CAPITOLUL III

Requirements Engineering

2 tipuri de use cases – sea level: sunt cele pe bune, adevărate, cele care reprezintă o interacțiune majoră a actorilor cu sistemul

– fish level: toate includerile, extinderile, scenariile

mai apare la sistemele foarte mari și fiate level – cazuri absolut excepționale
– face o grupare de use cases sea level

Întrebări

① Tehnica de analiză a cerințelor folosind Use Cases este specifică procesului de dezvoltare "Waterfall" pentru că aici cerințele trebuie analizate riguros la începutul proiectului.

?

② Doi sau mai mulți actori nu pot fi asociați (adică nu pot interacționa) cu același use-case pentru că aceasta ar însemna că sunt redundanți.

Afirmația este fundamental FALSĂ, deoarece un use case trebuie să aibă minimum un actor și poate să aibă maxim cât e nevoie.

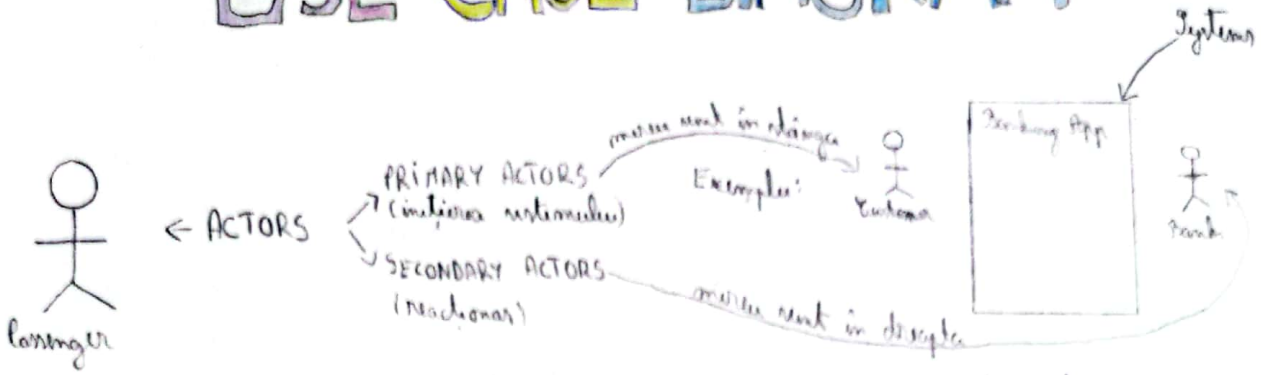
③ Use Case-urile de tip "Fish Level" se folosesc atunci când vrem să detaliem fiecare acțiune principală din cadrul unui use-case de tip „Sea Level".

Afirmația este fundamental ADEVĂRATĂ. Fish Level reprezintă toate use cases „principale" iar toate includerile și extinderile reprezintă „Sea Level", deci practic sunt detalii de implementare

④ În diagramele UML de Use Case relațiile «extend» și «include» sunt foarte asemănătoare și pot fi folosite interschimbabil întrucât ambele sunt folosite pt a da factor comun descrierea unei anumite funcționalități.

Afirmația este fundamental FALSĂ. Acestea nu pot fi interschimbabile, include are ca scop ori scoaterea factor comun, ori arătarea că un use case se va realiza automat în schimb extend e o relație între use caseuri și scenarii.

USE CASE DIAGRAM

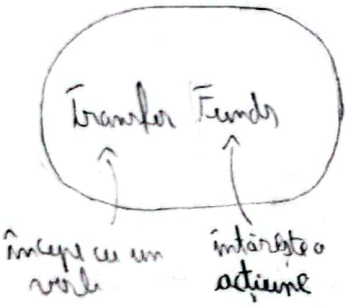


• Un actor (persoană, sistem, organizație, sursă externă) e mereu cineva ce este **extern** față de sistem, el nu face niciodată parte din sistem, el doar utilizează sau este utilizat de către sistem

Exemple:

1 Un călător ce cumpără bilete să meargă cu trenul e un actor

2 Un satelit prin care avem un semnal GPS poate fi văzut ca un actor (deoarece e complet decuplat de sistemul GPS în sine)



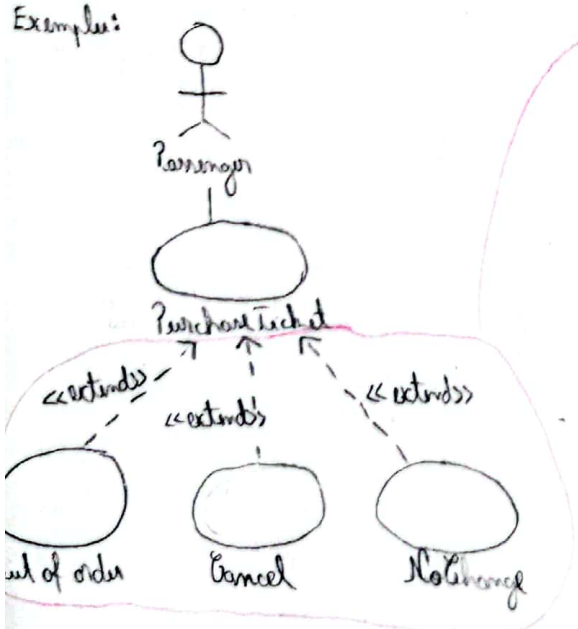
← USE CASES (dacă nu avem actor nu avem use cases)

→ instrumentul conceptual prin care arăt ce face sistemul

→ un mod de a interacționa cu sistemul

The «extends» Relationship

Exemplu:

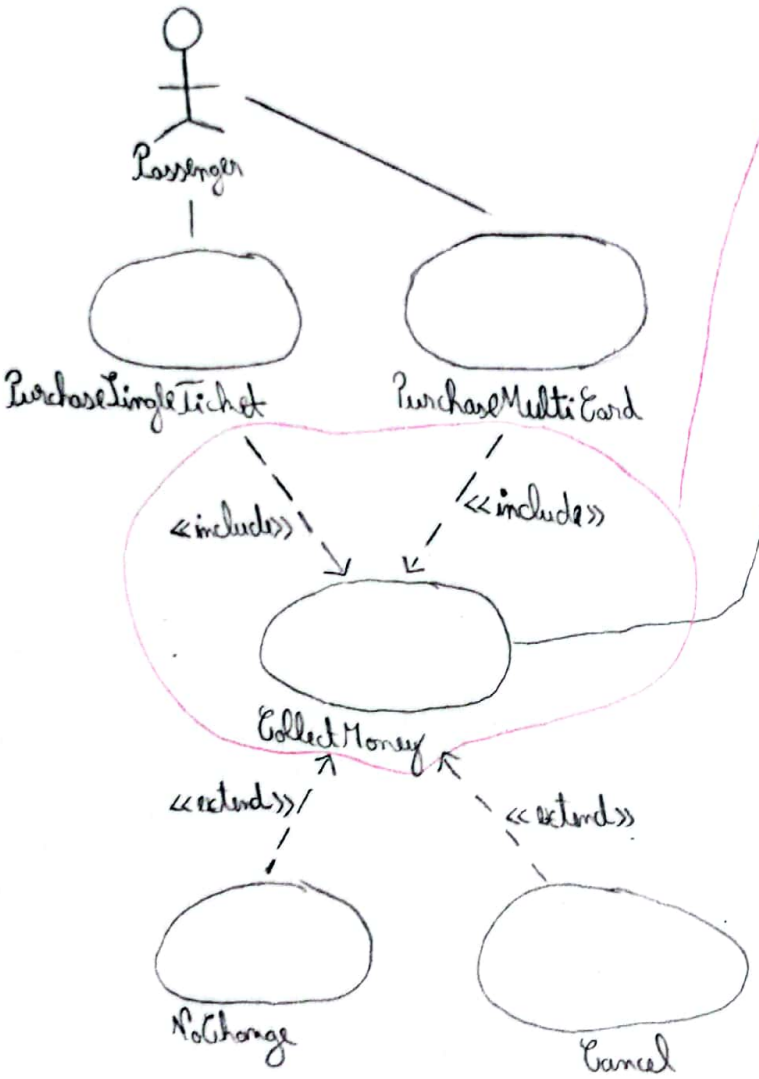


→ reprezintă acțiuni excepționale ale unui caz-de-utilizare

extend = e o relație între un cazuri și subcazuri

• Cost of order, Cancel și No Change se pot executa, dar nu este obligatoriu să se execute.

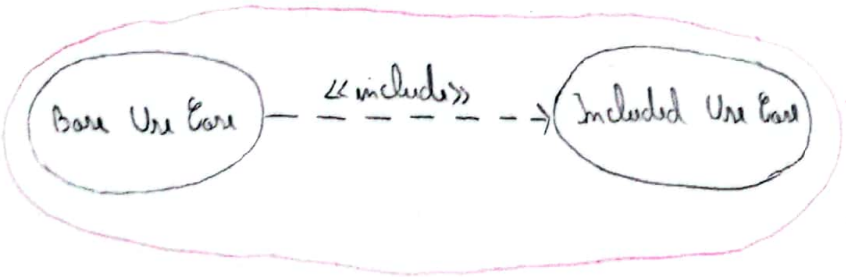
The «include» Relationship



→ reprezintă partea comună a două cazuri de utilizare PurchaseSingleTicket și PurchaseMultiCard

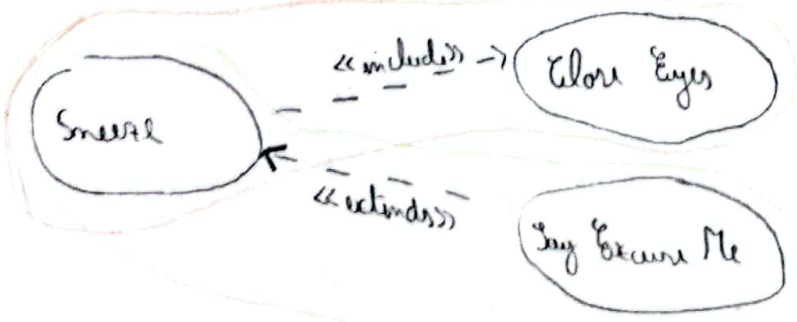
→ strict conceptual CollectMoney nu e un caz de utilizare deoarece nu produce nimic util pentru un actor (face ceva util dar o face indirect)

❗ Un caz de utilizare se numește doar ceea ce produce ceva util pentru un actor în cazul nostru nu: PurchaseSingleTicket și PurchaseMultiCard



→ de fiecare dată când Base Use Case se va executa și Included Use Case se va executa de asemenea

EXEMPLU CU DIFERENȚA DINTRE «include» și «extends»

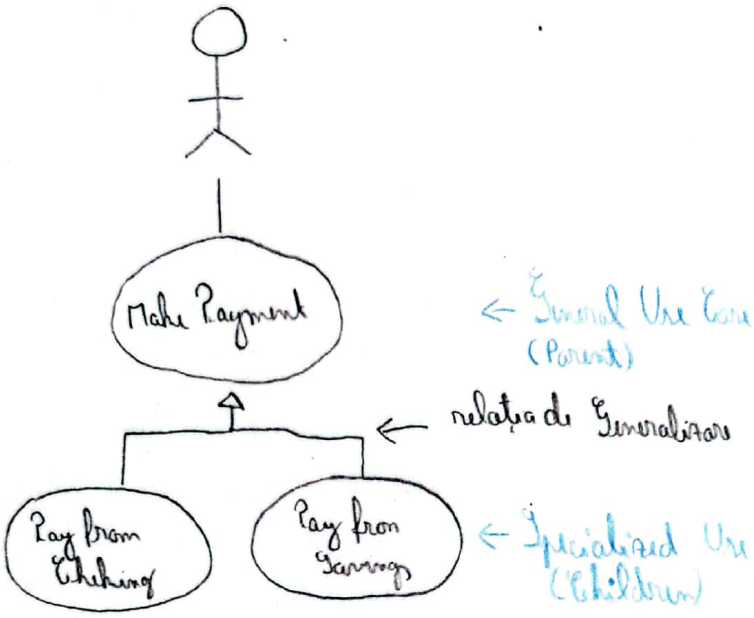


• Dacă strănuți automat vei închide ochii (se întâmplă de fiecare dată)

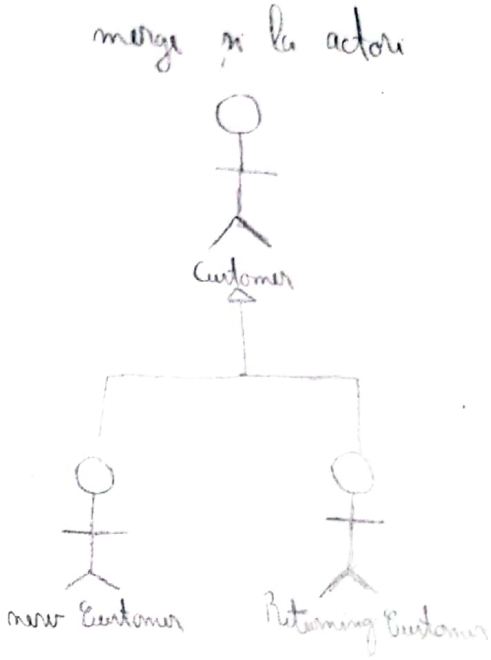
• Nu este obligatoriu să îți ceri scuze după ce ai strănutat (se poate întâmpla

The Generalization Relationship

- este cunoscută și sub relația de moștenire



merge și la actori



Use case with extension points