Inginerie Software 2024
Laboratorul 8
Modelare interacțiuni. Diagrama de secvențe*

Mădălina Erașcu, Alexandru Munteanu, Cristian Pal, Ionica Puiu

April 2024

1 Diagrama de secvențe

1.1 Introducere

Clasele definesc tipurile de comportament pe care le realizează obiectele.

Întregul comportament în sistemul orientat obiect este realizat de obiecte, nu de clase. Obiectele preiau responsabilitatea editatării datelor, memorării datelor, transferurilor de date din sistem, răspunsurilor la interogări, protejării sistemului, etc.

Obiectele colaborează pentru a-și îndeplini sarcinile comunicând unele cu altele. Examinarea modurilor în care anumite obiecte funcționează în anumite situații dezvăluie natura exactă a acomunicării.

Cu diagrame de secvențe se modelează în mod tipic:

1. *Scenarii ale cazurilor de utilizare.* Un scenariu de utilizare este o descriere a unuia din modurile în care poate fi folosit sistemul. Logica unui scenariu de utilizare poate fi o parte a unui caz de utilizare: fluxul principal, flux alternativ sau combinație de fluxuri. Poate fi, de asemenea, o combinație de mai multe cazuri de utilizare.

2. *Logica metodelor* este logica unei operații, funcții sau proceduri complexe. În acest caz, diagrama de secvențe este cod obiect vizual.

3. *Logica serviciilor.* Un serviciu este de fapt o metodă de nivel înalt, invocabilă de către mai mulți clienți.

Diagrama de secvențe:

- aparține categoriei de diagrame comportamentale din UML.

*Bazat pe resursele de laborator ale Conf. Dr. Cristina Mândruță

- ilustrează interacțiunile dintre obiecte; modelează obiecte și mesaje între obiecte.

- reprezentare orientată pe timp; utilizează o pictogramă și o linie de viață asociată pentru fiecare obiect.

Interacțiunile arată cum mesajele sunt transmise de la un obiect la altul. Dacă un obiect intenționează să trimită un mesaj altui obiect, acesta din urmă trebuie să aibă o modalitate de a-l recepționa. Mesajul trebuie să corespundă unei interfețe oferită de al doilea obiect.

Această împerechiere dintre mesaje și interfețe are o dublă utilitate pentru construirea și pentru testarea modelului:

1. Dacă un obiect trebuie să trimită un mesaj la un obiect țintă, trebuie verificat dacă obiectul țintă oferă interfața necesară.

2. Poate ajuta în descoperirea de noi elemente ale modelului. Dacă obiectul țintă nu are interfața corespunzătoare, atunci descoperim o nouă cerință de interfață. Dacă nu există un obiect țintă care să preia responsabilitatea primirii mesajului, atunci descoperim necesitatea unui nou tip de obiect, adică o nouă clasă.

Diagrama de secvențe oferă o cale de la descrierea textuală a comportamentului sub formă de scenarii la operațiile/interfețele necesare pe diagrama de clase.

Scenariile oferă, de asemenea, o bază pentru dezvoltarea cazurilor de testare și a unui plan de teste de acceptare.

1.2 Elemente de modelare

- **Obiect** - instanță a unei clase, care participă la un scenariu descris de diagramă.

- **Mesaj** – definiție a unui tip de comunicare între obiecte de un anumit tip. Poate invoca o operație, lansa un semnal sau determina crearea sau distrugerea unui obiect țintă.

- **Linia de viață a obiectului** – notație verticală a perioadei de timp în care obiectul există. Poziționarea unui obiect în partea superioară a diagramei arată că obiectul există deja înainte de începerea scenariului. Un obiect creat în cursul unui scenariu va fi poziționat mai jos, într-o poziție corespunzătoare momentului creării lui.

- **Activare** – reprezentarea perioadei de timp în care obiectul este activ, executând o metodă.

1.3 Notație

- **Obiect** – casete dreptunghiulare cu 2 nume. Numele sau numărul instanței și numele clasei apar în forma `obiect:clasa` (Figura 1). Oricare dintre nume poate fi omis. Se poate folosi și reprezentarea specifică diagramei de robustețe (Figura 2).

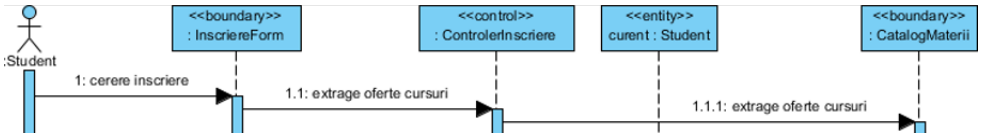


Figure 1: Notație Obiect (I)

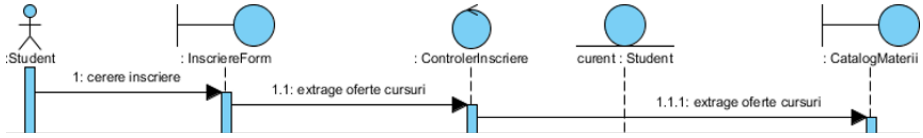


Figure 2: Notație Obiect (II)

- **Mesaj** – săgeată orizontală între obiecte.

Tipul săgeții descrie vizual tipul mesajului. Sunt plasate orizontal între liniile de viață ale obiectelor.

- O săgeată oblică indică o întârziere între trimitrerea și recepția mesajului.
- Poziția vericală relativă reprezintă ordinea apariției mesajelor.

1.4 Detalii de modelare

Sintaxa mesajului Mesajul este etichetat în special cu operația care definește interfața necesară obiectului receptor, către care indică săgeata.

Eticheta definește numele operației și opțional parametrii și tipul returnat. Parameterii sunt reprezentați în forma `nume:tip` și se pot specifica chiar și valori implicite și constrângeri.

Tipuri de mesaje (Figura 3)

- **Mesaj sincron**: emițătorul așteaptă mesajul de return înainte de a-și continua activitatea.
- **Mesaj return**: răspuns la un mesaj. Pentru a simplifica diagrama, se poate omite reprezentarea unor mesaje de return.
- **Mesaj self-reference**: emițătorul și receptorul sunt același obiect.

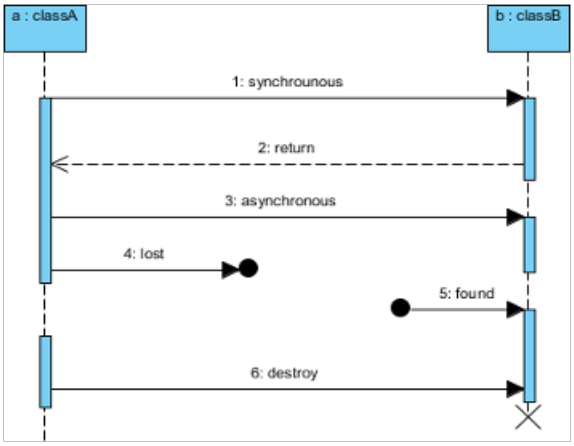


Figure 3: Principalele tipuri de mesaje

- **Mesaj asincron (semnal)**: emițătorul nu este obligat să aștepte răspunsul receptorului. Receptorul poate decide să nu facă nimic sau să proceseze mesajul.
- **Mesaj temporizat**: o condiție sau o constrângere asupra unui mesaj care exprimă parametrii de temporizare a acestuia.

Un exemplu de **modelare iterație pe un mesaj** este în Figura 4

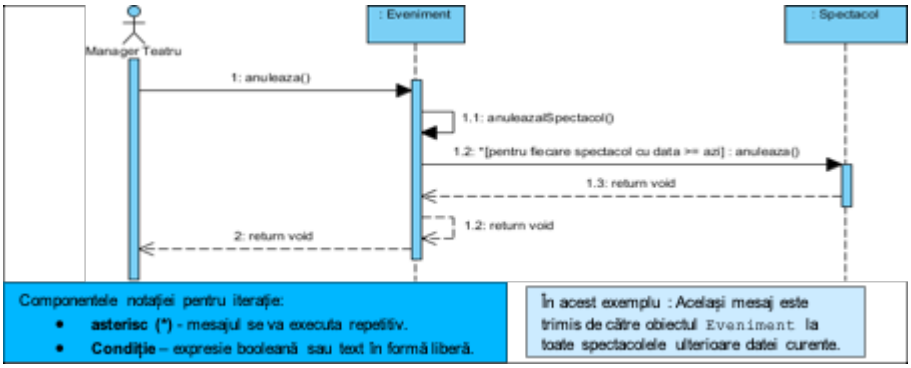


Figure 4: Modelare iterație pe mesaj

Remarcă. *Executarea repetitivă a unui mesaj implică executarea tuturor mesajelor subordonate la fiecare repetare.*

Pentru a explica fluxul mesajelor se pot modela condiții. În Figura 5, mesajul 3 spune că dacă locul care a fost selectat în mesajul 2 este liber, atunci mesajul 3 va returna o referință la obiectul ce reprezintă locul selectat. Fluxurile alternative se pot modela cu condiții (Figura 6).

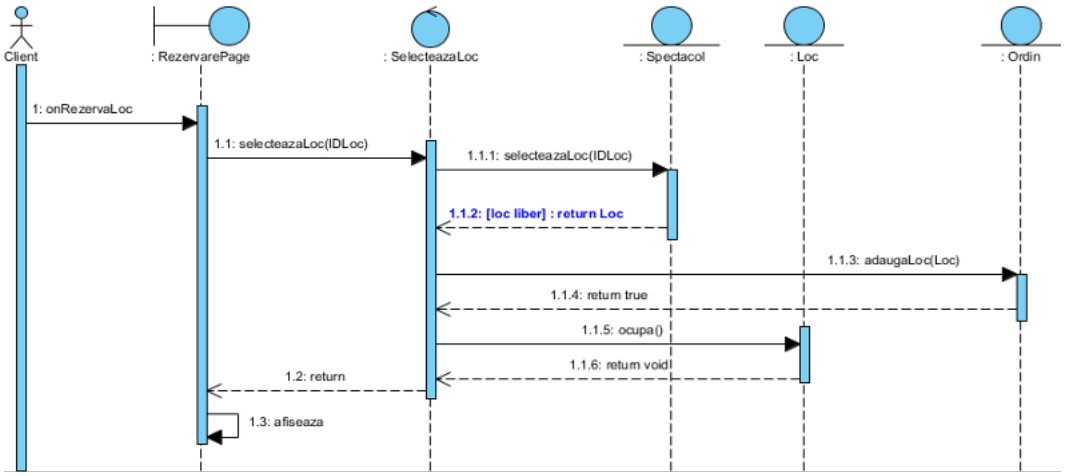


Figure 5: Modelare condiții

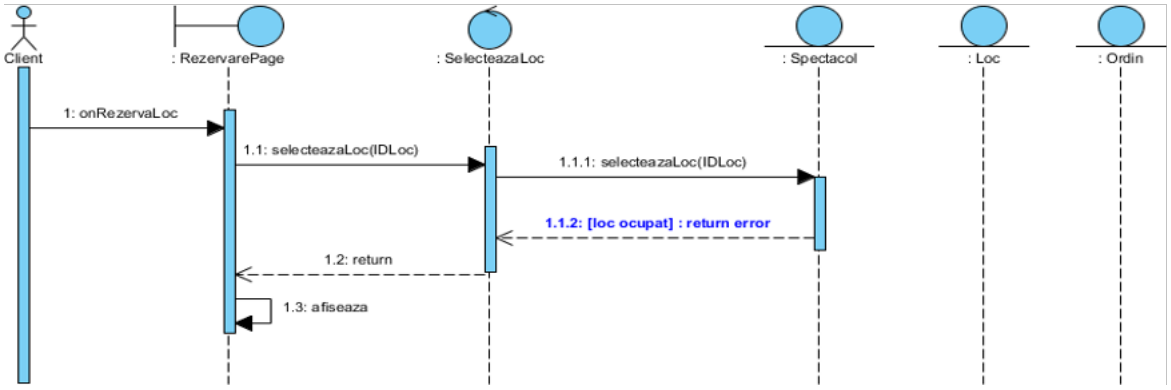


Figure 6: Modelarea scenariului alternativ utilizând o condiție ce explică variația

UML oferă, de asemenea, posibilitatea de a utiliza tipuri specifice de **frames (fragmente)** pentru reprezentarea căilor alternative, a acțiunilor repetitive, sau a paralelismului.

Definiție. *Un frame este un context portabil pentru o diagramă. Poate fi reutilizat (inclus) în alt frame sau în altă diagramă.*

Exemple de fragmente:

- referință (Figura **??**)
- alternativă (Figura 7)
- repetiție, paralelism (Figura 7)

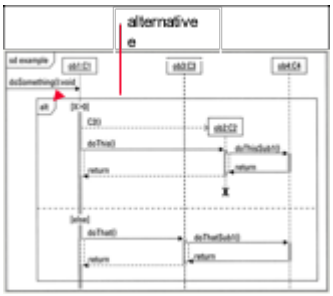


Figure 7: Exemplu frame `alt`

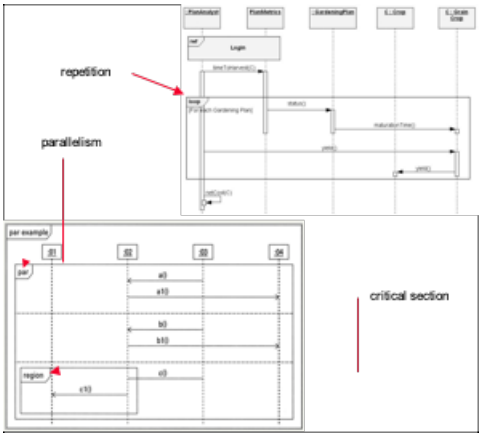


Figure 8: Exemplu frame `loop` și `par`

Modelarea activării (deținerea controlului)

- **Activare** – obiect activ, în cursul execuției unei operații – modelată cu un dreptunghi îngust pe linia de viață a obiectului.
- **Dezactivare** – obiect inactiv, în așteptarea unui mesaj – linie de viață punctată.

Unele obiecte pot păstra controlul întregului proces modelat de diagrama de secvențe. Un astfel de obiect face mai mult decât să răspundă la mesaje; el supervizează întregul set de interacțiuni.

Mesaj reentrant Execuțiile suprapuse de pe aceeași linie de viață sunt reprezentate prin dreptunghiuri suprapuse (Figura 9).

Mesaje de apel recursiv Această tehnică (Figura 10) este specifică realizării de acțiuni asupra ierarhiilor sau colecțiilor încuibate, cum ar fi locații la teatru.

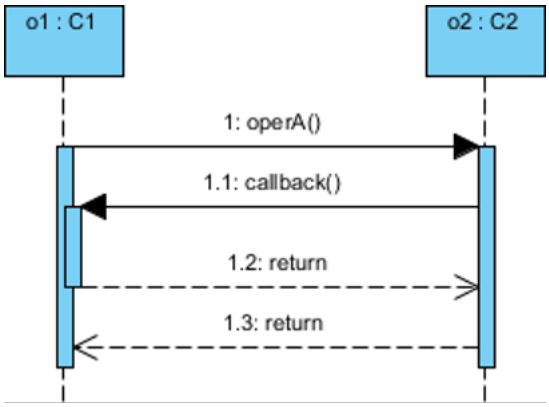


Figure 9: Exemplu mesaj reentrat

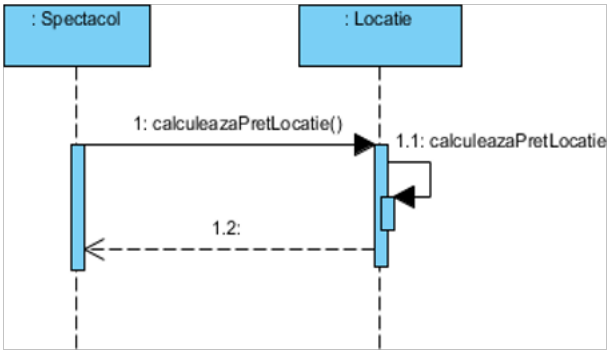


Figure 10: Exemplu mesaj recursiv

Exemplu. *Obiectul execută operația invocată de mesajul 3. Această operație se autoapelează astfel încât apare o nouă activare peste cea curentă. De exemplu, dacă o locație este o secțiune, atunci obiectul Locație apelează pretLocație pentru fiecare rând. Fiecare rând apelează pretLocatie pentru fiecare loc al rândului. Locurile nu conțin alte locații, deci apelul se oprește.*

Creare obiect Obiectul este creat prin apelul unuia din constructorii lui.

Exemplul modelează obiectul Eveniment care apelează constructorul Spectacol(eveniment), operația care crează o instanță a clasei Spectacol (Figura 11).

Distrugere obiect Un obiect terminat are un X plasat pe linia de viață corespunzător momentului distrugerii. Distrugerea se face de obicei după un apel de ștergere (delete) sau de anulare (cancel). Absența unui X pe linia de viață a obiectului arată că acesta supraviețuiește secvenței de evenimente descrisă în diagrama curentă.

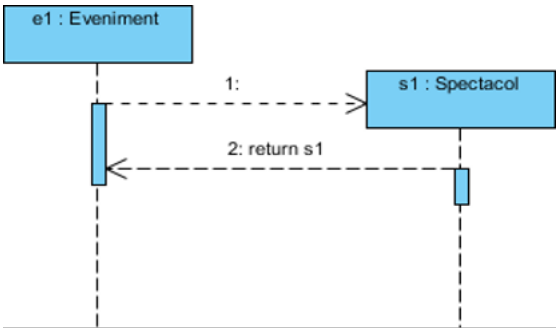


Figure 11: Exemplu creare obiect

2 Metodologie de realizare a unei diagrame de secvențe

Diagrama de secvențe este elementul central al părții dinamice a modelului obiect. După realizarea proiectării preliminare folosind analiza de robustețe, reluăm scenariile pentru a continua modelarea. Ele conduc alocarea comportamentului la clasele software (Figura 12).

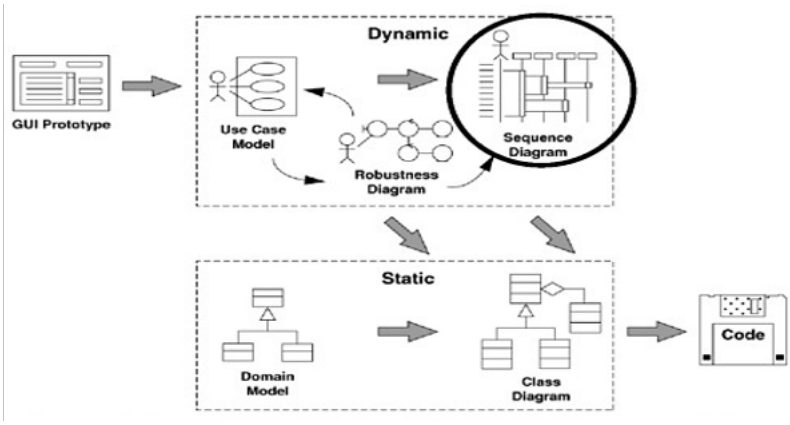


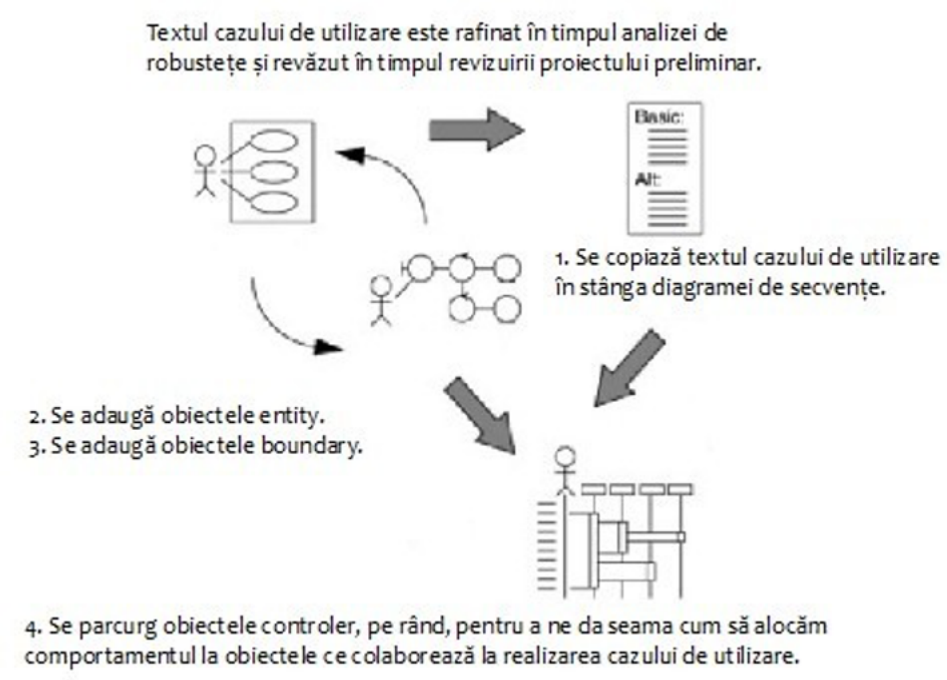
Figure 12: Procesul ICONIX

Principalele obiective ale modelării interacțiunilor:

- *Alocarea comportamentului la obiectele boundary, entity și control.* În timpul analizei de robustețe a fost identificat un set de obiecte care pot împreună să realizeze comportamentul dorit pentru cazul de utilizare. De asemenea, comportamentul a fost divizat în unități discrete sub formă de obiecte control. În continuare se va decide ce obiect este respionsabil pentru fiecare unitate de comportament.

- *Ilustrarea în detaliu a interacțiunilor care au loc în timp între obiectele asociate fiecărui caz de utilizare.* Obiectele interacționează la momentul execuției prin transfer de mesaje. Aceste mesaje servesc drept stimuli—adică un mesaj stimulează un obiect să execute o anumită acțiune. Pentru fiecare unitate de comportament din cadrul unui caz de utilizare trebuie identificate mesajele/operațiile corespunzătoare.
- *Distribuirea operațiilor la clase.*

2.1 Construirea unei diagrame de secvențe



1. Copiați textul cazului de utilizare într-o notă plasată în zona din stânga a diagramei.
2. Adăugați obiectele entity din diagrama de robustețe. Verificați totodată că acestea se regăsesc în diagrama de clase, și completați diagrama de clase dacă e cazul.
3. Adăugați obiectele boundary și actorii din diagrama de robustețe.

   Obiectele boundary fac parte din domeniul soluției; modelul domeniului adresează spațiul problemei. Reprezentarea obiectelor boundary pe diagrama de secvențe este primul pas în integrarea celor două spații.

4. Decideți ce operații sunt atribuite fiecărei clase; acest pas este esența modelării interacțiunilor.

   Alocarea operațiilor la clase presupune convertirea obiectelor control din diagrama de robustețe, pe rând, în seturi de operații și mesaje care cuprind comportamentul dorit. Uneori e posibil ca un controler să devină clasă de sine stătătoare.

Remarcă. *Un obiect control de pe diagrama de robustețe se poate traduce în mai multe operații și mesaje pe o diagramă de secvențe.*

3 Exemplu: Afișează detalii carte

Vezi Figurile 13 , 14, 15.

Text CAZ DE UTILIZARE

Identificator: UC 19
Descriere:
Permite explorarea listei de cărți și afișarea detaliilor pentru cea selectată.
Precondiții:
Clientul este logat în sistem.
Postcondiții:
Niciuna

Flux principal de acțiuni:
Clientul tipărește URL-ul pentru pagina Home a librăriei.
Sistemul afișează o listă de cărți din catalog pe pagina Home, sub formă de link-uri selectabile. Clientul selectează un link de pe pagina Home.
Sistemul extrage detaliile cărții selectate și le afișează pe pagina Detalii Carte.
Flux alternativ A: CarteNegasita
A.1. Sistemul afișează pagina CarteNegasita.

Figure 13: Descriere caz utilizare Afișează detalii carte

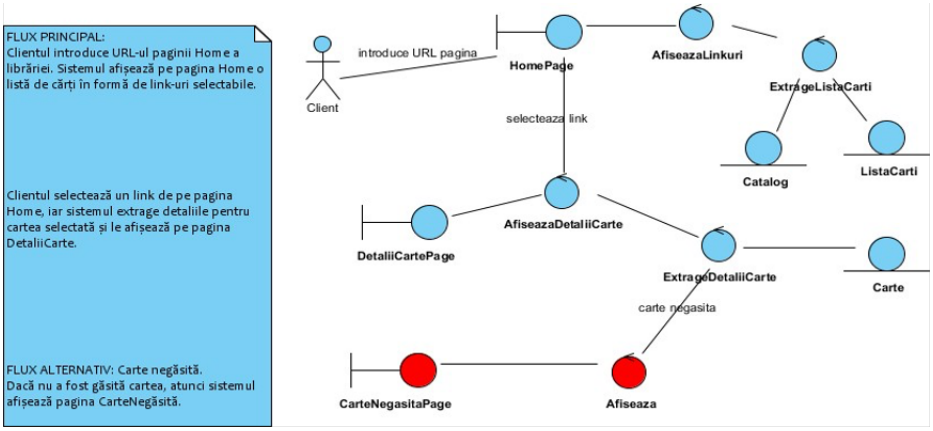


Figure 14: Diagrama de robustețe Afișează detalii carte

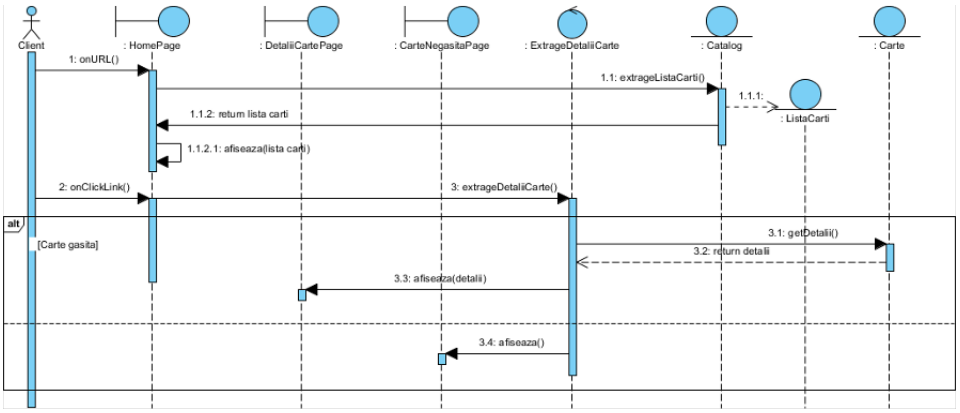


Figure 15: Diagrama de secvențe Afișează detalii carte

4 Recomandări

1. Realizați câte o diagramă de secvențe pentru fiecare caz de utilizare .

   După desenarea diagramelor de secvențe pentru toate fluxurile de evenimente din toate cazurile de utilizare, putem spune că am găsit toate rolurile pe care trebuie să le joace fiecare obiect în sistem, deci toate responsabilitățile fiecărui obiect.

2. În prealabil identificați toate obiectele necesare în cursul unei analize de robustețe.

   Diagrame de robustețe bune, asociate cu cazuri de utilizare riguros definite, simplifică realizarea diagramelor de secvențe.

3. Specificații detaliile de comunicare.

   Pe diagrama de robustețe comunicarea între obiecte este reprezentată la un nivel superior de abstractizare. Diagrama de secvențe însă trebuie să ilustreze interacțiunile între obiecte la un nivel ridicat de detaliere.

4. Nu transformați diagrama de secvențe într-o reprezentare de flux de activități; diagrama de secvențe este un instrument pentru alocarea comportamentului la obiecte.

   Diagramele de secvențe trebuie utilizate pentru asignarea de operații la clase, deci numele mesajului va trebui conectat cu numele unei operații a clasei din care este instanțiat obiectul țintă. Alocarea comportamentului – decizia referitoare la ce operații sunt executate de care clase – are o importanță critică în abordarea ICONIX. Deciiziile realizate în timpul acestei etape a proiectului fac diferența dintre un proiect bun și unul prost.

5. Concentrați-vă pe operațiile interesante, nu pe operațiile accesor (”getter” și ”setter”).

Prin explorarea comportamentului dinamic al sistemului sunt aflate atributele și operațiile necesare pentru clasele conținute în modelul dinamic. Adăugați atribute și operații claselor imediat ce puteți decide unde anume se potrivesc în contextul diagramelor de secvențe. Nu trebuie însă să insisteți pe desenarea mesajelor corespunzătoare apelurilor de tip "getAtribut" și "setAtribut.". Aceasta nu înseamnă că nu trebuie să respectați principiul încapsulării care impune accesul la atribute doar prin operații accesor.

6. Analizați cu atenție originile mesajelor (cu alte cuvinte, ce obiect deține controlul în fiecare moment).

   Dacă poe diagrama de robustețe nu este esențial să reprezentăm cu precizie săgețile, pe diagrama de secvențe reprezentarea lor corectă este esențială. Fluxul de control trebuie să fie explicit; trebuie să fie evident în fiecare moment ce obiect deține controlul.

7. Urmați principiile de bază ale proiectării OO condusă de responsabilități atunci când alocați comportament prin desenarea mesajelor.

   Un obiect (și, prin extensie, o clasă) ar trebui să aibă o singură "personalitate". Clasa trebuie să fie centrată pe un set puternic corelat de comportamente. Aceasta conduce la îndeplinirea dezideratului de a avea obiecte puternic coezive și slab cuplate. Alte principii sunt reutilizabilitatea (cu cât obiectele și clasele sunt mai generale, cu atât este mai mare probabilitatea ca acestea să poată fi reutilizate în alte proiecte) și aplicabilitatea (o operație asignată la un obiect trebuie să se "potrivească" cu acesta și să fie relevantă pentru obiectul respectiv).

8. Actualizați modelul static prin construirea de diagrame de clase locale pentru fiecare pachet de cazuri de utilizare.

   În continuarea activității de proiectare este necesară realizarea diagramei de clase care unifică spațiul problemei (representat în diagrama claselor de domeniu) cu spațiul soluției (descoperit în cursul proiectării de detaliu). În cazul proiectelor complexe se recomandă realizarea unei astfel de diagrame pentru fiecare pachet de cazuri de utilizare (grupate de obicei pe actori). Această metodă permite o mai bună gestionare a complexității și a distribuirii sarcinilor între echipe.

5 Exemple

Următoarele exemple, provenite din diagramele de secvențe ale modelului aplicației pentru librărie accesibilă pe Internet, au rolul de a ilustra aplicarea recomandărilor anterioare.

Exemplul 1 O diagrama de secvențe pentru cazul de utilizare *Căutare după autor* este în Figura 16. Varianta îmbunătățită este în Figura 17.

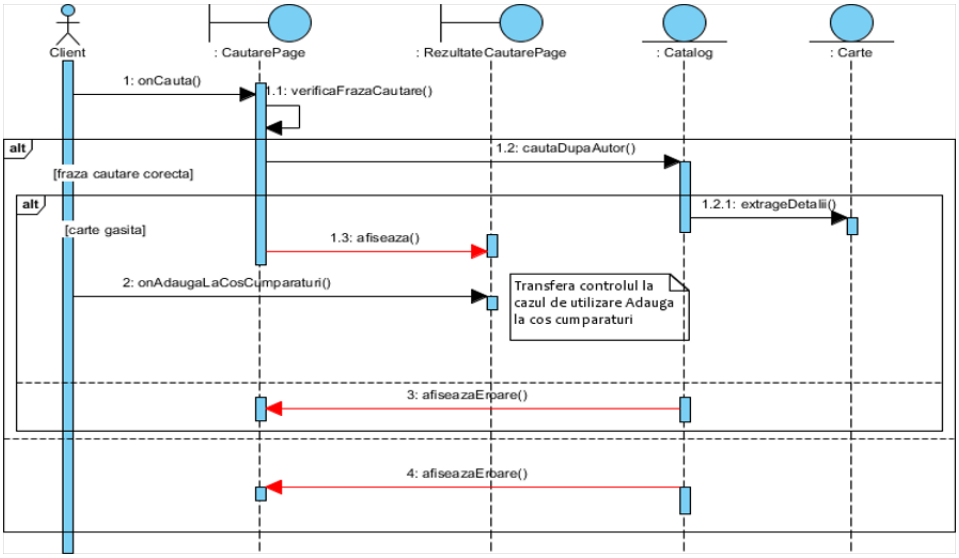


Figure 16: Diagrama de secvențe *Căutare după autor*

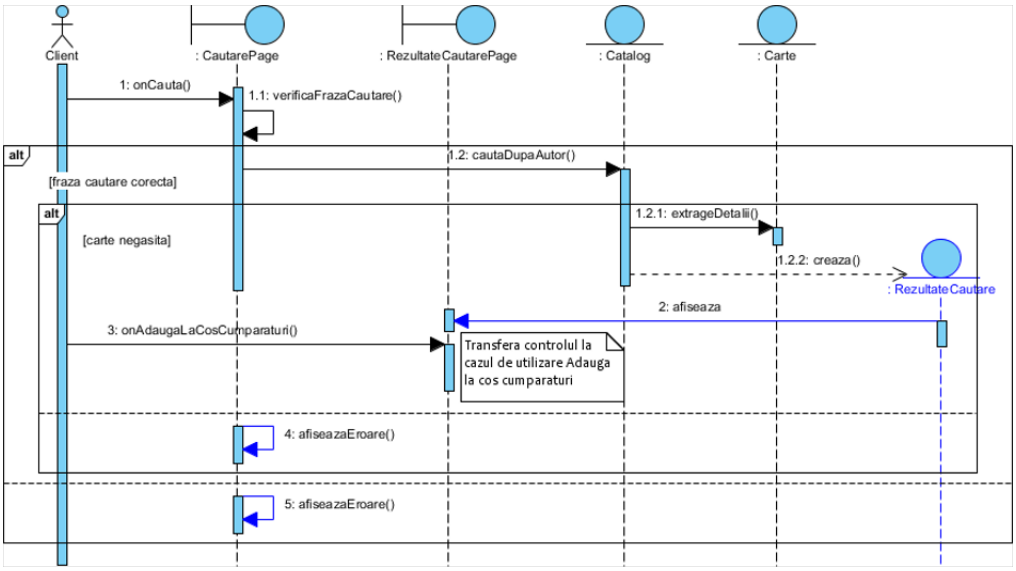


Figure 17: Diagrama de secvențe *Căutare după autor* (corectată)

Explicație. *În Figura 16, lipsește obiectul* `RezultateCautare`*. Acesta a fost descoperit în timpul analizei de robustețe.* `CautarePage` *trimitea un mesaj de afișare, deși este evident că obiectul* `Catalog` *deține controlul. Obiectul* `Catalog` *invocă operația* `afiseazaEroare` *pe* `CautarePage`*.*

Exemplul 2 Cazul de utilizare Login. O diagrama de secvențe pentru cazul de utilizare *Căutare după autor* este în Figura 18. Varianta îmbunătățită este în Figura 19.

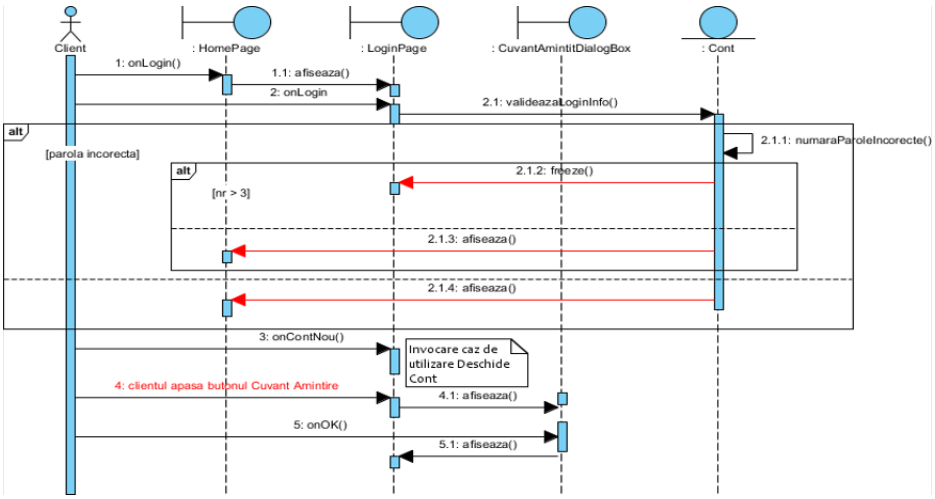


Figure 18: Diagrama de secvențe *Login*

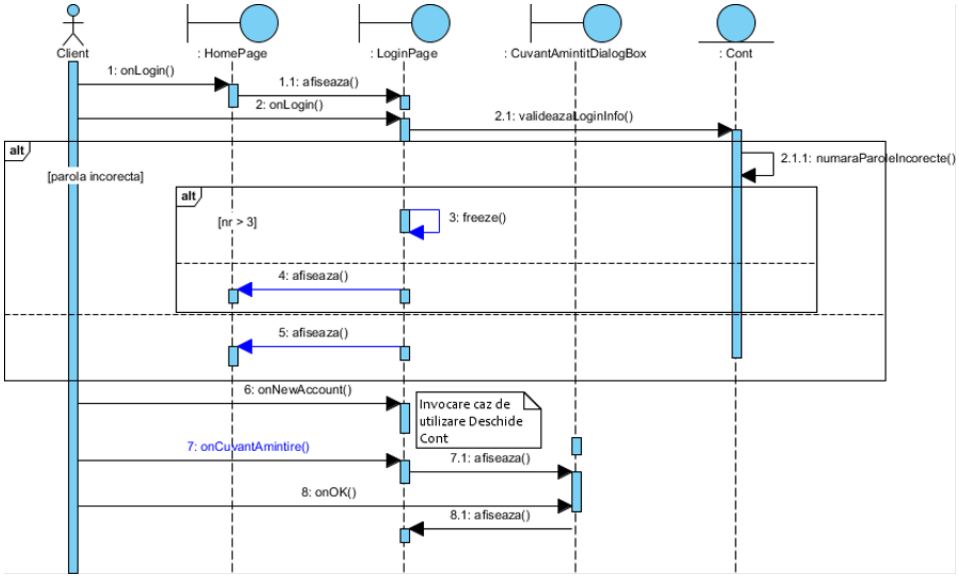


Figure 19: Diagrama de secvențe *Login* (corectată)

Explicație. *În Figura 18, obiectul* `Cont` *trimitea un mesaj de afișare chiar dacă diagrama arată că textttLoginPage deține controlul. Mesajul asociat cu fluxul*

alternativ "cuvânt amintit" este etichetat cu text în format liber în loc de a se realiza alocarea comportamentului prin stabilirea numelui unei operații.

Exemplul 3 Cazul de utilizare Livrare ordin. O diagrama de secvențe pentru cazul de utilizare *Livrare ordin* este în Figura 20. Varianta îmbunătățită este în Figura 21.

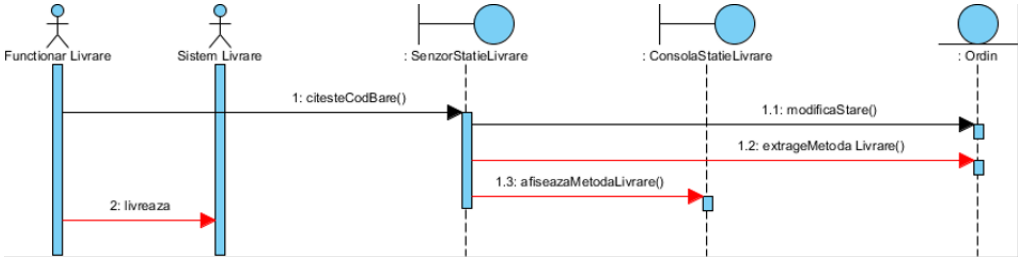


Figure 20: Diagrama de secvențe *Livrare ordin*

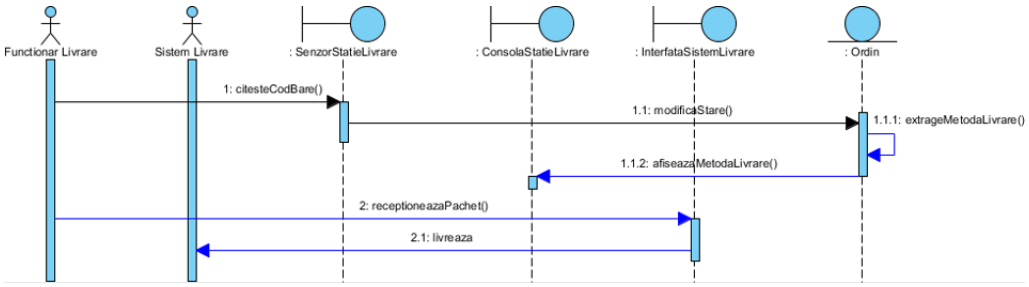


Figure 21: Diagrama de secvențe *Livrare ordin* (corectată)

Explicație. *În Figura 20,* `InterfataSistemLivrare` *lipsește; faptul că* `FunctionarLivrare` *comunică direct cu* `SistemLivrare` *înseamnă că diagrama nu arată cum este înregistrată livrarea. Obiectul* `SenzorStatieLivrare` *avea controlul, chiar dacă acest lucru nu este logic în situații de față.*

Exemplul 4 Cazul de utilizare Editare conținut coș de cumpărături. O diagrama de secvențe pentru cazul de utilizare *Căutare după autor* este în Figura 22. Varianta îmbunătățită este în Figura 23.

Explicație. *În Figura 22, al doilea apel al operației* `getArticol` *introduce confuzie în diagramă. Obiectul* `Articol` *trimitea un mesaj stergeArticol la obiectul* `CosCumparaturi`*. Mesajele "destroy" sunt necesare dacă nu știm deocamdată că implementarea va fi făcută într-un limbaj ce asigură automat garbage collection.*

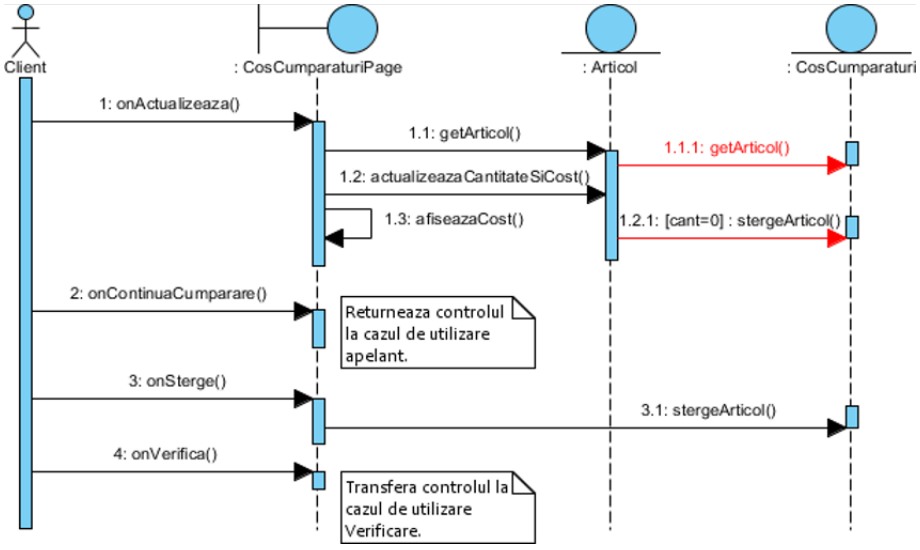


Figure 22: Diagrama de secvențe Editare conținut coș de cumpărături

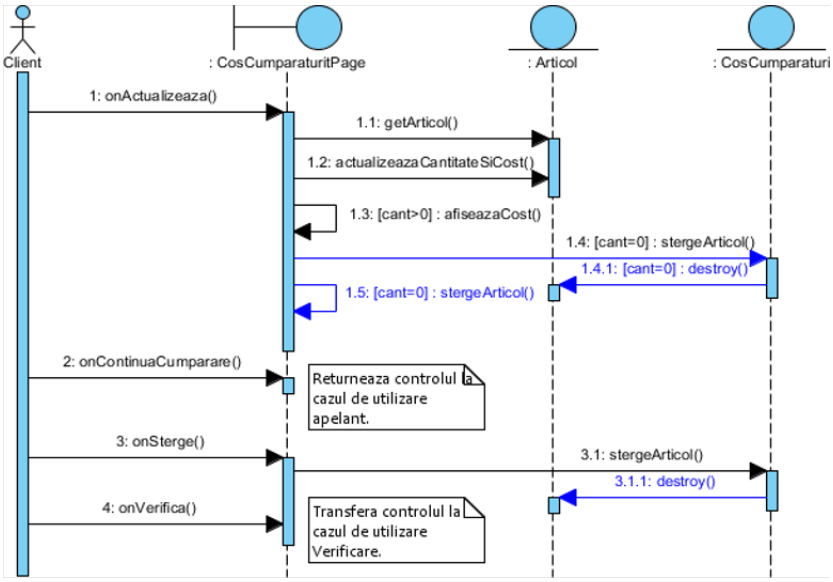


Figure 23: Diagrama de secvențe Editare conținut coș de cumpărături (corectată)

Exemplul 5 Cazul de utilizare Urmărire ordine recente. O diagrama de secvențe pentru cazul de utilizare *Urmărire ordine recente* este în Figura 24. Varianta îmbunătățită este în Figura 25.

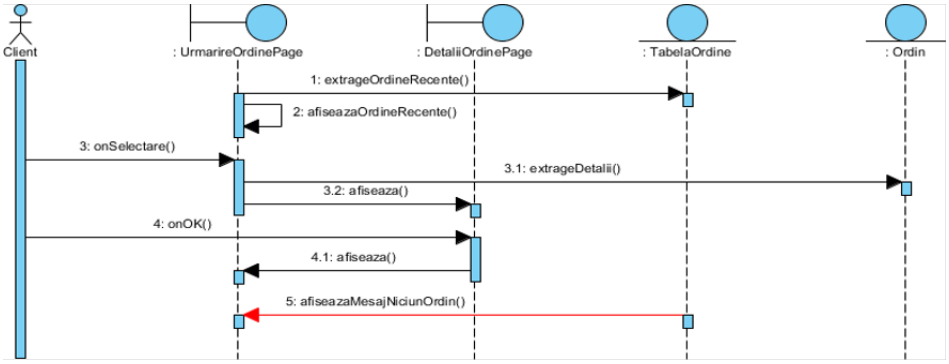


Figure 24: Diagrama de secvențe Urmărire ordine recente

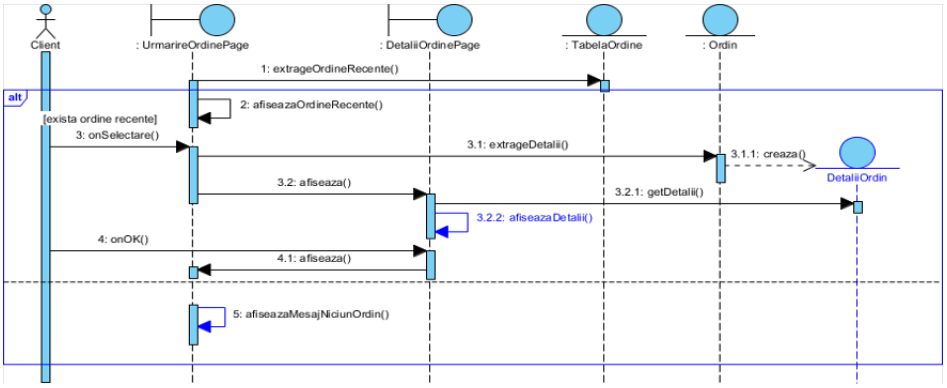


Figure 25: Diagrama de secvențe Urmărire ordine recente (corectată)

Explicație. *În Figura 24, nu există obiectul* `DetaliiOrdin`*, care a fost identificat în cursul analizei de robustețe. Obiectul* `TabelaOrdine` *invocă operația* `afiseazaMesajNiciunOrdin` *pe* `UrmarireOrdinePage`*. Este o idee proastă ca tabele din baza de date sau obiectele asociate acestora să invoce operații pe interfața cu utilizatorul.*

6 Completarea diagramei de clase

Figura din fișierul Extras_DClase_r.bmp conține un extras dintr-o diagramă de clase la nivelul proiectării (model static) pentru aplicația Librărie pe Internet.

Sunt evidențiate elementele de proiectare adăugate versiunii anterioare a diagramei, anume clase boundary (de interfață cu utilizatorul) și operații ale claselor existente și ale noilor clase.

7 Temă

1. Studiați la diagrama de secvențe de la `http://www.visual-paradigm.com/product/vpuml/provides/umlmodeling.jsp`.
2. Studiați informațiile de la: `http://www.agilemodeling.com/artifacts/sequenceDiagram.htm` și `http://www.agilemodeling.com/style/sequenceDiagram.htm`
3. Realizați diagramele de secvențe pentru fiecare caz de utilizare al aplicației *sistem rezervare mașini.*