Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator Nr.2

Disciplina: Analiza și Modelarea Orientată pe Obiect

Tema: Realizarea diagramelor use-case

A efectuat: Zavorot Daniel, st.gr.TI-194

A verificat: asistent universitar Rusu Cristian

Chișinău 2021

**Scopul:** Studierea noțiunilor de actor, caz de utilizare, nota, pachet și relațiile între entități UML

**Sarcina:** De realizat 4 diagrame use case pentru sistemul informațional ales

**Considerații teoretice**

O ***diagramă use case*** este una din diagramele folosite în UML pentru a modela aspectele dinamice ale unui program alături de diagrama de activităţi, diagrama de stări, diagrama de secvenţă şi diagrama de colaborare.

Elementele componente ale unei diagrame use case sunt:

- *use case*-uri;

- *actori;*

- *relaţiile* care se stabilesc între use case-uri, între actori şi între use case-uri şi actori.

Un *use case* (caz de utilizare) reprezintă cerinţe ale utilizatorilor. Este o descriere a unei mulţimi de secvenţe de acţiuni (incluzând variante) pe care un program le execută atunci când interacţionează cu entităţile din afara lui (*actori*) şi care conduc la obţinerea unui rezultat observabil şi de folos actorului. Un use case descrie ce face un program sau subprogram, dar nu precizează nimic despre cum este realizată (implementată) o anumită funcţionalitate.

Fiecare use case are un *nume* prin care se deosebeşte de celelalte use case-uri. Acesta poate fi un şir arbitrar de caractere, însă de regulă numele sunt scurte fraze verbale care denumesc un comportament ce există în vocabularul sistemului ce trebuie modelat.

Figura 1 prezintă notaţia grafică pentru use case.

Figura 1. Notația grafică pentru use case

Un *actor* reprezintă idealizarea unei peroane, proces sau obiect exterior care interacţionează cu un sistem, subsistem sau o clasă. Actorii sunt entităţi exterioare sistemului. Ei pot fi utilizatori (persoane), echipamente hardware sau alte programe. Fiecare actor are un *nume* care indică rolul pe care acesta îl joacă în interacţiunea cu programul.

Notaţie grafică pentru un actor este ilustrată în figura 2.

Figura 2. Notația grafică pentru actor

***Relaţiile*** exprimă interacţiuni între use case-uri, între actori şi între use case-uri şi actori. Relaţiile pot fi de mai multe tipuri: asociere, dependenţă şi generalizare.

***Relaţia de asociere*** se defineşte între actori şi use case-uri, sau între use case-uri. Este folosită pentru a exprima interacţiunea (comunicarea) între elementele pe care le uneşte. Relaţia de asociere se reprezintă grafic printr-o linie şi este evidenţiată în exemplele din figurile 3 şi 4.



Figura 3. Exemplu de asociere între actor și use case

***Relaţia de dependenţă*** se poate stabili numai între use case-uri. Acest tip de relaţie modelează două situaţii:

* cazul în care un use case foloseşte funcţionalitatea oferită de un alt use case – dependenţa de tip *include*;
* există variante ale aceluiaşi use case – dependenţa de tip *extend*.

Figura 4. Dependență de tip *include*

În acest caz comportamentul use case-ului B este inclus în use case-ul A. B este de sine stătător, însă este necesar pentru a asigura funcţionalitatea use case-ului de bază A.

Dependenţa de tip *include* se foloseşte şi pentru a scoate în evidenţă un comportament comun (B poate fi inclus în mai multe use case-uri de bază – vezi figura 5).

Figura 5. Dependență de tip *include* în care un use case este

inclus în mai multe use case-uri de bază

Dependenta de tip ***extend.***

******

Figura 6. Dependență de tip extend

În acest caz comportamentul use case-ului B poate fi înglobat în use case-ul A. A şi B sunt de sine stătătoare. A controlează dacă B va fi executat sau nu.

***Relaţia de generalizare*** se stabileşte între elemente de acelaşi tip (doi actori, sau doua use case-uri). Este similară relaţiei de generalizare (moştenire) care se stabileşte între clase. Figura 7 ilustrează notaţia grafică pentru relaţia de generalizare între use case-uri. Elementul derivat B moşteneşte comportamentul şi relaţiile elementului de bază A. Use case-ul B este o specializare a use case-ului A.

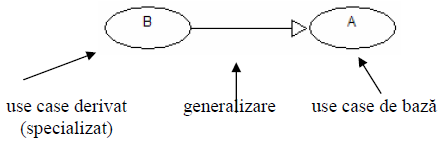
În cazul unei relaţii de generalizare între use case-uri comportamentul poate fi modificat sau extins; use case-ul derivat B poate înlocui în anumite situaţii use case-ul de bază A. Case-ul derivat B controlează ce anume se execută şi ce se modifică din use case-ul de bază A.

Figura 7. Notația grafică pentru relația de generalizare între use case-uri

După cum am precizat mai sus, relaţia de generalizare se poate aplica şi între actori. În exemplul din figura 8 este prezentată o relaţie de generalizare între actori.

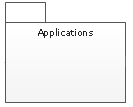
Figura 8. Relație de generalizare între actori

 ***Notațiile*** sunt utilizate pentru a furniza informațiile necesare ale unui sistem.

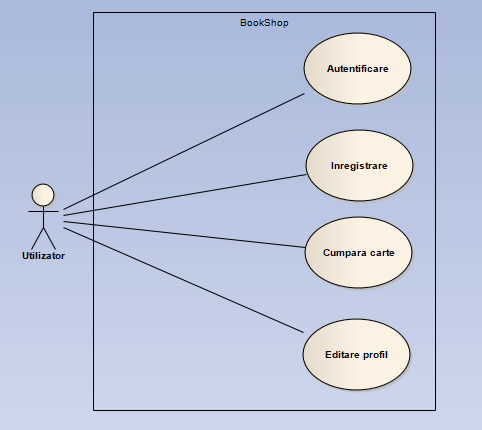
Figura 9. Notațiile în limbajul UML

***Pachetul*** reprezintă un mecanism universal de organizare a elementelor in grup.

Diagrama de pachete este utilizată pentru a simplifica diagrame de clase complexe, oferă posibilitatea de a grupa clase în pachete.

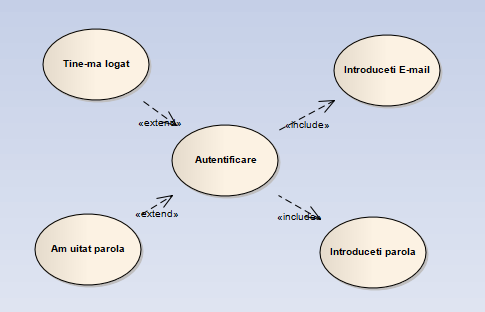
 Figura 10. Pachetul în limbajul UML

**Implementare, rezultate practice:**



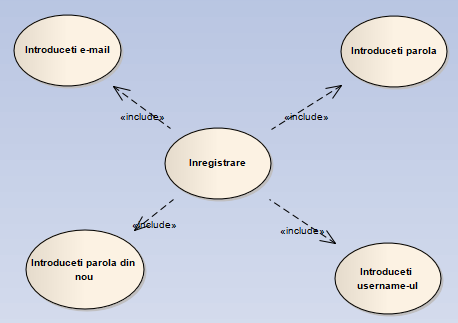
***Figura 1.*** Diagrama sistemului informational ,,BOOKShop”

In acest sistem noi am reprezentat diagramele use-case si relatiile intre ele.Aceasta reprezentare ne ajuta sa intelegem si sa modelam relatiile intre niste actorii in cazul sistemului nostru fiind utilizatorii si adminul. La fel noi modelam actiunile ce se pot intimpla in cadrul unor evenimente din sistem. De asemenea in aceasta diagrama avem relatii de asociere.



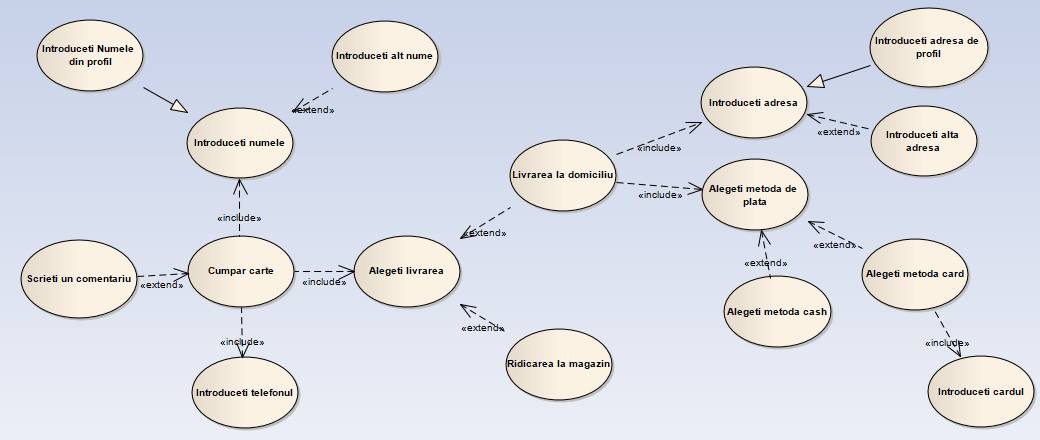
***Figura 2.*** Diagrama de „Autentificare”

In fig 2 sunt reprezentate relatiile „extend” si „include”.Aceste relatiile ne ofera posibilitatea sa aratam ce se intimpla cind se foloseste un use-case.De exemplu in fig 3 daca utilizatorul doreste sa intre la el pe cont el la inceput trebuie sa introduca datele contului sau si obligatoriu deja la nivelul websiteului se verifica datele.Acesta este „include”cind un lucru se va intimpla numaidecit dupa indeplinirea unei actiunii.Totusi daca datele sunt gresite la fel trebuie de anuntat despre acest lucru pe utilizator.



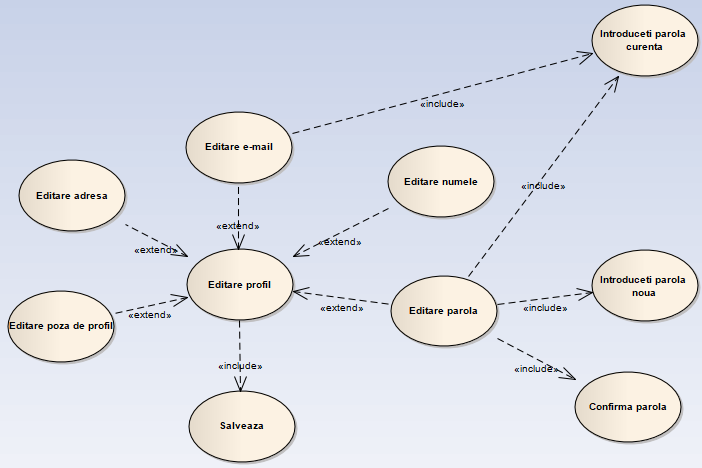
***Figura 3.*** Diagrama relatiilor „extend” si „include” pe baza use-caselui „Inregistrare”

In fig 3 sunt de asemenea reprezentate relatiile „extend” si „include”.Aceste relatiile ne ofera posibilitatea sa aratam ce se intimpla cind se foloseste use-case „Inregistare”, obligatoriu trebuie sa introducem parola, parola din nou, username-ul si email-ul.



***Figura 4.*** *Diagrama use-caseului „Cumpar carte”*

In fig 4 deja se utilizeaza relatiile ”extend”, „include” si „generalization”.Use-caseul de baza fiind „Cauta carte”. „Introduceti numele” si „Introduceti adresa” care deja au ca copil pe „Introduceti numele din profil” si „Introduceti adresa de profil”.



**Figura 5.** Diagrama relatiilor „extend” si „include” pe baza use-caselui *„Editare profil”*

In fig 5 sunt de asemenea reprezentate relatiile „extend” si „include”.Aceste relatiile ne ofera posibilitatea sa aratam ce se intimpla cind se foloseste use-case-ul „Editare profil”. Obligatoriu o sa trebuiasca sa introducem parola curenta daca modificam email-ul sau creem o parola noua, in alte cazuri este de ajuns doar sa apasam pe „Salveaza”.

**Concluzii:**

În această lucrare de laborator s-a analizat puternic și s-a însișit precesul de modelare a diagramelor Use Case. S-au proiectat 4 diagrame use case pentru sistemul informațional Book Shop, folosind aplicația Enterprise Architect. Pentru elaborarea acestui sistem s-a utilizat actori, caz de utilizare și relații care reprezintă interacțiunea dintre actor - caz de utilizare sau actor – actor. Pentru reprezentarea interacțiunii dintre actor-caz de utilizare s-a utilizat relațiile de asociere și dependență, iar penru actor-actor s-a utilizat relația de generalizare. Apoi pentru fiecare use case în parte s-a elaborat relația de dependență cu alte use case-uri care împreună formează o funcționalitate a sistemului. În final s-a obținut un sistem Book Shop.

Bibliografie

1. Melnic R., Sava N. Indrumar metodic “Analiza si modelarea sistemelor informationale”.
2. Моделирование бизнес процессов|CASE средства|Rational Rose, [Электронный ресурс].-Режим доступа: <http://www.kpms.ru/Automatization/Rational_Rose.htm>