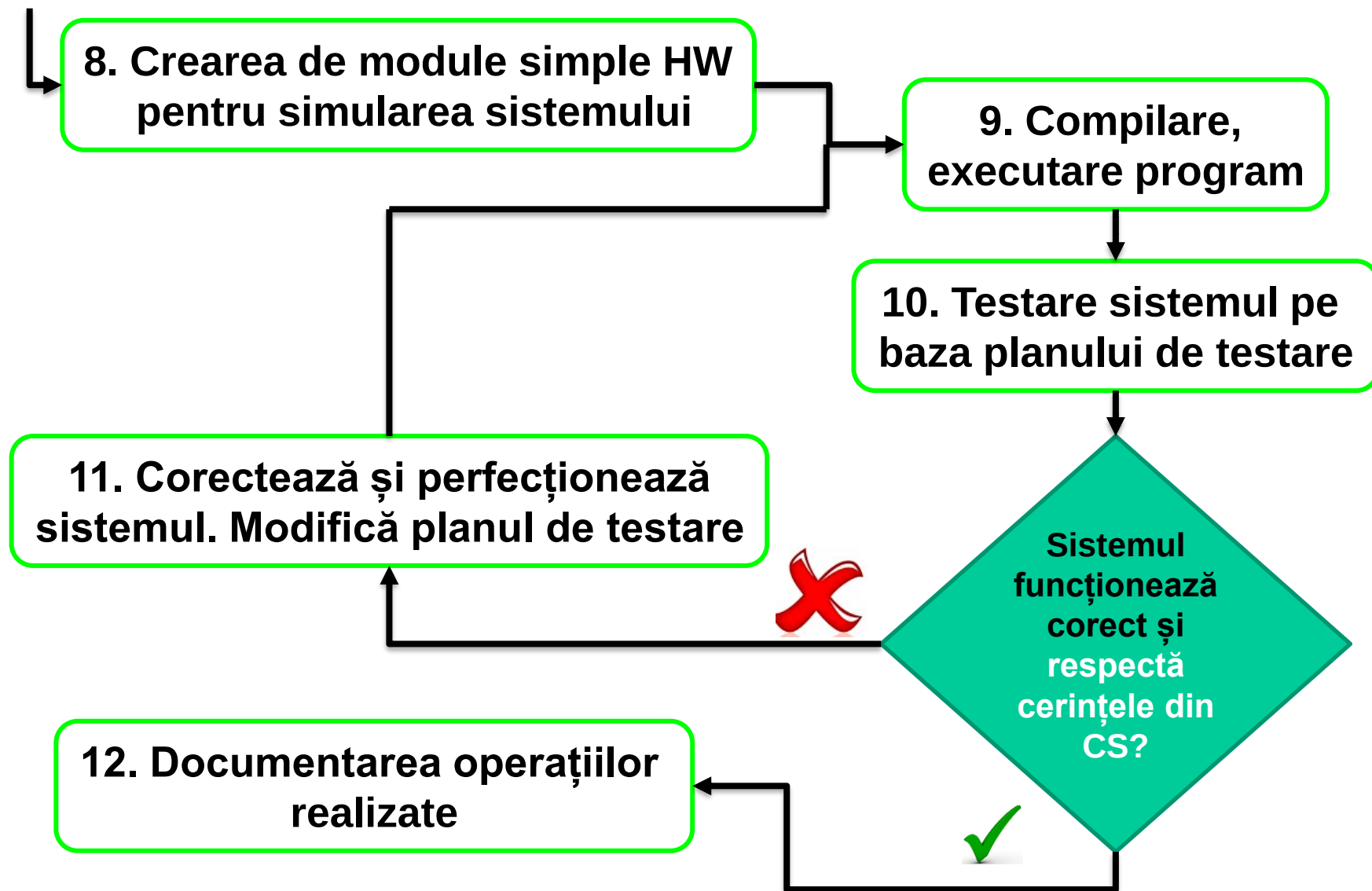
# Proiectare și testare

- Schema de principiu a celor două etape este următoarea:





# Proiectare și testare



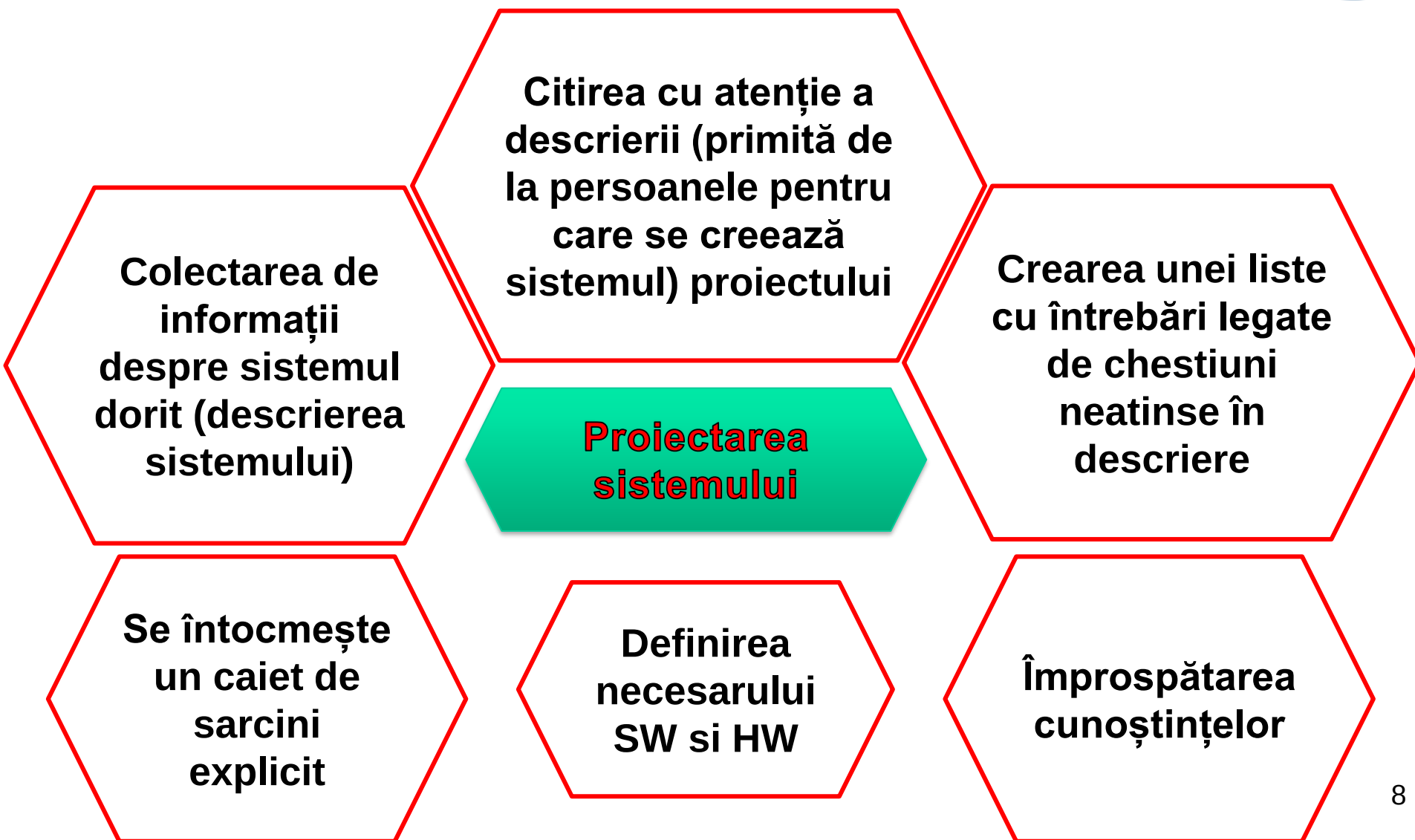


# Exemplu

- Sunteți contactat pentru a realiza un sistem care este capabil să controleze un ansamblu de porți
- Un document despre cum trebuie să funcționeze este trimis de firma contractantă
- Au loc mai multe întâlniri între firma solicitanta și dumneavoastră:
  - **Principiul de funcționare este schițat pe baza documentului care descrie sistemul**
  - **Se decide utilizarea de motoare de putere de cc**
  - **Controlul motorului se va face cu PWM**
  - **Responsabilitatea dvs este aceea de a crea algoritmul de control**
  - **Interfața cu partea HW a porților este realizată de altă companie**



# Descrierea sistemului, caietul de sarcini





# Exemplu

- Pentru un sistem de control al porților se utilizează:
  - **Butoane cu sau fără reținere pentru:**
    - Închidere, deschidere
    - leșire de urgență
    - Blocare
    - Închidere de urgență
- Dvs trebuie să elaborați o listă de acțiuni pe care trebuie să le implementați pentru buna funcționare a sistemului
- Trebuie să definiți toate circumstanțele în care poate funcționa:
  - **Disfuncționalități ale porților**
  - **Secvență de intrare incorrectă**
    - Apăsare închidere când poarta e deja închisă
    - Apăsarea a două butoane în același timp





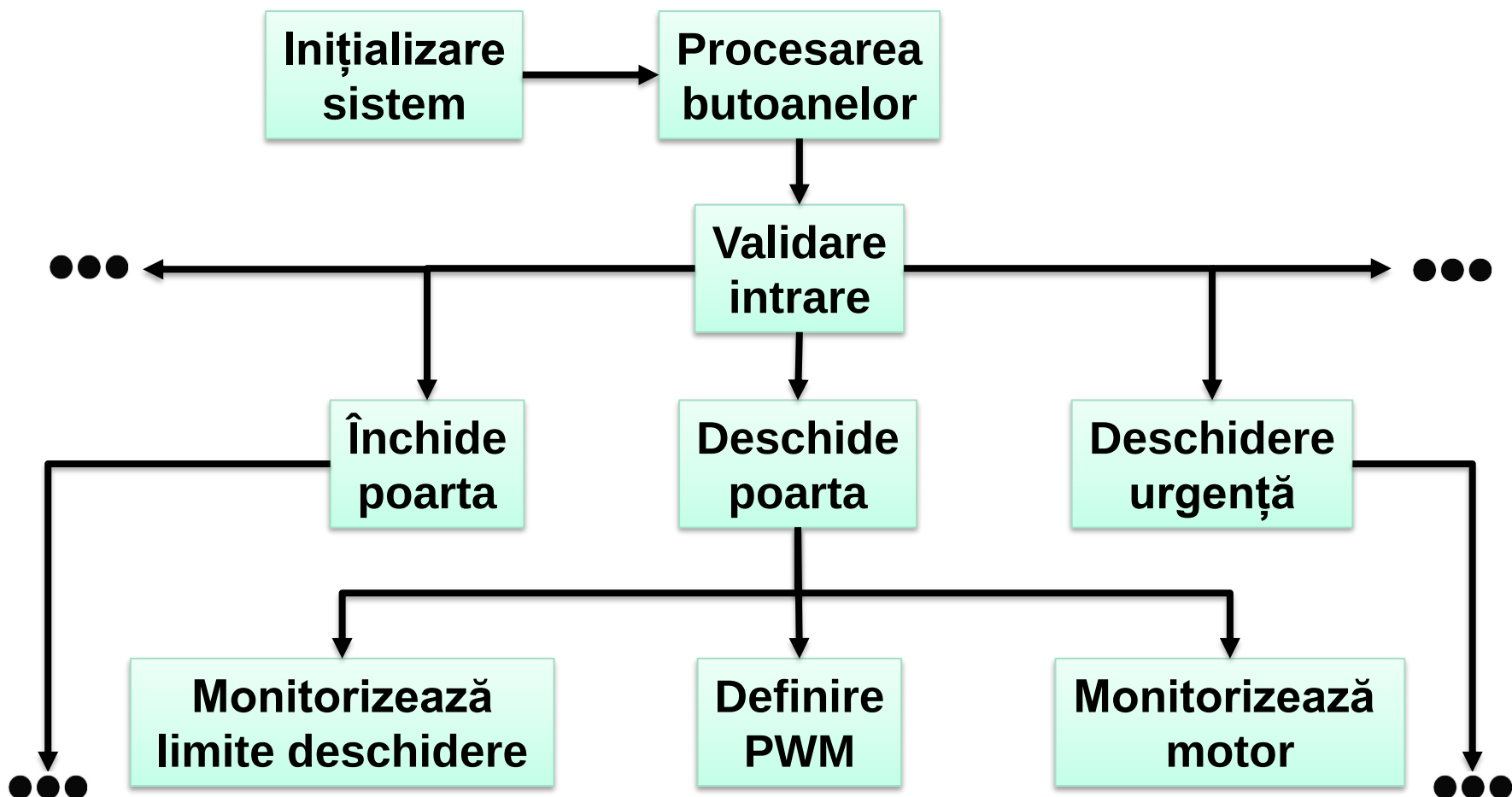
# Partiționarea sistemului

- Având caietul de sarcini disponibil, este necesară împărțirea sistemului în subsisteme
  - **Problema este modalitatea de interconectare între subsisteme**
- Se utilizează diagrame bloc pentru definirea subsistemelor
  - **Prezintă ierarhia dintre subsisteme**
  - **Interacțiunea dintre componentele SW și HW**
  - **Interfața către alte module**
- Partiționarea se realizează până la nivel unic funcțional
  - **Ex: Inițializare PORT intrare**
- Este necesară alegerea modului de implementare pentru fiecare subsistem: nivel SW sau nivel HW



# Exemplu

- Pentru un sistem de deschidere porți:





# Detalierea subsistemelor

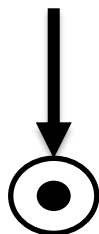
- Se poate utiliza UML (*Unified Modeling Language*)
  - **UML – metodă standardizată de documentare a sistemelor**
  - **Se poate utiliza diagrama de activități**
    - Este un tool din UML pentru documentarea operațiilor și funcționalității unui sistem
  - **Diagrama UML trebuie să fie suficient de explicită încât să se poată scrie algoritmi direct din ea**



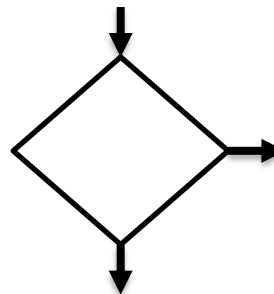
Etapă  
START



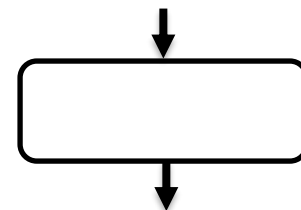
Transfer  
control



Etapă  
SFÂRȘIT



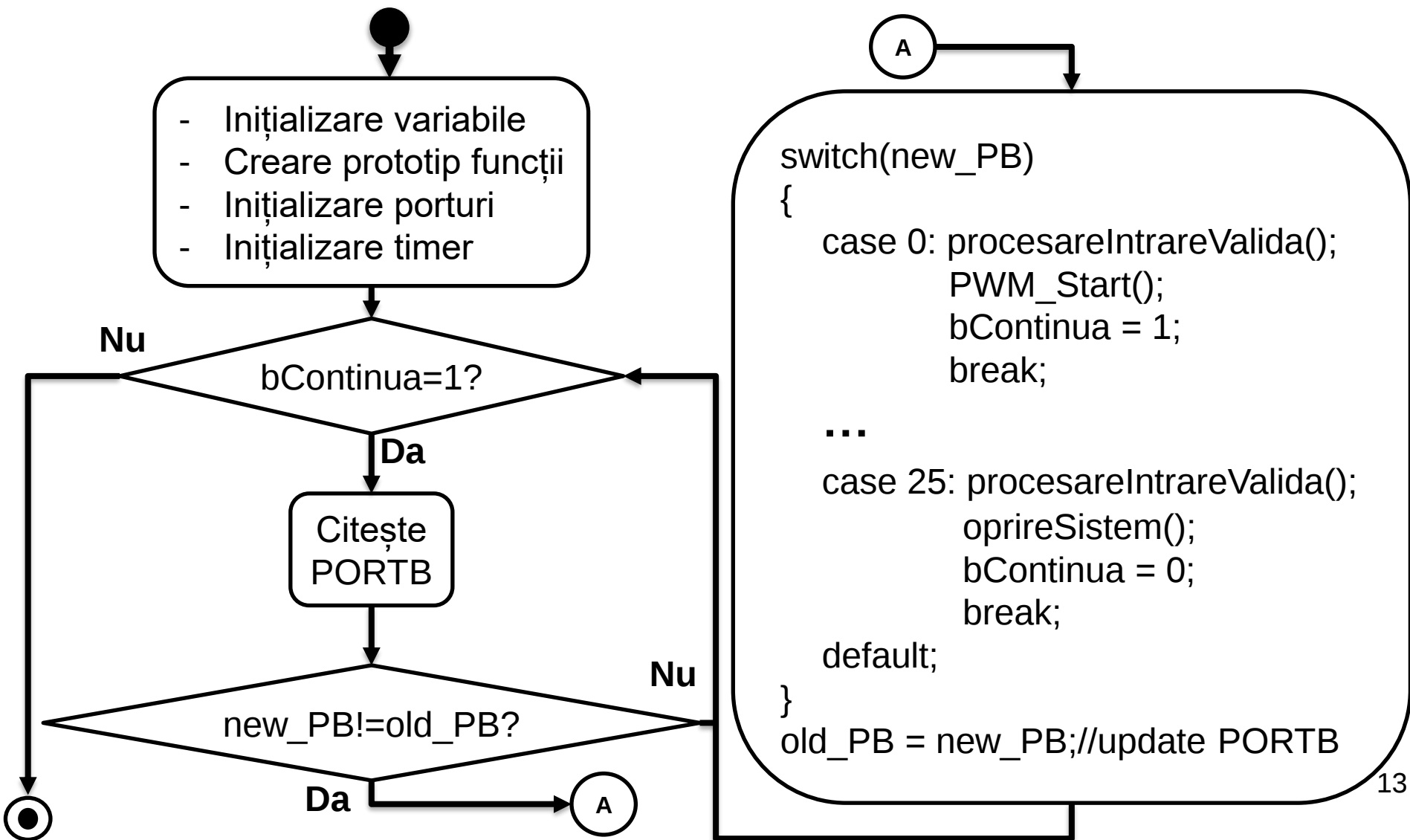
Etapă  
decizie



Acțiuni

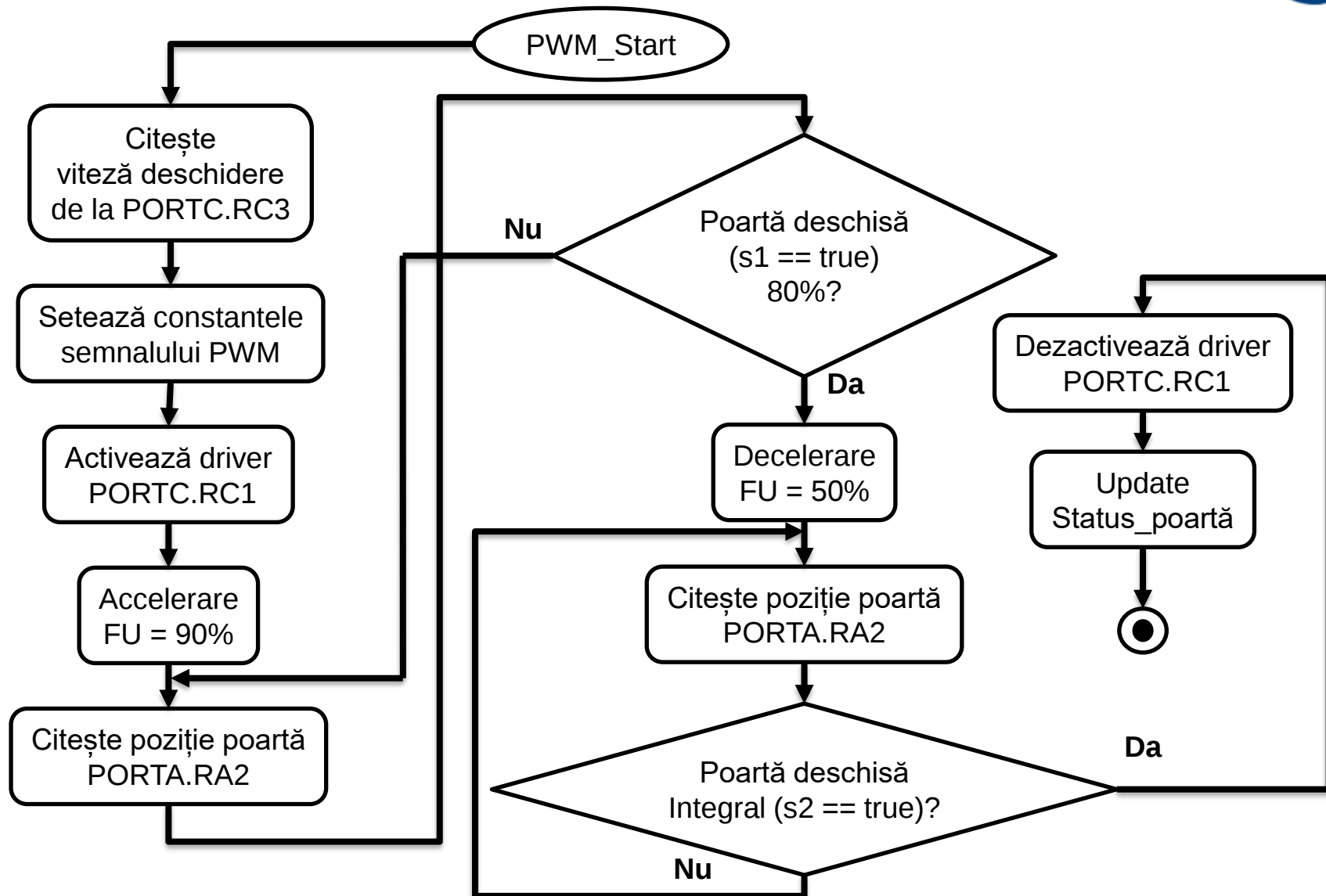


# Exemplu – activitatea principală





# Exemplu – activitatea principală





# Scrierea și testarea primară a codului

- Diagrama detaliată a tuturor subsistemelor este finalizată
- Codul în limbajul C sau asamblare este implementat
  - Se au în vedere avantajele/dezavantajele ambelor limbaje
- Validarea corectitudinii codului → testarea
  - Nu este indicat să testăm direct pe echipamentul HW al sistemului (poate fi distrus)
  - Se utilizează simulatoare pentru verificarea funcționării
  - Se implementează HW simplu care imită comportamentul celui real: butoane, leduri, osciloscop, debugging pe PC sau LCD
- Ex: intrările cu butoane, ieșirile cu LED-uri, semnale de reacție cu potențiometre (analog) sau butoane (digital), PWM cu osciloscop



# Planul de testare

- După ce un modul low-cost de testare a fost implementat este necesar un plan detaliat de testare
- Testele trebuie dezvoltate astfel încât:
  - **Să îndeplinească toate cerințele din caietul de sarcini**
  - **Să satisfacă criterii de performanță într-un mediu real**
  - **Să verifice funcționarea în scenarii accidentale**
- După fiecare etapă de testare trebuie elaborată documentația corespunzătoare
- După eventuale erori testele trebuie rulate din nou
  - **Este posibil ca noi teste să fie incluse după modificările făcute**





# Planul de testare

- După testele de funcționare corectă ale sistemului (pe platforma de simulare) este necesară o testare atipică
  - Se încearcă blocarea sistemului
  - Se încearcă combinații ale mărimilor de intrare care nu au fost prevăzute în proiectarea inițială
    - Sistemul trebuie să fie intolerabil la astfel de modificări
- După terminarea tuturor testelor pe modulul de simulare, se realizează aceleași teste și pe HW real
- Etapa de testare poate include și etape de trimitere a modelelor de simulare către firma contactată pentru a fi testată și de aceasta





# Documentarea

- O parte din documentație este deja disponibilă (etapele anterioare)
- Documentația trebuie să cuprindă:
  - **Descrierea sistemului, a cerințelor (caietul de sarcini), diagramele bloc, diagramele UML, planul de testare, rezultatele testelor, codul documentat**
- Codul trebuie documentat, descrierea funcțiilor (claselor dacă este cazul), parametrii de intrare, de ieșire, valori returnate
  - **Algoritmii din interiorul funcțiilor**
  - **Numele variabilelor și funcțiilor trebuie să fie sugestive**
  - **Se pot utiliza notații consacrate pentru nume de variabile (de ex. în funcție de tipul lor: `int nContor = 0;`)**



**Contact:**

**Email:** [tiberiu.cocias@unitbv.ro](mailto:tiberiu.cocias@unitbv.ro)

**elearning.unitbv.ro – Microcontrollere si Microprocesoare**