INGINERIA PROGRAMELOR

LUCRAREA DE LABORATOR NR.2

UTILIZAREA LIMBAJULUI DE MODELARE *UML*

-Diagrama cazurilor de utilizare -

Utilizarea limbajului de modelare UML
Partea I. Diagrama cazurilor de utilizare

1. Introducere

Elaborarea cerințelor specifice pentru fiecare aplicație este o etapă foarte importantă a procesului de dezvoltare software. Prin intermediul acestei etape se realizează o legătura directă între beneficiar și echipa de realizare a produsului. Așa cum a fost prezentat în prima lucrare de laborator cerințele beneficiarului (cerințe utilizator și cerințe de sistem) sunt exprimate în limbaj natural.

Înainte de a putea implementa un program (de a scrie efectiv codul aplicației) se impune identificarea și modelarea tuturor scenariilor de utilizare posibile și a tuturor relațiilor între utilizatorul aplicației și sistem. Această operație se realizează prin utilizarea unui limbaj grafic destinat modelării denumit **UML – Unified Modeling Language** *(limbajul unificat de modelare).*

2. Limbajul UML

2.1. Generalități

Limbajul unificat de modelare (engl. Unified Modeling Language), UML, este un limbaj pentru specificarea, vizualizarea, construirea și documentarea elementelor sistemelor software, însă poate fi folosit și pentru alte sisteme, cum ar fi cele de modelare a afacerilor. UML reprezintă o colecție de practici inginerești optime, care au fost încununate de succes în modelarea sistemelor mari și complexe.

UML 1.0 a fost propus spre standardizare în cadrul OMG (Object Management Group) în ianuarie 1997. Până la sfârșitul anului 1997 echipa care lucra la UML s-a extins, urmând o perioadă în care UML a primit o specificare formală mai riguroasă. Versiunea UML 1.1 a fost adoptată ca standard de către OMG în noiembrie 1997. În martie 2003 a fost publicată versiunea 1.5. În momentul de față a apărut la versiunea 2.0.

În UML există numeroase diagrame (modele), aceasta favorizând existența mai multor puncte de vedere privind sistemul. După cum am văzut, procesul de dezvoltare software are multe componente, fiecare cu propria sa perspectivă: analiști, proiectanți, programatori, testeri, echipe de asigurarea calității, autori ai documentației, clienți. Fiecare este interesat de un alt aspect al sistemului, la un nivel diferit de detaliu. De exemplu, programatorul trebuie să înțeleagă arhitectura sistemului pentru o converti în cod de nivel scăzut. Dimpotrivă, autorul documentației trebuie să înțeleagă comportamentul global al sistemului pentru a ști cum funcționează produsul.

UML încearcă să rezolve problema modelării la toate aceste nivele de detaliu.

2.2. Tipuri de diagrame UML

Notațiile limbajului universal de modelare sunt în special elemente grafice. Folosind notațiile specifice se obțin elementele de bază ale modelului UML denumite *diagrame.* În continuare sunt enumerate principalele tipuri de diagrame:

- Diagrame specifice etapei de analiză
  - Diagrama cazurilor de utilizare
  - Diagrama de activități
- Diagrame specifice etapei de proiectare
  - Diagrama de clase
  - Diagrama pachetelor

- Diagrama de stări
- Diagrama de interacțiuni
  - Diagrama de secvențe
  - Diagrama de colaborare

❖ Diagrame specifice etapei de implementare
- Diagrama de componente
- Diagrama de lansare

3. Diagrama cazurilor de utilizare (Use case diagram)

Diagrama cazurilor de utilizare este un instrument UML foarte puternic care descrie ceea ce sistemul va executa în cadrul procesului său de funcționare. Prin construirea unei colecții de diagrame de cazuri de utilizare, putem descrie întregul sistem într-o manieră clară şi concisă.

Diagrama cazurilor de utilizare (UCD) este folosită în special pentru a traduce cerințele sistemului într-un limbaj unificat, ușor de înțeles de toți membrii echipei, reprezentând un model inițial conceptual al sistemului.

Esența acestei diagrame constă în faptul că: sistemul proiectat se reprezintă ca o colecție de entități și actori care colaborează cu sistemul cu ajutorul așa numitor cazuri de utilizare. Totodată orice entitate care colaborează cu sistemul din afară poate fi numită *actor*. Aceasta poate fi un om, o instalare tehnică, un program sau oricare alt sistem care poate fi sursă de acțiune pentru sistemul proiectat. La rândul său, *use case-ul* este creat pentru descrierea serviciilor pe care sistemul le oferă actorului. Cu alte cuvinte fiecare caz de utilizare definește o colecție de acțiuni executate de sistem în timpul dialogului cu actorul. Este important de menționat faptul că diagrama nu prezintă modul în care se realizează colaborarea între actori și sistem sau modul în care sunt implementate anumite funcții ale sistemului.

O altă definiție poate fi că diagrama cazurilor de utilizare reprezintă un caz special de graf, în care sunt prezentate cazurile de utilizare concrete, actorii și legăturile între aceste elemente.

UCD poate conține următoarele tipuri de obiecte care vor fi prezentate succint în continuare:

- **Cazuri de utilizare;**
- **Actori**;
- **Relații**.

Atenție: Atât *actorii* cât și *cazurile de utilizare* trebuie să aibă denumiri unice!

3.1. Cazul de utilizare (Use Case)

Cazul de utilizare reprezintă o descriere a unei mulțimi de secvențe de acțiuni (incluzând variante) pe care un program le execută atunci când interacționează cu entitățile din afara lui (actori) şi care conduc la obținerea unui rezultat observabil.

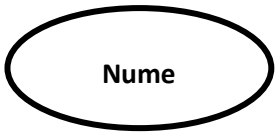


Figura 1. Caz de utilizare

Diagrama cazurilor de utilizare poate fi completată cu text explicativ, care detaliază sensul sau semantica componentelor ce o compun. Acest text se numește adnotare sau scenariu.

Fiecare caz de utilizare:

- se simbolizează cu o elipsă în interiorul căreia se notează denumirea prescurtată sau numele sub forma unui verb, așa cum este ilustrat în figura 1.
- reprezintă o funcționalitate a sistemului;
- precizează ce face un program sau un subprogram;
- nu precizează cum se implementează o funcționalitate.

Exemple de cazuri de utilizare pot fi acțiunile următoare:

- verificarea stării contului curent al clientului;
- întocmirea unei facturi la o comandă dată într-un magazin on-line;
- generarea unui raport cu notele studenților unei grupe;
- obținerea informației suplimentare despre solvabilitatea clientului;
- reprezentarea unei forme grafice pe ecranul monitorului.

3.2. Actorul

Actorul reprezintă un rol pe care utilizatorii unui caz de utilizare îl joacă atunci când interacţionează cu acesta. Un actor poate fi un utilizator uman, un sistem hardware sau un sistem software.

Fiecare actor:

- este o entitate exterioară sistemului;
- interacţionează cu sistemul:
  - iniţiază execuţia unor cazuri de utilizare;
  - oferă funcţionalitate pentru realizarea unor cazuri de utilizare;

Notaţia pentru un actor este prezentată în figura 2. Numele unic indică rolul pe care actorul îl joacă în interacţiunea cu cazul de utilizare.

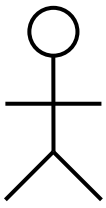


Figura 2. Actor

Identificarea actorilor se face răspunzând în principal următoarelor întrebări:

- Cine dorește sau este interesat de informațiile aflate în sistem?
- Cine modifică date?
- Cine interacționează cu sistemul?

Exemple de actori:

- Clientul unei bănci;
- Angajatul unei bănci;
- Vânzătorul de la un magazin;

- Pasagerul unui autocar sau al unui avion;
- Secretarul unei facultăți;
- Un cititor de coduri de bare;
- Un program care declanșează o alarmă sau o acțiune pre-programată.

3.3. Relația

Între elementele diagramei cazurilor de utilizare se pot stabili diverse relații, care descriu modul de comunicare între elementele pe care le conectează.

Tipuri de relaţii:

- ***actor - caz de utilizare***:
  - **asociere**: direcţia de navigare a relaţiei (săgeata) sugerează cine iniţiază comunicarea (figura 3).
- ***actor – actor***:
  - **generalizare**: semnifică faptul că un actor poate interacţiona cu sistemul în toate modalităţile prin care interacţionează un altul. Se reprezintă ca o relaţie de extindere între două cazuri de utilizare fără a avea stereotip (figura 4).
  - **dependenţă**: semnifică faptul că, pentru a interacţiona cu sistemul informatic prin intermediul unui caz de utilizare, un actor depinde de alt actor. Se reprezintă printr-o linie punctată având la un capăt o săgeată (figura 4).
- ***caz de utilizare – caz de utilizare***:
  - **dependenţă**: are loc între un caz de utilizare şi oricare alt caz de utilizare ce foloseşte funcţionalitatea primului. Se reprezintă grafic printr-o linie având la capătul corespunzător cazului de utilizare folosit un triunghi si este etichetat cu stereotipul *<<include>>* (figura 5).
  - **generalizare**: este folosită pentru a sugera un comportament opţional, un comportament care are loc doar în anumite condiţii sau fluxuri diferite ce pot fi selectate pe baza selecţiei unui actor. Reprezentarea grafică este similară cu cea a relaţiei de dependenţă, dar eticheta este *<<extend>>* (figura 5).

Pentru majoritatea sistemelor, un anumit actor poate interacţiona cu mai multe cazuri de utilizare, iar un anumit caz de utilizare poate fi iniţiat de actori diferiţi (figura 3).

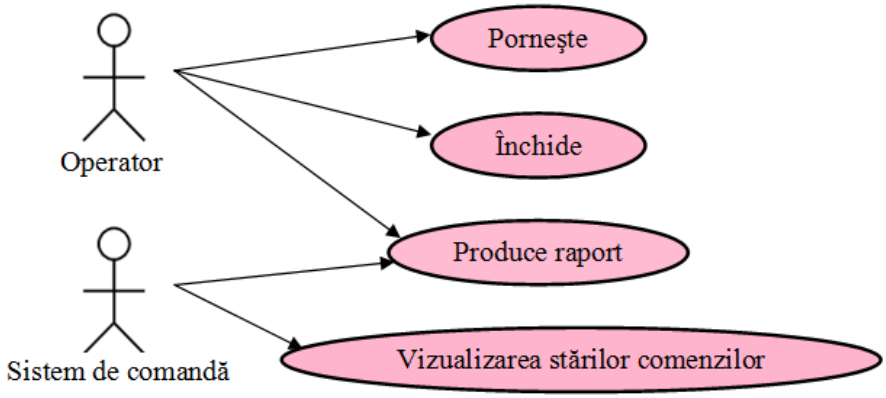


Figura 3. Cazuri de utilizare cu multiplii actori

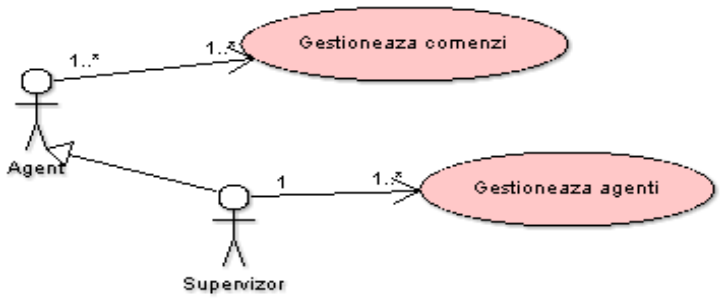


Figura 4. Reprezentarea grafică a relaţiilor de generalizare şi dependenţă între actori

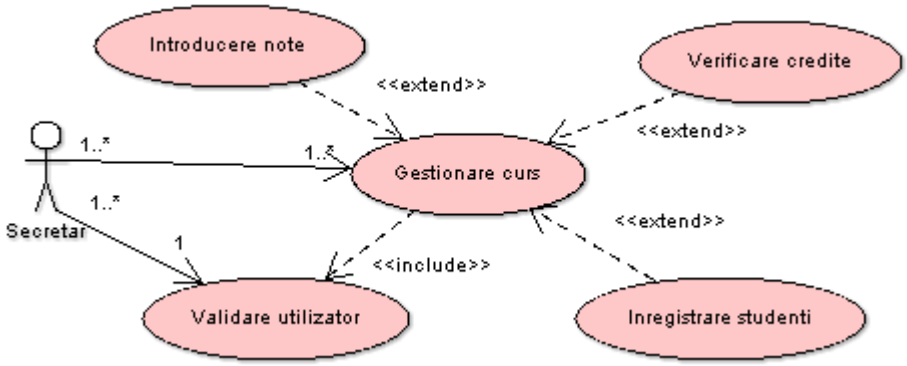


Figura 5. Reprezentarea grafică a relaţiilor de generalizare şi dependenţă între cazuri de utilizare

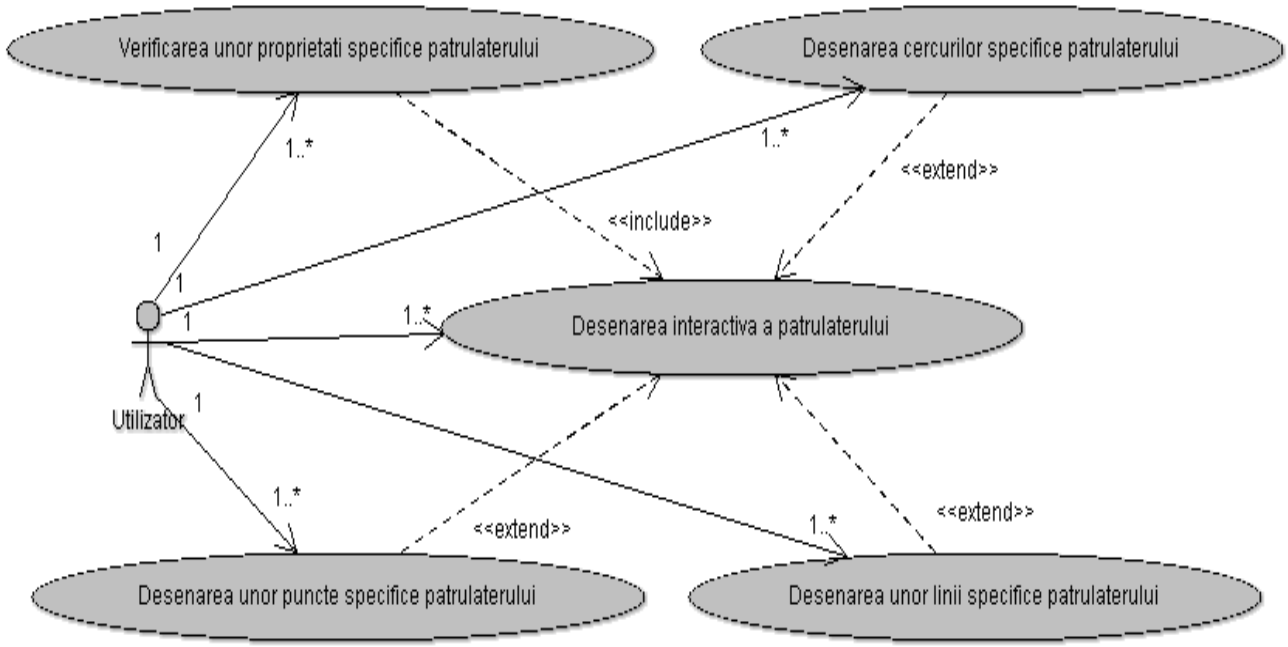
Deşi par foarte simple, ignorarea diagramelor cazurilor de utilizare este o mare greşeală. Acestea sunt foarte importante deoarece:

- definesc domeniul sistemului, permiţând vizualizarea dimensiunii şi sferei de acţiune a întregului proces de dezvoltare;
- sunt similare cerinţelor, dar cazurile de utilizare sunt mai clare şi mai precise datorită structurii riguroase de notaţie;
- suma cazurilor de utilizare este sistemul ca întreg; ceea ce nu este acoperit de un caz de utilizare se situează în afara sistemului de construit;
- permit comunicarea dintre client şi dezvoltatori, de vreme ce diagrama este foarte simplă şi poate fi înţeleasă de oricine;
- ghidează echipele de dezvoltare în procesul de dezvoltare;
- ajută echipele de testare şi autorii manualelor de utilizare.

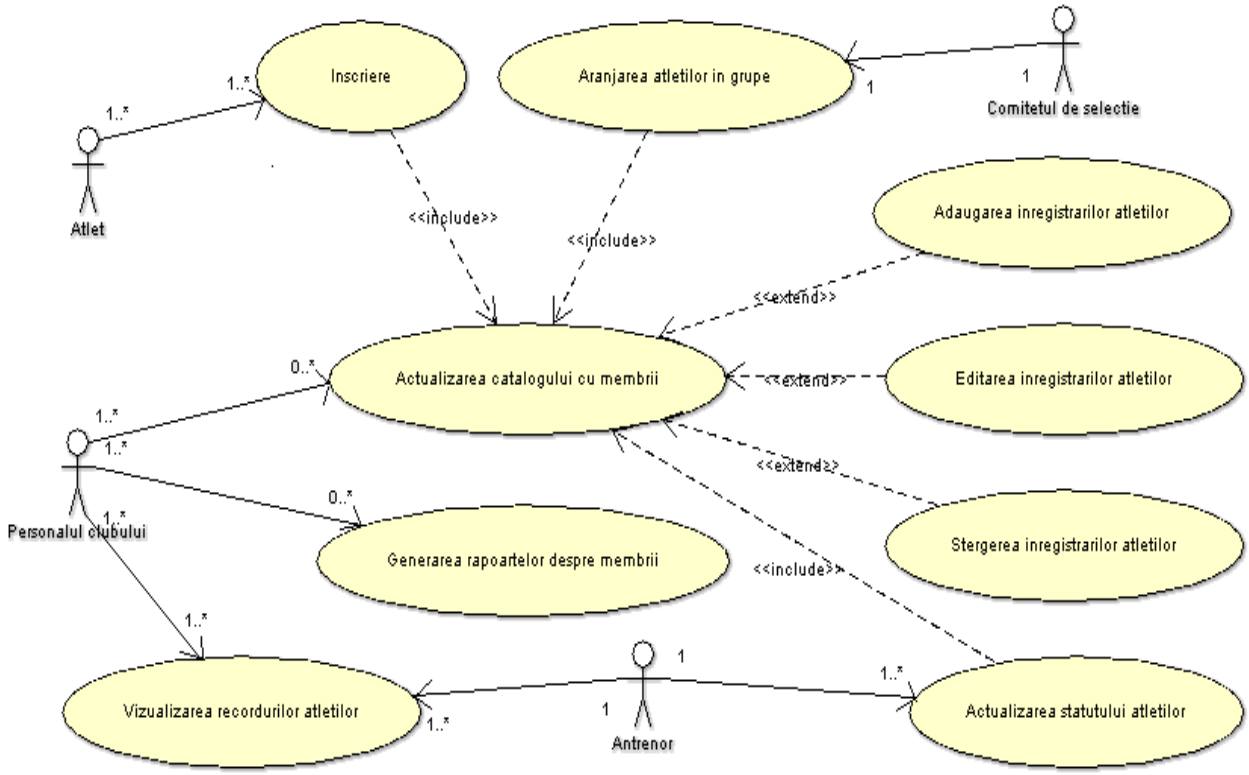
4. Exemple de diagrame – UCD

1. Se doreşte dezvoltarea unui sistemului informatic interactiv destinat studiului patrulaterului prin atingerea următoarelor scopuri:
   - desenarea interactivă a patrulaterului prin înlocuirea creionului şi a riglei cu *mouse*-ul;
   - verificarea unor proprietăţi specifice unui patrulater;
   - desenarea unor puncte, linii şi cercuri specifice unui patrulater.

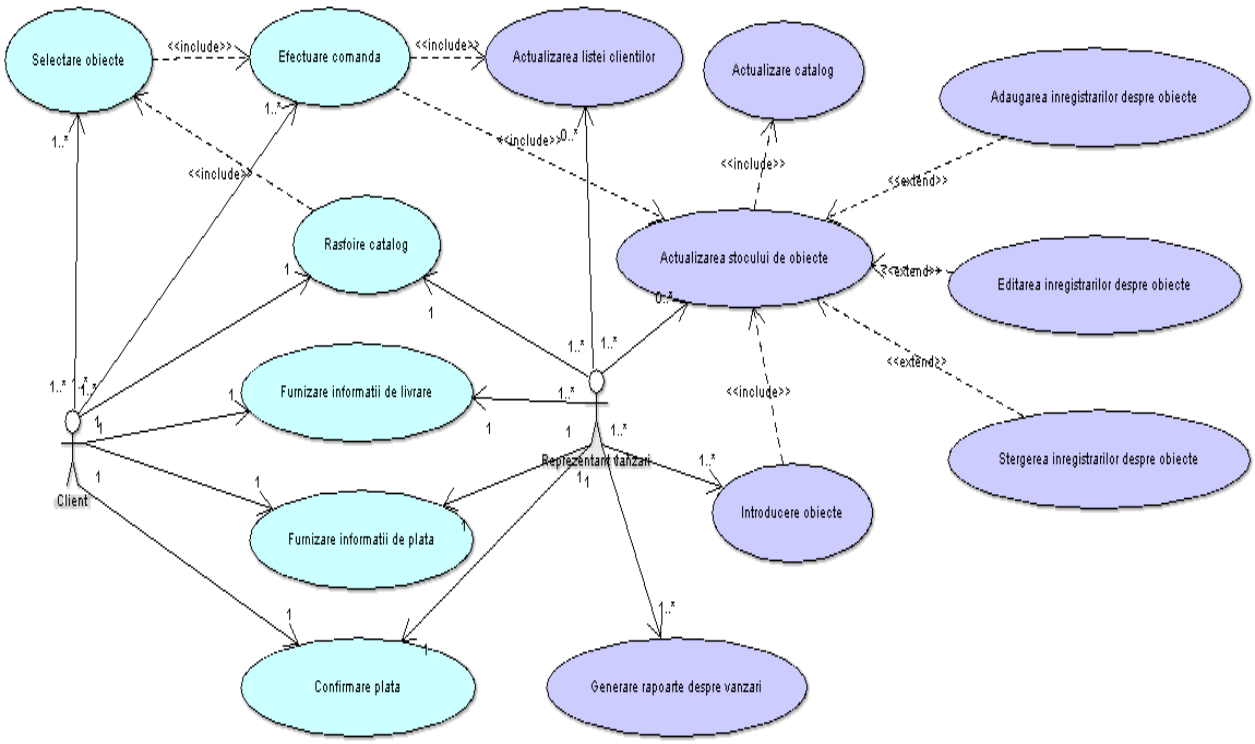
   Diagrama cazurilor de utilizare asociată acestui soft interactiv este prezentată în continuare:



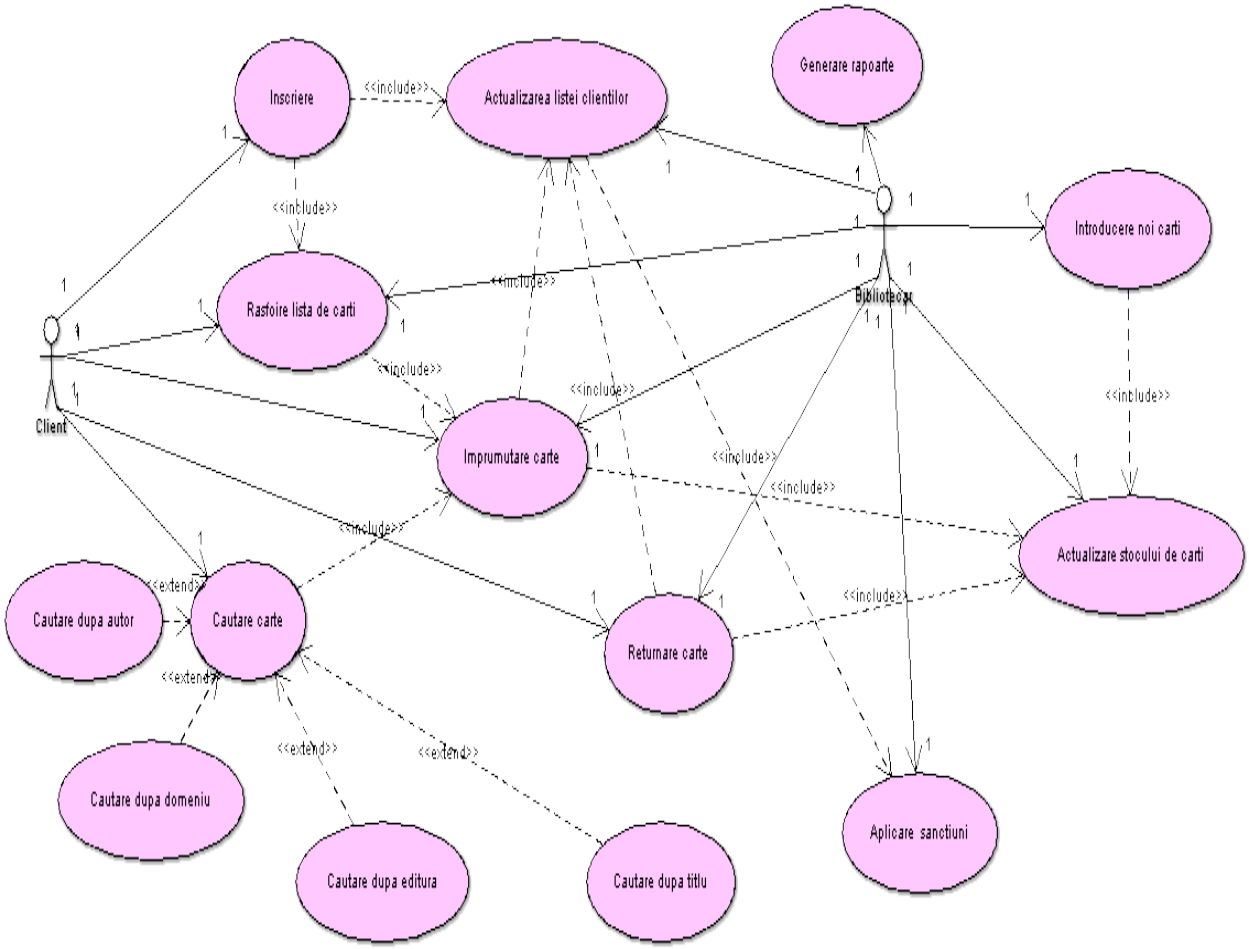
2. Să se dezvolte o aplicaţie ce permite gestionarea operaţiilor dintr-un club de atletism.



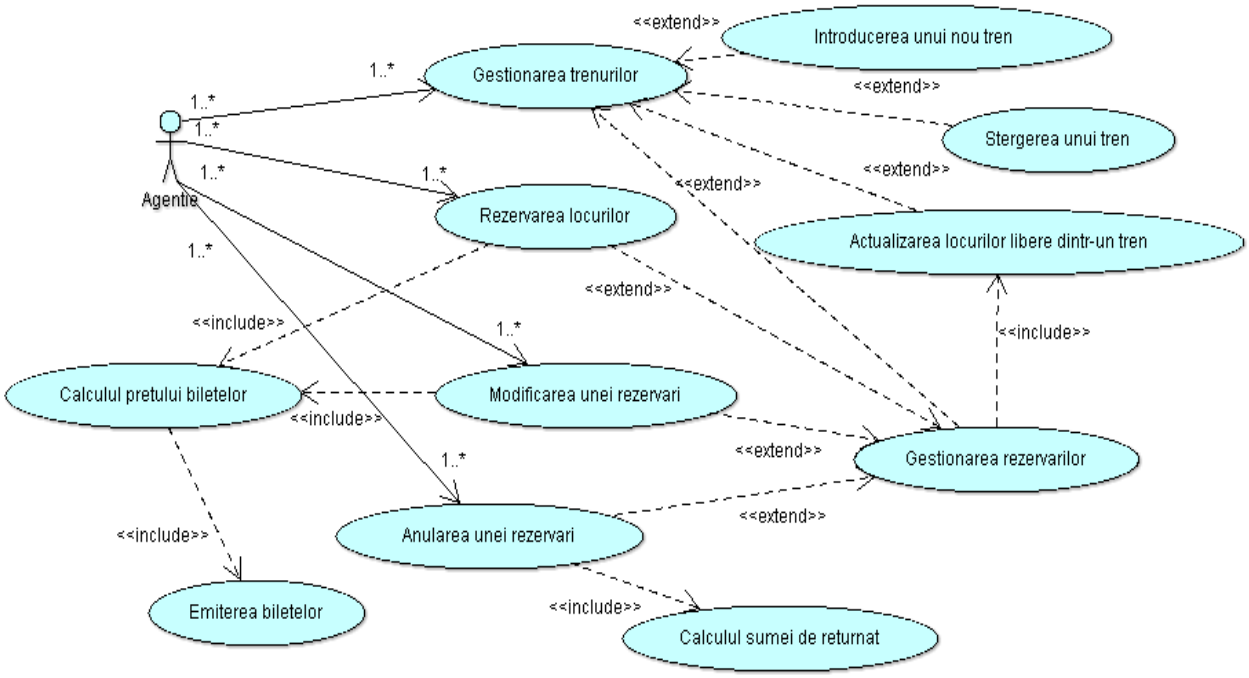
3. Să se dezvolte o aplicaţie ce permite gestionarea operaţiilor dintr-un magazin online.



4. Să se dezvolte o aplicaţie ce permite gestionarea cărţilor dintr-o bibliotecă.



5. Să se dezvolte o aplicaţie ce permite rezervarea de locuri la o companie feroviară într-o agenţie de voiaj.



5. Probleme propuse

1. O noua companie care fabrică telefoane mobile vă angajează pentru a scrie programul care controlează funcționalitatea telefonului. Programul trebuie să ofere cel puțin următoarele funcții:
   - apelarea unui număr (format de utilizator, din agenda);
   - preluarea unui apel;
   - scrierea/citirea unui mesaj ;
   - introducerea unui număr în agendă;
   - ștergerea unui număr din agendă;
   - vizualizarea apelurilor ratate.

   Dezvoltați un proiect care să descrie modul de interacţiune al unui utilizator cu telefonul.
2. Dezvoltați un proiect care să descrie interacțiunea cu un utilizator al unui automat de băuturi. Exemplu de facilităţi:
   - oferire ceai;
   - oferire cafea;
   - selectare nivel de zahar;
   - selectare lapte (doar pentru cafea);
   - gestionare sumă utilizator, oferire rest ;
   - anularea comenzii.
3. Creați UCD pentru aplicațiile pentru care ați creat cerințele în laboratorul anterior.