**TEMA CURS INGINERIE SOFTWARE**

**UML Diagrame de Interacțiune - Secvență și Colaborare**

**Student: ANDREI Emanuel Vlad**

**Grupa: 442A**

Cuprins

[1.Introducere](#h.h5tfbwjayhqo)

[1.1 Ce este UML?](#h.n9zh33nswuw3)

[1.2 Descrierea termenului “Unified” (Unificat)](#h.4i81t1h79mcp)

[1.3 Definiția unei clase](#h.ceflpg20rqzt)

[Figura 1: Notația unei clase](#h.lv6gw6a2cwy9)

[2. Diagrama de Secventa](#h.7kzh31unqvb8)

[2.1 Specificații de Execuție](#h.o6kfdo2qjhux)

[3 Diagrama de colaborare](#h.6jmucxc2jzij)

[4. Conluzii](#h.ptdnovainsgu)

# 

# 1.Introducere

### 1.1 Ce este UML?

UML (Unified Modeling Language) este un limbaj standard pentru specificarea, vizualizarea, construirea și documentarea tuturor abstractizărilor unui sistem software.

UML reprezintă notația standard a industriei grafice ce descrie analiza și dizainul software. Totuși acestă notație este foarte ambiguă, astfel lăsând loc interpretărilor. Prin urmare standardizarea prevede comunicarea eficientă și conduce la mai puține erori cauzate de neînțelegere.

UML poate fi folosit într-o varietate de modalități pentru a sprijinii o metodologie de dezvolatre software, dar în sine acest limbaj nu specifică metodologia sau procesul în sine.

Structura acestor diagrame definește arhitectura statică a unui model. Ele sunt folosite pentru a modela ”lucruri” care alcătuiesc un model: clase, obiecte, interfețe și comportamente fizice. De altfel aceste diagrame sunt folosite pentru a modela relații și dependențe între elemente.

UML capteză informații despre structura statică și dinamică a unui sistem. Un sistem este modela ca o colecție de obiecte discrete care interacționează pentru a efectua munca de care beneficiază în cele din urmă un utilizator. Structura statică definește tipurile de obiecte importante pentru un sistem și punerea lor în aplicare, precum și relațiile dintre obiecte.

Comportamentul dinamic definește istoria obiectelo în timp și comunicările între obiecte pentru a realiza obiectivele.

UML conține de asemenea constructori organizatorici pentru aranjarea modelelor în pachete care permit programatorului să partiționeze un sistem larg în bucăți fezabile, de a înțelege și controla dependețele dintre pachete și să gestioneze unitățile de model într-un mediu de dezvoltare complex.

### 

### 

### 

### 1.2 Descrierea termenului “Unified” (Unificat)

### 

* **Metode și Notații:** UML combină conceptele comune acceptate de majoritatea metodelor oobiect-orientate, selectând o definiție clară pentru fiecare concept ca și cum ar fi o notație sau o terminologie. UML poate reprezenta multe modele deja existente mult mai bine decât metodele originale.
* **Ciclu de dezvoltare:** Același set de concept și notații pot fi utilizate în diferite stadii de dezvoltare și chiar amestecate într-un singur model. Nu este necesar de a traduce de la o etepă la alta.
* **Domenii de aplicare:** Modelul UML este destinat multor domenii de aplicare, inclusiv sistemelor care implică date și calcule real-time, complexe, distribuite. Există limbaje cu destinații speciale care pot fi mai utile decât acesta, dar UML este considerat cel mai util limbaj de modelare pentru cele mia multe domenii de aplicare.
* **Limbaj de implementare și platfome:** UML este destinat să fie folosit pentru sistemel implementate în diferite limbaje de implementare și platforme, inclusiv limbaje de programare , baze de date, 4GLs, organizare documente, firmware. Interfețele trebuie să fie similare în toate cazurile în timp ce sorsele diferă pentru fiecare mediu.
* **Procesele de dezvoltare:** UML este un limbaj de modelare, nu o descriere a unui proces de dezvoltare detaliat. Acesta este destinat a fi utilizat ca limbaj de modelare care stă la baza mai multor procese de dezvoltare existente sau noi, la fel cum un limbaj de programare poate fi folosit în diferite stiluri. Acesta este destinat în special pentru a sprijini stilul iterativ, incremental de dezvoltare, pe care îl recomandă.
* **Concepte interne:** În construirea metamodelului UML s-au descoperit și reprezentat relațiile care stau la baza diferitelor concepte, încercând să captureze concepte de modelare într-un mod larg aplicabil multor situații cunoscute și necunoscute. Acest proces a condus la o mai bună înțelegere a conceptelor și aplicabilității generale a acestora. Acest lucru nu a fost scopul inițial al unicității, dar a fost unul dintre cele mai importante rezultate.

# 

### 

### 1.3 Definiția unei clase

Clasa unei diagrama descrie tipurile de obiecte într-un sistem și a diferitelor tipuri de relații statice care există între ele. În UML o clasă este reprezentată de un dreptunghi cu una sau mai multe compartimente orizontale. Partea superioară deține numele clasei. Numele clase este singurul lucru necesar într-o diagramă de clasă. Compartimentul din centru deține lista clase: atribute, membri, iar compartimentul inferior deține: operații, metode.

# 

# Notația unei clase

# 

# 

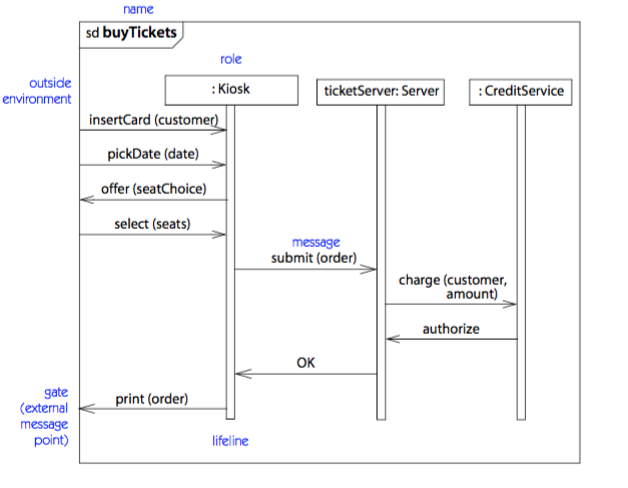
# 2. Diagrama de Secventa

O diagramă de secvență afișează o interacțiune ca o diagramă bidimensională. Dimensiunea verticală este axa timpului; timpul continuă în josul paginii. Dimensiunea orizontală arată rolurile care reprezintă obiecte individuale în colaborare. Fiecare rol este reprezentat de o coloană verticală care conține un header și o linie verticală ”lifeline”.

Obiectul ce există în timp este reprezentat printr-o linie punctată. De-a lungul timpului specificațiile de execuție a unei proceduri asupra obiectului sunt desenate cu linie dublă.

În general o diagramă de secvență arată doar secvențe de mesaje ci nu intervale exacte de timp. O diagramă de distribuție poate fi utilizată atunci când timpul metric este important, dar pentru a înțelege semantica interacțiunilor diagrama de secvență este suficentă.

Un mesaj este afișat ca o săgeată de la ”linia vieții” a unui obiect la cea a altui obiect. Săgețile sunt aranajte în secvență cronologică în josul diagramei. Un mesaj asincron este reprezentat cu un vârf de săgeată.



Schema unei diagrame secvențiale cu mesaje asincrone.

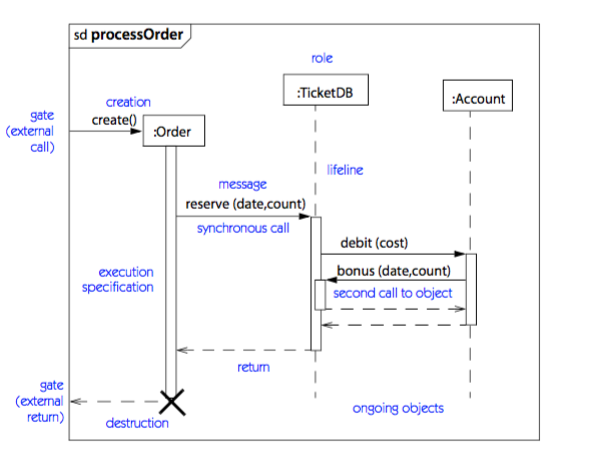
### 2.1 Specificații de Execuție

O specificație de execuție (de activare), este executarea unei proceduri, incluzând orice moment de timp necesar procedurilor să se execute. Este reprezentat de linia dubla care înlocuiește părți ale ”liniei vieții” din diagrama de secvență.

Un apel este indicat de o săgeată care duce către inițializarea apelului. Un apel asincron este reprezentat cu un vârf de săgeată triunghiular umplut. Un apel recursiv apare atunci cand se operează asupra unui obiect, dar cel de al doilea apel execută specificații diferite față de primul. Recursivitatea este reprezentată intr-o diagramă de secvență prin stivuirea liniilor de activare.

Revenirea controlului de la un apel este reprezentat printr-o săgeată punctată. Într-un mediu de procedură aceste săgeți pot fi omise deoarece acestea sunt implict la sfârșitul unei specificații de execuție, dar este mult mai clar dacă acestea sunt reprezentate.

Un obiect activ este cel care deține rădăcina, o stivă de execuții. Fiecare obiect activ are propriul fir de control care se execută în paralel cu alte obiecte active. Un obiect activ este reprezentat printr-o linie dublă pe fiecare parte a simbolului său. Obiectele care sunt numite de către un obict activ sunt obiecte pasive; primesc control doar atunci când sunt solicitate și dau randament atunci când se întorc.



Prezintă o schemă de secvență cu flux procedural de control.

# 3 Diagrama de colaborare

O colaborare este o descriere a unei colecții de obiecte care interacționează pentru a implementa un comportament într-un context. Acesta descrie o societate de obiecte asamblate ce cooperează pentru a efectua scopul propus. O colaborare conține roluri care sunt ocupate de obiecte în timpul execuției. Un rol reprezintă o descriere a obiectelor care participă într-o execuție de colaborare. Un connector reprezintă o descriere a asocierii între rolurile colaborării. Relațiile dintre roluri și conectori în interiorul unei colaborări sunt semnificative doar în acest context.

Rolurile și conectorii pot avea tipuri (clasificatori și asocieri) care specifică ce obiecte pot fi atașate. Tipurile de asociere sunt opționale, deoarece relațiile dintr-o colaborare pot fi tranzitorii și implementare se face folosind alte mecanisme pe post de paramterii. Relațiile descrise de o colaborare se aplică doar la obiectele legate de roluri care au un caz particular de colaborare. Acestea nu se aplică clasificatorilor de bază și asocierilor din afara colaborării.

Vederea statică descrie proprietățile inerente ale unei clase. De exemplu, un bilet are o prezentare și un set asociat cu aceasta. Această relație este valabilă pentru toate instanțele din clasă. O colaborare descrie toate proprietățile instanțelor clasei ce au un rol particular în colaborare. De exemplu, un bilet, într-o colaborare de vânzare are un vânzător, ceea ce nu este relevant față de tiket în general, dar este esențial în colaborarea de vânzare.

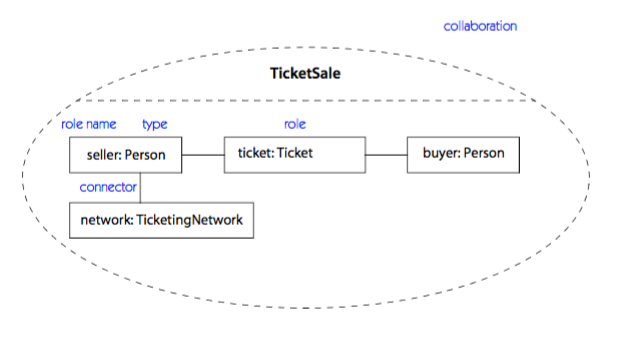
Un obiect într-un sistem poate participa la mai mult de o colaborare. Colaborările în care apare nu trebuie să fie legate în mod direct, cu toate acestea executarea lor este conectată printr-un obiect comun. De exemplu, o persoană poate fi un cumpărător într-o colaborare și un vânzător într-o altă colaborare. De foarte puține ori un obiect poate juca același rol într-o colaborare.

Figura 3: Colaborarea pentru vânzarea unui bilet.

# 4. Conluzii

Este clar că diagramele de secvență au o serie de avantaje foarte puternice. Ele descriu in mod clar succesiunea de evenimente, arată atunci când obiectele sunt create și distruse, ilustrează operații cocurente. Cu toate aceste avantaje ele totuși nu sunt niște tool-uri perefecte. Acestea ocupă mult spațiu și nu sunt prezente în relațiile de colaborare între obiecte. Cu toate acestea diagramele de secvență nu oferă de densitatea și eleganța diagramelor de colaborare.

Prin urmare consider că diagramele de secvență sunt recomandabile atunci când sunt necesare competențele specifice acestora.