Restaurant System

Requirements: A restaurant has many waiters and one cook. For each, the application opens a window. The waiters take orders from the clients and introduce them using their windows. An order contains the table number, the dishes and the quantity for each type of dish. After an order was introduced, it is sent to the cook, where the list of orders is updated. The cook prepares the food, and after they are ready, he marks the corresponding order as finished. Automatically, this order appears as a ready order in the waiter's window (and may be delivered), and is deleted from the list of the cook. Statistics are required, such as finding out how many times a dish type has been ordered. CRUD functionality is also required, at least for dish types.

Seminar 1 **ISS 2019-2020**

Modelul functional al unui sistem.
Diagrama UML a cazurilor de utilizare (eng. UML Use Case Diagram - UCD)

Biblio:
1. Martin Fowler , *UML Distilled* (3 ed.)
2. OMG, *UML 2.5.1 Specification*
3. Martina Seidl et al., *UML@Classroom*

Modelul functional
- produs al activitatii *Colectarea cerintelor*, alături de descrierea cerintelor nefunctionale ale sistemului
- descris în termeni de *actori* și *cazuri de utilizare*
- instrumente de reprezentare: *diagrama UML a cazurilor de utilizare* + *sabloane tabelare* pentru descrierea narativa a continutului cazurilor de utilizare

1) Caz de utilizare (CU)

Def.1: Scenariu = o secvența de pași/evenimente, care descrie o interacțiune tipica a unui „utilizator" cu sistemul.

Ex.1: scenariu/flux normal privind plasarea unei comenzi într-un magazin virtual (eng. MSS – main success scenario)

1. Clientul populeaza cosul de cumpărături, selectand produse din catalogul online.
2. Sistemul actualizeaza continutul cosului.
3. Clientul opteaza pentru vizualizarea continutului cosului.
4. Sistemul afiseaza detaliile cosului.
5. Clientul opteaza pentru finalizarea comenzii.
6. Sistemul solicita, prin afisarea unui formular, introducerea detaliilor de livrare (destinatar, adresa, tip livrare – poșta/curierat/pick-up point).
7. Clientul completeaza detaliile de livrare și submite informatiile.
8. Sistemul afiseaza prețul final al comezii (actualizat cu costurile de livrare) și solicita alegerea unei modalitati de plata (ramburs/transfer bancar/onlne, cu card).
9. Clientul opteaza pentru plata cu cardul.
10. Sistemul solicita detaliile cardului (nume deținător, nr. card, data expirării, CVV)
11. Clientul introduce datele cardului și opteaza pentru finalizarea platii.
12. Sistemul valideaza cardul și notifica utilizatorul referitor la plasarea cu succes a comenzii, atât printr-un mesaj în interfata, cât și printr-un email ulterior.

Ex. 2: scenariu alternativ 1 (de excepție): autorizarea cardului esueaza, datele introduse sunt incorecte

12ª. Sistemul afiseaza un mesaj de eroare si solicita reintroducerea datelor de card.
12ª.1 Utilizatorul reintroduce datele (corecte).
Se revine in MSS la punctul 12.

Ex. 3: scenariu alternativ 2: clientul e unul deja înregistrat în sistem, având datele de livrare și de card salvate

6ª. Sistemul solicita utilizarea sau suprascrierea datelor preferentiale salvate, legate de livrare și plata cu cardul.
6ª.1 Clientul opteaza pentru utilizarea acestora.
Se revine în MSS la punctul 12.

... samd

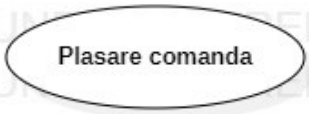
In scenariile alternative pot apărea oricati pași, funcție de natura interactiunii (nu doar unul, așa cum se întâmpla în scenariile descrise anterior). La fel, funcție de natura interactiunii, se poate reveni sau nu la unul din pasii din scenariul principal.

Punctul comun al tuturor scenariilor anterior descrise = un același obiectiv final al „utilizatorului" (plasarea unei comenzi).

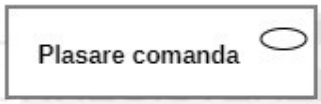
Def.2: Caz de utilizare = mulțimea tuturor scenariilor care au ca și punct comun un același obiectiv final (scop) al „utilizatorului" - realizarea aceleiasi functionalitati.

Caz de utilizare (CU) = o mulțime de secvente de evenimente, care descriu toate interactiunile posbile intre un „utilizator" și sistem, în scopul realizarii unei anumite functionalitati.

Sintaxa UML – elipsa (cu numele cazului sub/in interior) sau reprezentare de clasificator - dreptunghi (cu icon elipsa). Numele unui CU este o expresie verbala!!! (fie ea și substantivizata): *Plaseaza comanda* sau *Plasare comanda*, NU Comanda – ca și substantiv!!! (sa rezulte functionalitatea oferita, o comanda se poate plasa, pune în așteptare, onora etc.)



sau



„utilizator" = actor

2) Actor (A) = *rol* jucat de o entitate *externa* sistemului, care *interactioneaza* cu sistemul.

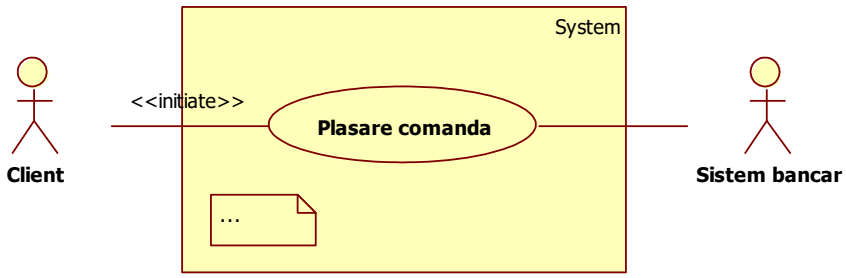
Instantele actorilor pot fi persoane, alte sisteme (sistemul bancar cu care aplicația comunica pentru validarea cardurilor), echipamente hardware/dispozitive (imprimanta, senzori externi).
O instanța poate juca mai multe roluri (un manager poate juca rol de vânzător, la nevoie), un rol poate fi jucat de mai multe instante (orice persoana poate fi client al magazinului virtual).
Un actor poate fi (și este, în general) implicat în mai multe cazuri de utilizare; la un caz de utilizare pot participa mai mulți actori (în CU *Plasare comanda* intervine atât clientul, cât și sistemul bancar la care se apeleaza pentru validarea cardului). Doar unul dintre ei initiaza interacțiunea (stereotipizare cu <<initiate>> a relatiei de comunicare aferente).

Sintaxa UML - eng. *stick figure* (denumirea sub/deasupra) sau reprezentare de clasificator (cu stereotip <<actor>> explicit sau icon) sau imagine (reprezentativa pentru tipul de actor, de ex. imaginea unei imprimante)

sau

3) Frontiera sistemului (eng. *system boundary* sau *subject* - în ultimele versiuni ale specificatiei) – diferentiaza interiorul sistemului de mediul sau: actorii sunt înafara frontierei, cazurile de utilizare înăuntru.

Sintaxa UML – dreptunghi etichetat cu numele sistemului (conform specificatiei UML, eticheta se plaseaza în coltul stânga sus)



4) Relatii

4.1. Comunicare

- intre A și CU
- rel. de asociere bidirectionala – comunicarea/schimbul de informație e în ambele sensuri

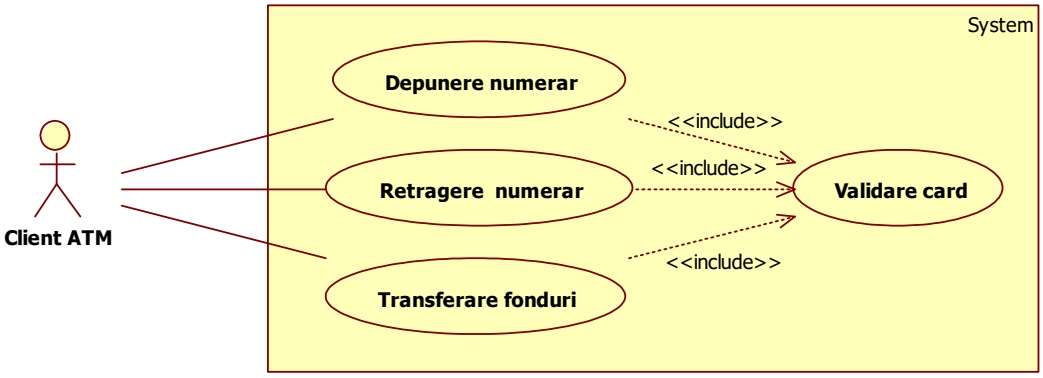
Poate fi stereotipizata cu <<initiate>>, pentru a indica acel actor care initiaza comunicarea, în cazul în care exista mai mulți actori care interactioneaza cu CU.

4.2 Incluziune

- intre doua CU
- rel. de dependenta, stereotipizata cu <<include>>

Sensul dependentei e de la CU care include (de baza) la CU inclus. Dpdv semantic, comportamentul cazului de utilizare inclus va fi inserat în cadrul comportamentul cazului de utilizare care include. Comportamentul cazului de utilizare care include nu este complet definit (nu are sens), decât în prezenta comportamentului cazului de utilizare inclus (nu se poate face retragere de numerar fără validarea cardului).

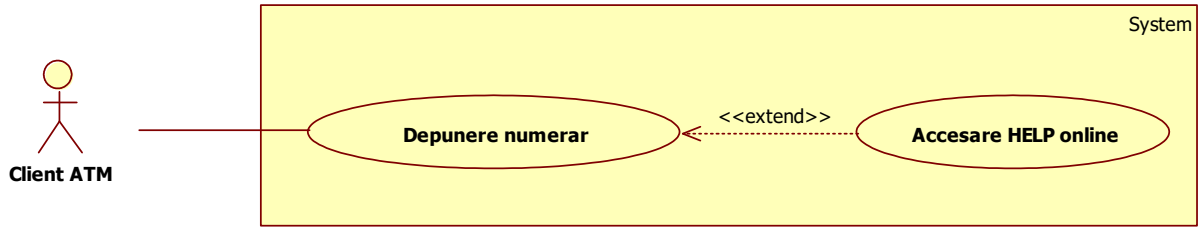
Relația de incluziune se recomanda a fi folosită atunci când doua sau mai multe cazuri de utilizare partajeaza o aceeași secvența de comportament. Secvența comuna poate fi izolată într-un nou caz de utilizare, ce va fi inclus în cele initiale. Scopul este evitarea redundantelor și simplificarea descrierii cazurilor de utilizare (maximizarea inteligibilitatii descrierilor textuale). Similitudine cu apelul de procedura din limbajele de programare. În descrierea textuala a cazurilor de utilizare care includ, secvența comportamentala factorizata (succesiunea de evenimente/pasi) poate fi inlocuita cu un singur pas, care refera numele cazului de utilizare inclus (hiperlink).



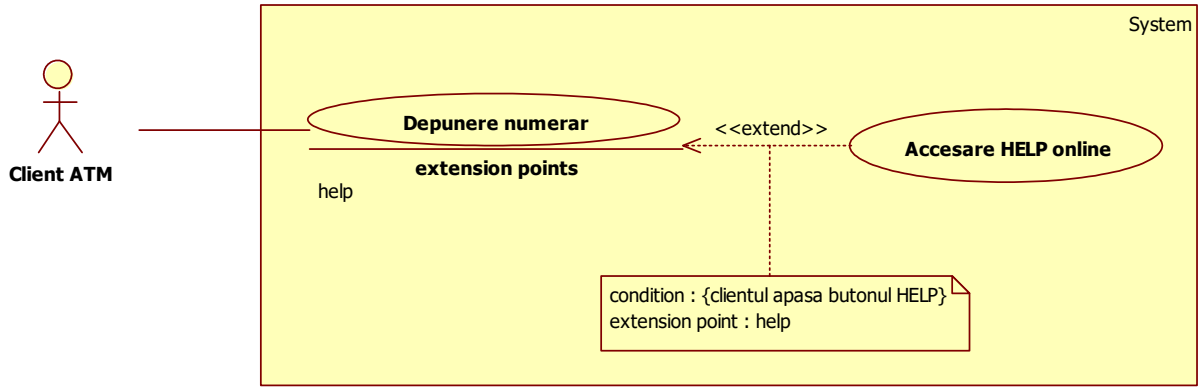
4.3 Extindere

- intre doua CU
- rel. de dependentă, stereotipizata cu <<extend>>

Sensul dependentei e de la CU care extinde la CU extins (de baza). Dpdv semantic, relația precizeaza *când* (în ce condiții) și *unde* în comportamentul cazului de utilizare de baza se poate insera comportamentul descris de cazul de utilizare care extinde. Comportamentul cazului de utilizare extins este însă complet definit în absenta celui care extinde (se poate face depunere fără accesare de help online). În schimb, cazul de utilizare care extinde poate fi doar un increment comportamental, fara aiba semnificatie de sine statatoare (în absenta cazului de care depinde).

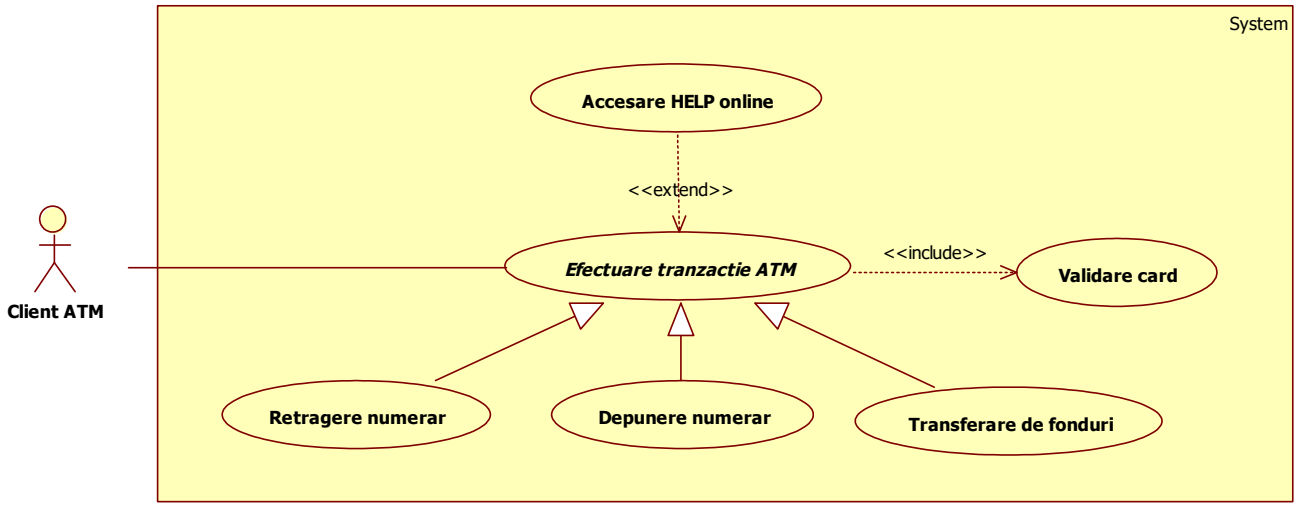


Pentru specificarea completă a relatiei de extindere este necesara precizarea conditiilor în care se face extinderea și localizarea acesteia (folosind o nota UML atașată relatiei de extindere). Localizarea se face cu ajutorul așa-numitor *puncte de extindere* (eng. *extension points*), care se definesc la nivelul cazului de utilizare de baza. Pentru fiecare punct de extindere se precizeaza numele (ales de utilizator) și locația (raportata la modalitatea de descriere a continului cazului de utilizare). În cazul descrierii tabelare/narative a cazului de utilizare, localizarea unui punct de extindere se poate face prin prezizarea numerica a pasului după care se aplica. Un același punct de extindere poate avea mai multe locatii posibile (un același increment se poate aplica în mai multe locuri la nivelul unui caz de utilizare). În cazul în care incrementul descris de cazul de utilizare care extinde se poate aplica oriunde în comportamentul de baza, omitem precizarea locatiei punctului de extindere aferent.



? Gestionarea datelor unui client deja înregistrat (flux alternativ al cazului de utilizare plasare comanda în magazin virtual) poate fi izolată la nivelul unui caz de utilizare care extinde plasarea comenzii? Dar fluxurile de excepție?

4.4 Generalizare
- intre A sau intre CU
- semantica uzuala



Practic: Diagrama cazurilor de utilizare pentru sistemul de preluare a comenzilor din cadrul unui restaurant (vezi modelul din Seminar 2).

5) Descrierea continutului cazurilor de utilizare: Relativ la realizarea modelului functional, UML ofera doar diagrama cazurilor de utilizare, însă nu și o modalitate standard de a descrie continutul acestora (motiv pentru care nu este formalizata nici modalitatea de localizare a punctelor de extensie discutate anterior). De regula, în acest scop se folosesc sabloane tabelare, în care un caz de utilizare este descris textual, în limbaj natural, precizand cel puțin: numele cazului de utilizare, actorii participanti, fluxul normal de evenimente și fluxuri alternative sau excepții. Mai pot apărea pre/post-conditii și cerinte de calitate.

Ex.4: Descrierea cazului de utilizare *Plasare comanda* al sistemului *MagazinVirtual*

Nume	*Plasare comanda*
Actori participanti	Initial de *Client* Comunica cu *Sistem bancar*
Flux de evenimente (scenariu normal)	1. Clientul populeaza cosul de cumpărături, selectand produse din catalogul online. 2. Sistemul actualizeaza continutul cosului. 3. Clientul opteaza pentru vizualizarea continutului cosului. 4. Sistemul afiseaza detaliile cosului. 5. Clientul opteaza pentru finalizarea comenzii. 6. Sistemul solicita, prin afisarea unui formular, introducerea detaliilor de livrare (destinatar, adresa, tip livrare – poșta/curierat/pick-up point). 7. Clientul completeaza detaliile de livrare și submite informatiile. 8. Sistemul afiseaza prețul final al comezii (actualizat cu costurile de livrare) și solicita alegerea unei modalitati de plata (ramburs/transfer bancar/onlne, cu card). 9. Clientul opteaza pentru plata cu cardul. 10. Sistemul solicita detaliile cardului (nume deținător, nr. card, data expirării, CVV) 11. Clientul introduce datele cardului și opteaza pentru finalizarea platii. 12. Sistemul valideaza cardul și notifica utilizatorul referitor la plasarea cu succes a comenzii, atât printr-un mesaj în interfata, cât și printr-un email ulterior.

Scenarii alternative	scenariu alternativ 1 (de excepție): autorizarea cardului esueaza, datele introduse sunt incorecte 12ª. Sistemul afiseaza un mesaj de eroare si solicita reintroducerea datelor de card. 12ª.1 Utilizatorul reintroduce datele (corecte). Se revine in MSS la punctul 12. scenariu alternativ 2: clientul e unul deja înregistrat în sistem, având datele de livrare și de card salvate 6ª. Sistemul solicita utilizarea sau suprascrierea datelor preferentiale salvate, legate de livrare și plata cu cardul. 6ª.1 Clientul opteaza pentru utilizarea acestora. Se revine în MSS la punctul 12.

Seminar 3 **ISS 2019-2020**

Generarea codului pe baza diagramelor UML de clase

Biblio:
1. Dominik Gessenharter, *Mapping the UML2 Semantics of Associations to a Java Code Generation Model*
2. OMG, *UML 2.5.1 Specification*

Reguli privind reprezentarea asocierilor intre clase la nivelul modelului conceptual:

1. Numirea explicita a relatiilor de asociere, folosind expresii verbale sugestive (fac excepție agregarile/compunerile, care se numesc implicit prin „este format/compus din")

2. Precizarea explicita a numelor de roluri (substantive) și a multiplicitatilor aferente ambelor capete ale asocierii

3. Conventii: numele de rol se scriu cu litera mica (ca și atributele); dacă multiplicitatea capatului este de tip *many*, se foloseste pluralul

Reguli privind translatarea elementelor unei diagrame de clase în cod:

1. Claselor din model le corespund clase în limbajul de programare ținta

2. Mulțimea atributelor unei clase din cod se obtine reunind:
+ mulțimea atributelor clasei sursa din model si
+ mulțimea numelor de roluri corespunzatoare capetelor opuse ale tuturor asocierilor navigabile dinspre clasa în cauza spre alte clase
! atenție la conflictele de nume, mai ales în cazul în care o clasa are asocieri diferite cu o aceeași alta clasa, navigabile înspre cea din urma

3. Tipul unui atribut din cod aferent unui atribut din model e reprezentat de:
+ corespndentul, în limbajul ținta ales, al tipului UML al atributului sursa (ex: Integer -> int, UnlimitedNatural -> long, Real -> double, String -> String, Boolean -> bool) – în cazul în care multiplicitatea aferenta la nivelul modelului este *one* (1 sau 0..1)
+ colecție, având ca și tip de baza al elementelor tipul menționat anterior, dacă multiplicitatea aferenta la nivelul modelului este de tip *many* (* sau 0..* sau 1..*). Tipul colectiei depinde de constrangerile de unicitate și ordonare stabilite (ex.: valorile implicite *unique* = true & *ordered* = false => *Set*)

4. Tipul unui atribut din cod aferent unui nume de rol e reprezentat de:
+ tipul clasei de la capătul corespunzator rolului, în cazul în care multiplicitatea setata pentru capătul respectiv e *one* (1 sau 0..1)
+ colecție, având ca și tip de baza al elementelor tipul menționat anterior, dacă multiplicitatea este de tip *many* (* sau 0..* sau 1..*). Tipul colectiei depinde de constrangerile de unicitate și ordonare stabilite (ex.: valorile implicite *unique* = true & *ordered* = false => *Set*)

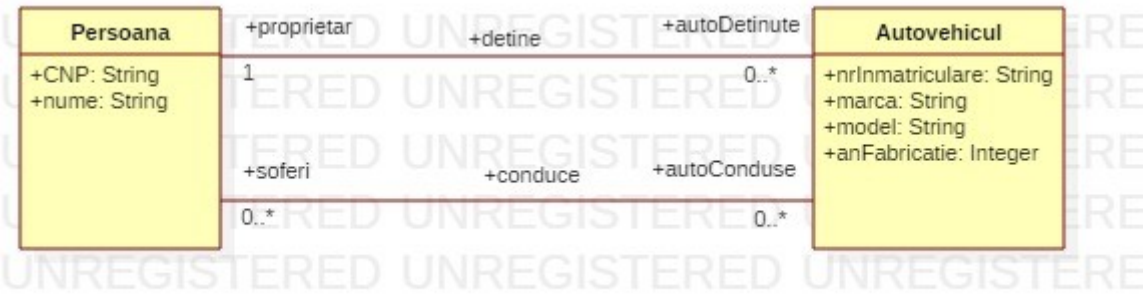
5. La nivelul codului,
+ unui atribut de tip simplu (primitiv sau referinta) ii vor corespunde operatii *get*/*set* (după sablonul *get*<numeAtribut>, *set*<numeAtribut>)

+ unui atribut de tip colecție ii vor corespunde operatii *get*/*add*/*remove* (după sablonul *get*<numeAtribut>, *addTo*<numeAtribut>, *removeFrom*<numeAtribut>)

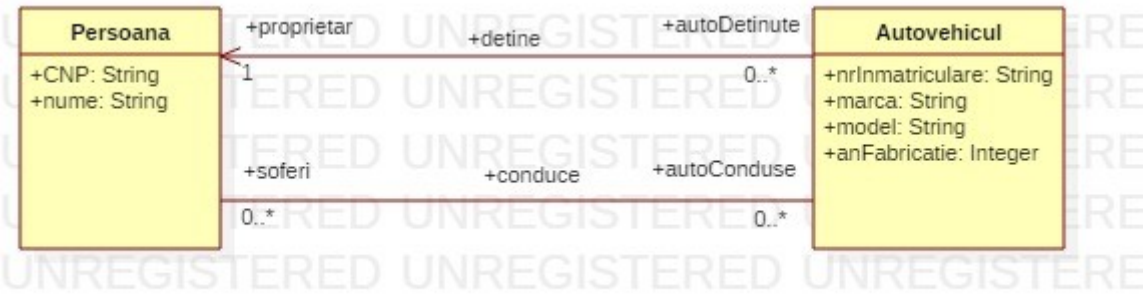
6. ! In cazul asocierilor bidirectionale, cele doua capete ale asocierii trebuie mentinute sincronizate (asociere bidirectionala = doua asocieri unidirectionale + constrangere legată de sincronizarea capetelor) - vezi codul aferent asocierilor bidirectionale din exemplul urmator

Exemplu: Generarea codului pe baza unui model structural UML constand din doua clase (Persoana și Autoturism) și doua relații posibile intre acestea (relația de proprietate și cea de utilizare/sofat) – fragment dintr-un model conceptual aferent unui sistem informatic din domeniul asigurarilor auto.

v1: ambele asocieri sunt bidirectionale



v2: prima asociere este unidirectionala, iar a doua bidirectionala



Seminar 4 **ISS 2019-2020**

Proiectarea obiectuala (detaliata). Modele dinamice ale unui sistem
Diagrame UML de secventa

Biblio:
1. Martina Seidl et al., *UML@Classroom – pp.107-139 (atasat)*

Într-un sistem orientat-obiect, obiectele comunica intre ele prin schimb de mesaje. Un *mesaj* transmis unui *obiect* are ca și efect apelul unei *metode* din clasa obiectului ținta, ceea ce determina, în general, la rândul sau, transmiterea de noi mesaje către obiectele din sistem.

Diagramele UML de secvența fac parte din categoria *diagramelor de interacțiune* (aici intra și *diagramele de comunicare*, abordate în seminarul următor). Diagramele de interacțiune surprind comportamentul dinamic al sistemului, prin prisma interactiunilor dintre obiectele care îl compun, cu scopul oferirii functionalitatii cerute.

Cele doua tipuri de diagrame de interacțiune mentionate anterior sunt echivalente; informația redata de acestea este aceeasi, doar perspectiva difera (la nivelul instrumentelor CASE, exista posibilitatea generarii automate a unui tip de diagrama din celalalt). In diagrama de secvența primeaza perspectiva temporala (secventierea, în timp, a mesajelor transmise în cadrul interactiunii), în timp ce, la nivelul diagramei de comunicare, accentul cade pe evidentierea comunicarii/legaturilor dntre obiectele participante. Alegerea unui anumit tip de diagrama depinde de tipul interactiunii: dacă numărul de obiecte participante este mare, dar numărul de mesaje transmise e mic, se recomanda utilizarea unei diagrame de comunicare (cea de secvența s-ar extinde mult pe orizontala, putinele mesaje transmise fiind greu de urmărit); dacă numărul de mesaje e mare și numărul de obiecte participante mai mic, se opteaza pentru o diagrama de secvența.

In etapa de proiectare, o diagrama de interacțiune se utilizeaza, de regula, pentru a reprezenta un scenariu asociat unui caz de utilizare (= o interacțiune concreta, în care se cunoaște numărul de pași dintr-un ciclu sau alternativa aleasa). Ea poate fi utilizata însă și pentru reprezentarea unui caz de utilizare în ansamblu (exista mecanisme de descriere a alternativelor și ciclurilor) sau a comportamentului asociat unei operatii.
(Obs,: Pentru laborator, se cere câte o diagrama de secvența pentru fiecare dintre scenariile normale ale cazurilor de utilizare descrise la faza 1.)

Exemplu* (atasat):*
- Diagrama de secvența aferenta scenariului normal al cazului de utilizare *Login* al sistemului de gestiune a comenzilor din cadrul unui restaurant (*Restaurant System* - vezi Seminar 2), corespunzand logarii cu succes a unui *Waiter* în sistem

- Fragmentul din diagrama de clase aferenta modelului structural de proiectare, incluzand clasele obiectelor care participa la interacțiunea descrisa prin diagrama de secvența de la punctul precedent și relatiile stabilite intre acestea (asocieri și dependente).

Reguli privind realizarea diagramei de secvența:

1. Fiecarui obiect participant la interacțiune ii corespunde care o *linie de viața* (*lifeline*) la nivelul diagramei, reprezentata în plan vertical.

2. Timpul curge „de sus în jos”: dacă un mesaj e reprezentat deasupra altuia în diagrama, înseamnă ca el a fost transmis, în timp, înaintea celui de-al doilea.

3. Dacă obiectul exista deja la începutul interactiunii, caseta aferenta va apărea în partea de sus a diagramei, altfel, va apărea la nivelul corespunzător momentului instantierii sale (nu e o regula obligatorie, în scris se pot reprezenta toate casetele în partea de sus, dacă acest lucru mărește inteligibilitatea diagramei; momentul crearii e oricum evident, prin transmiterea unui mesaj de tip <<create>> către obiect).

4. Mesajele se reprezintă ca și săgeți etichetate cu numele metodelor/operatiilor cărora le corespund + argumente. Exista notatii diferite pentru diferite tipuri de mesaje (sincron/asincron, create/distroy, retur etc.).

5. Mesajele pot fi numerotate (în stil imbricat, de preferinta), însă nu e obligatoriu; perspectiva temporala e implicita într-o diagrama de secvența. Numerotarea imbricata este însă foarte utila în momentul generarii diagramei de comunicare echivalente (pe diagrama de comunicare lipseste perspectiva temporala, numerotarea imbricata e obligatorie pentru a asigura echivalenta celor doua).

6. Cand un mesaj atinge linia de viața a obiectului ținta, începe așa-numitul „dreptunghi de activare", care corespunde intervalului de timp în care se executa metoda aferenta mesajului. Dacă în cadrul acelei metode se trimit mesaje către alte obiecte, acestea vor avea originea în acel dreptunghi de activare și, în numerotarea imbricata, se numeroteaza relativ la mesajul initial (dacă mesajul initial e numerotat 2.5 și, din dreptunghiul sau de activare, pleacă alte 3 mesaje, acestea vor fi numerotate 2.5.1, 2.5.2 si 2.5.3).

7. ! Atenție la dimensiunile dreptunghiurilor de activare, revenirea dintr-un apel (finalul dreptunghiului de activare aferent) ar trebui să fie înaintea revenirii din apelul din cadrul caruia a fost efectuat.

<u>Reguli privind corespondenta dintre cele doua diagrame:</u>

1. Fiecarui *obiect* care participa la interacțiunea descrisa prin diagrama de secvența (are un *lifeline* asociat) ii corespunde o *clasa* la nivelul diagramei de clase.

2. Fiecarui *mesaj* reprezentat la nivelul diagramei de secvența ii corespunde, la nivelul diagramei de clase, o *operație/metoda în clasa obiectului ținta.*

3. În cadrul diagramei de clase, reprezentarea relatiilor intre acestea are la baza urmatoarele reguli:

> a. Dacă un obiect *o1* (instanța a clasei *C1*) trimite un mesaj unui alt obiect *o2* (instanța a clasei *C2*), atunci *o1* are o referinta spre *o2*; poate fi vorba de o relație de *asociere* intre clasele corespuzatoare sau doar de o *dependentă* (*asocierea* se traduce prin date membru/atribute de timpul respectiv, *dependenta* prin utlizare de parametri, variabile locale; asocierea este „mai puternica" decât dependentă).
>
> b. Dacă cele doua obiecte mai comunica și în afara apelului de metoda în cadrul caruia a fost transmis mesajul în cauza, e necesara o asociere intre clasele aferente; altfel, e suficienta o dependentă. Navigabilitatea asocierii e dictata de sesnsul în care se transmit mesajele intre cele doua obiecte: dacă mesajele sunt transmise într-un singur sens (de la *o1* către *o2, spre exemplu*), e suficienta doar o asociere unidirectionala (navigabila dinspre C1 înspre C2), altfel, e necesara o asociere bidirectionala (navigabila în ambele sensuri)
> ! Atenție, informațiile privind comunicarea dintre obiecte se extrag analizand interactiunile la nivelul sistemului în ansamblu (prin consultarea tuturor diagramelor de secvența

realizate); o singura diagrama poate dicta doar necesitatea unei simple dependente, în timp ce analiza mai multora poate sa impuna o asociere.

c. Numele de rol utilizate în diagrama de secvența corespund celor din diagrama de clase.

d. Particularizarea asocierilor ca și agregari/compuneri se va face pe baza semanticii cunoscute a celor din urma.

Observatii:

1. Exemplul prezentat are la baza o arhitectura în care obiectele *control* corespund cazurilor de utilizare; un obiect *control* e creat de obiectul *boundary* care initiaza cazul de utilizare, urmând ca, ulterior, acesta să fie responsabil de crearea de noi obiecte *boundary* (vezi curs).

2. ! Pentru activitatea de laborator, diagramele de secvența realizate vor reflecta arhitectura aleasa pentru dezvoltarea sistemului, cu sabloanele de comunicare aferente; exemplul de seminar are doar caracter orientativ, pentru a înțelege regulile pe baza cărora (în etapa de proiectare detaliata) se elaboreaza diagramele de interacțiune și se rafineaza diagrama de clase aferenta modelului conceptual (corespunzatoare etapei de analiza).

3. În evaluarea activitatii de laborator, se va urmări sincronizarea dintre intreractiunile descrise prin diagrame de secvența în etapa de proiectare (obiecte participante, schimb de mesaje) și codul aferent. Ca urmare, în cazul în care, în etapa de implementare, se decide schimbarea unora dintre deciziile luate în proiectare, se va reveni asupra diagramelor de interacțiune, cu modificarea acestora, pentru a reflecta aceste schimbari (codul și modelele UML trebuie pastrate sincronizate!).

4. În proiectul UML de laborator, NU se va realiza câte o diagrama de clase pentru fiecare diagrama de secvența descrisa, ci informațiile colectate din toate diagramele de secvența se vor integra la nivelul unui singur model structural de proiectare. În cazul în care dimensiunea diagramei de clase aferente e mare, cu efecte negative asupra inteligibilitatii acesteia, se vor crea view-uri/diagrame diferite, segmentand astfel informația (ex.: o diagrama de clase incluzand toate clasele și relatiile dintre acestea, adnotate corespunzător, dar fără detalii privind atributele/operatiile claselor + diagrame separate ilustrand structura claselor SAU partitionarea modelului structural în mai multe diagrame de clase, fiecare acoperind o parte a sa, cu toate atributele/relatiile/operatiile corespunzatoare).

Seminar 5 **ISS 2019-2020**

Proiectarea obiectuala (detaliata). Modele dinamice ale unui sistem
Diagrame UML de comunicare – introducere
Diagrame UML de secvența - fixare

Organizare:
1h (prima) – studiu individual, pe baza materialului de seminar atașat
1h (a doua) – consultatii pe baza materialului atașat
– mail: vladi@cs.ubbcluj.ro, skype id: vladiela.petrascu

Biblio:
1. Martina Seidl et al., *UML@Classroom – pp.136 (atașat)*
2. OMG UML Spec 2.5.1 – pp. 599, https://www.omg.org/spec/UML/About-UML/

Într-un sistem orientat-obiect, obiectele comunica intre ele prin schimb de mesaje. Un *mesaj* transmis unui *obiect* are ca și efect apelul unei *metode* din clasa obiectului ținta, ceea ce determina, în general, la rândul sau, transmiterea de noi mesaje către obiectele din sistem.

Diagramele UML de comunicare fac parte din categoria *diagramelor de interacțiune* (aici intra și *diagramele de secventa*, abordate în seminarul anterior). Diagramele de interacțiune surprind comportamentul dinamic al sistemului, prin prisma interactiunilor dintre obiectele care îl compun, cu scopul oferirii functionalitatii cerute.

Cele doua tipuri de diagrame de interacțiune mentionate anterior sunt echivalente; informația redata de acestea este aceeasi, doar perspectiva difera (la nivelul instrumentelor CASE, exista posibilitatea generarii automate a unui tip de diagrama din celalalt). In diagrama de secvența primeaza perspectiva temporala (secventierea, în timp, a mesajelor transmise în cadrul interactiunii), în timp ce, la nivelul diagramei de comunicare, accentul cade pe evidentierea comunicarii/legaturilor dntre obiectele participante. Alegerea unui anumit tip de diagrama depinde de tipul interactiunii: dacă numărul de obiecte participante este mare, dar numărul de mesaje transmise e mic, se recomanda utilizarea unei diagrame de comunicare (cea de secvența s-ar extinde mult pe orizontala, putinele mesaje transmise fiind greu de urmărit); dacă numărul de mesaje e mare și numărul de obiecte participante mai mic, se opteaza pentru o diagrama de secvența.

In etapa de proiectare, o diagrama de interacțiune se utilizeaza, de regula, pentru a reprezenta un scenariu asociat unui caz de utilizare (= o interacțiune concreta, în care se cunoaște numărul de pași dintr-un ciclu sau alternativa aleasa). Ea poate fi utilizata însă și pentru reprezentarea unui caz de utilizare în ansamblu (exista mecanisme de descriere a alternativelor și ciclurilor) sau a comportamentului asociat unei operatii.
(Obs,: Pentru laborator, se cere câte o diagrama de secvența pentru fiecare caz de utilizare – pe scenariul normal sau unul dintre scenariile alternative și 3 diagrame de comunicare, la alegere, pentru scenarii reprezentate anterior cu diagrame de secvența).

***Exemplu* (atasat)**:
- Diagramele de secvența/comunicare aferente scenariului normal al cazului de utilizare *Upload Syllabus (inacrcare syllabus/fisa disciplinei pentru un curs)* al unui sistem de gestiune a activitatilor din mediul academic (*SEMS*)

- Fragmentul din diagrama de clase aferenta modelului structural de proiectare, incluzand clasele obiectelor care participa la interacțiunea descrisa prin diagramele de secvența/comunicare de la

punctul precedent și relatiile stabilite intre acestea (asocieri și dependente).

Reguli privind realizarea diagramei de comunicare:

1. Fiecarui obiect participant la interacțiune ii corespunde care o *linie de viața* (*lifeline*) la nivelul diagramei.

2. Mesajele se reprezintă ca și săgeți etichetate cu numele metodelor/operatiilor cărora le corespund + argumente.

3. Mesajele trebuie numerotate imbricat (se va găsi o alternativa, în cazul în care instrumentul CASE nu ofera aceasta functionalitate).

Observatii:

1. Exemplul prezentat are la baza o arhitectura în care obiectele *control* corespund cazurilor de utilizare; un obiect *control* e creat de obiectul *boundary* care initiaza cazul de utilizare, urmând ca, ulterior, acesta să fie responsabil de crearea de noi obiecte *boundary* (vezi curs).

2. ! Pentru activitatea de laborator, diagramele de secvența/comunicare realizate vor reflecta arhitectura aleasa pentru dezvoltarea sistemului, cu sabloanele de comunicare aferente; exemplul de seminar are doar caracter orientativ, pentru a înțelege regulile pe baza cărora (în etapa de proiectare detaliata) se elaboreaza diagramele de interacțiune și se rafineaza diagrama de clase aferenta modelului conceptual (corespunzatoare etapei de analiza).

3. În evaluarea activitatii de laborator, se va urmări sincronizarea dintre diagramele de comunicare și diagramele de secvența corespunzatoare, precum și intre diagramele de interacțiune și cod.

4. Relativ la realizarea diagramelor aferente fazei de proiectare cu ajutorul instrumentului StarUML (se considera o structura a proiectului conforma template-ului UML Conventional):

- tot ceea ce tine de proiectare (model structural - diagrama de clase rafinata + model dinamic - diagrame de secventa și comunicare) trebuie inclus in proiect in partea de Design Model (anterior, modelul functional a fost inclus la sectiunea UseCase Model, iar cel conceptual la sectiunea Analysis Model).

- trebuie lucrat in paralel pe partea de model structural si model dinamic, dupa cum urmeaza:

a. Se dorește crearea unei diagrame de secventa si e nevoie de un lifeline care sa reprezinte un actor => se adauga un lifeline pe diagrama si lui i se va asocia un rol – Role1, de ex. Se acceseaza Role1 - acesta are ca și proprietate un type. Pentru acel type, se navigheaza in model si se selecteaza actorul din modelul functional (Use Case Model) de care e nevoie. După care, daca acesta nu e reprezentat implicit ca si actor, click dreapta > Format > Stereotype Display > Icon.

b. E nevoie de un lifeline care sa reprezinte un obiect din aplicatie => se adauga pe diagrama acel lifeline, lui i se asociaza un rol - Role2. Se adauga mai intai pe diagrama de clase aferenta modelului structural de proiectare clasa a carui instanta e respectivul obiect. Dupa care se seteaza atributul type al lui Role2 la acea clasa (selectata din Design Model). Se adauga din nou si clasele din modelul conceptual, fiindca va fi nevoie ca acestea să fie rafinate aici (prin adaugare de operatii, atribute, relatii, etc.).

c. Numele de roluri setate pentru lifeline-urile din diagrama de secvența vor corespunde numelor de

rol folosite în adnotarea relațiilor de asociere din diagrama de clase aferenta modelului structural de proiectare. Ele pot fi omise, dacă nu sunt relevante (pentru actori, de ex.).

d. E nevoie de un mesaj intre doua lifeline-uri => se adauga obiectul mesaj din toolbar pe diagrama, dupa care acesta trebuie etichetat cu un nume de metoda. Se acceseaza mai intai clasa obiectului tinta, se adauga acea operatie, dupa care se revine la mesaj si i se seteaza atributul signature, alegand din model operatia nou adaugata. Ulterior, pot fi adaugate și argumente (daca e cazul), pentru ca logica interactiunii sa poată fi urmarita în mod natural.

Organizare:
1h (prima) – studiu individual, pe baza materialului de seminar atașat
1h (a doua) – consultatii pe baza materialului atașat
– mail: vladi@cs.ubbcluj.ro, skype id: vladiela.petrascu

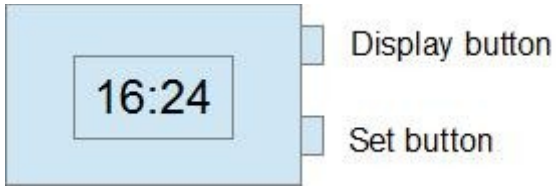
I. Proiectarea obiectuala (detaliata). Modele dinamice ale unui sistem Diagrame UML de tranzitie a starilor

Biblio:
1. Curs 2
2. Martina Seidl et al., *UML@Classroom* – pp. 85 (atașat)
3. E. Gamma et al., *Design Patterns. Elements of Reusable Object-Oriented Design* - State Pattern (atasat)
4. *OMG UML Spec 2.5.1* *https://www.omg.org/spec/UML/About-UML/* - pp.305 (pentru aspecte punctuale)

La fel ca și diagramele de interacțiune (si cele de activități), diagramele UML de tranzitie a starilor sunt instrumente folosite pentru a reprezenta modele dinamice ale unui sistem.
Dacă diagramele de secvența/comunicare surprind comportamentul dinamic prin prisma obiectelor care iau parte la interacțiune și a schimbului de mesaje intre acestea, diagramele de tranzitie a starilor surprind comportamentul prin prisma starilor prin care trece un obiect (subsistem/grup de obiecte relationate) de-a lungul ciclului sau de viața și a tranzitiilor posibile intre aceste stari. Astfel de tranzitii pot fi declansate de aparitia unor evenimente externe, de indeplinirea anumitor condiții sau de trecerea unui interval de timp.

O diagrama de interacțiune se realizeaza, de regula, pentru un scenariu al unui caz de utilizare. Diagramele de tranzitie a starilor se asociaza claselor (grupurilor de clase) din sistem, însă nu tuturor, ci doar celor caracterizate de un comportament dinamic semnificativ (răspund diferit la stimulii din mediu funcție de starea în care se afla).

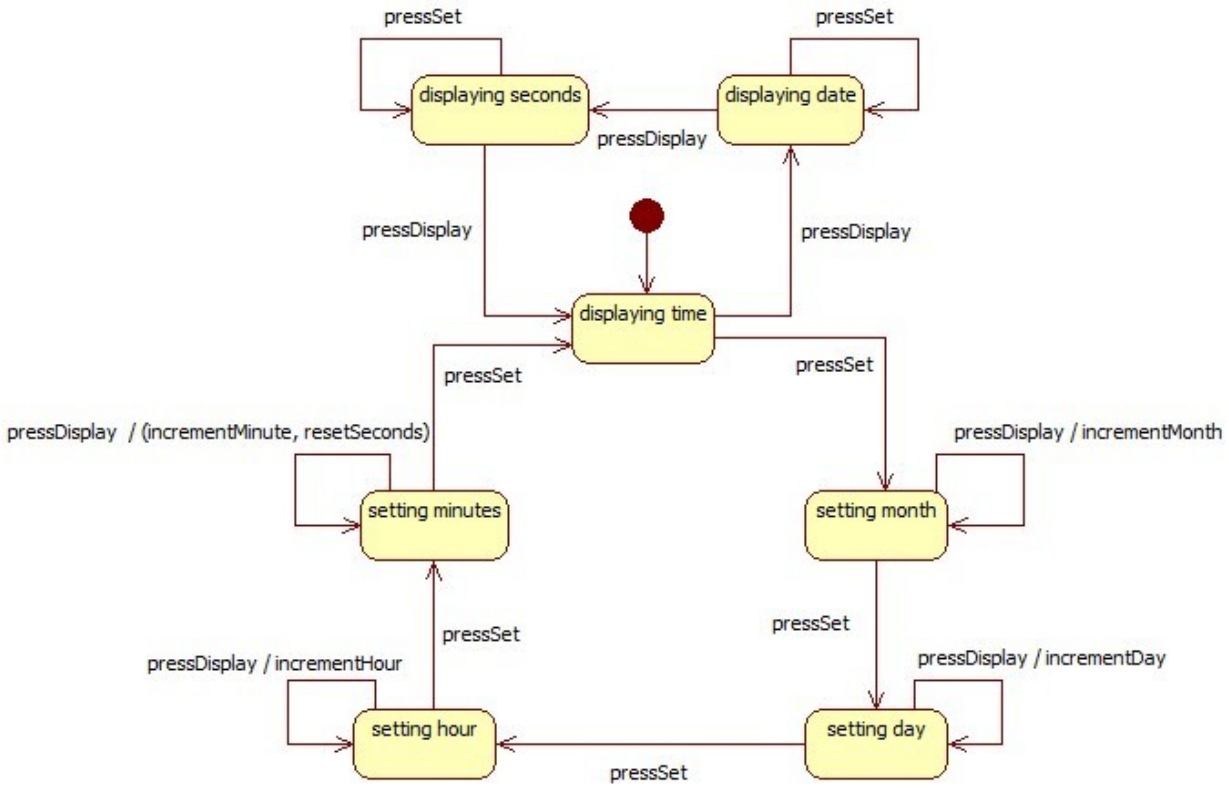
Exemplu: Consideram exemplul unui ceas electronic simplu, care ofera functionalitati de vizualizare a timpului curent (ora, minut, secunde) și a datei curente (zi, luna) și de setare a diferitelor sale componente (ora, minut, luna, zi – secundele nu se seteaza manual, ci se reseteaza la fiecare schimbare a minutului).



În interfata, ceasul ofera utilizatorului doua butoane, unul pentru afisaj (Display button) și unul pentru reglaje (Set button).
După inserarea bateriei, ceasul intra în modul de afisaj al timpului curent (în formatul *ora* :

minut). Din aceasta stare, prin actionarea repetata a butonului Display, se permite ciclarea intre starile de afisaj: afisare ora -> afisare data (format *zi* : *luna*) -> afisare secunde (format _ : *secunde*) -> afisare ora ... s.a.m.d. În starile de afisaj a date și a secundelor, actionarea butonului Set nu are nici un efect.
Din starea de afisaj a timpului curent, prin actionarea repetata a butonului Set, se cicleaza între starile de reglaj: reglaj luna -> reglaj zi -> reglaj ora -> reglaj minut -> afisare timp -> reglaj luna ... s.a.m.d. Într-o stare de reglaj, actionarea butonului Display permite incrementarea componentei curente cu o unitate (pana la capătul intervalului aferent de valori și apoi de la capăt). Într-o stare de reglaj, componenta curenta reglata este afisata intermintent.

Diagrama de tranzitie a starilor corespunzatoare acestui comportament este următoarea (acțiunile utilizatorului de apasare a celor doua butoane corespund evenimentelor care declanseaza tranzitiile intre stari; în starile de reglaj, tranzitiile generate de apasarea butonului Display au acțiuni asociate):



Un astfel de ceas prezinta un comportament dinamic dependent de stare: un același eveniment (apasarea unui buton) are efecte diferite, funcție de starea curenta în care se afla obiectul.
Considerand o structura a ceasului care conține o parte de interfata, un control (ClockController) și o memorie asociata (ClockMemory), diagrama de tranzitie a starilor anterioara va corespunde clasei de tip control.

Variante de proiectare a clasei ClockController:

1. ClockController reține o referinta catre obiectul memorie asociat și un atribut de tip enumerare/intreg, corespunzător starii sale curente. Evenimentele și acțiunile de pe

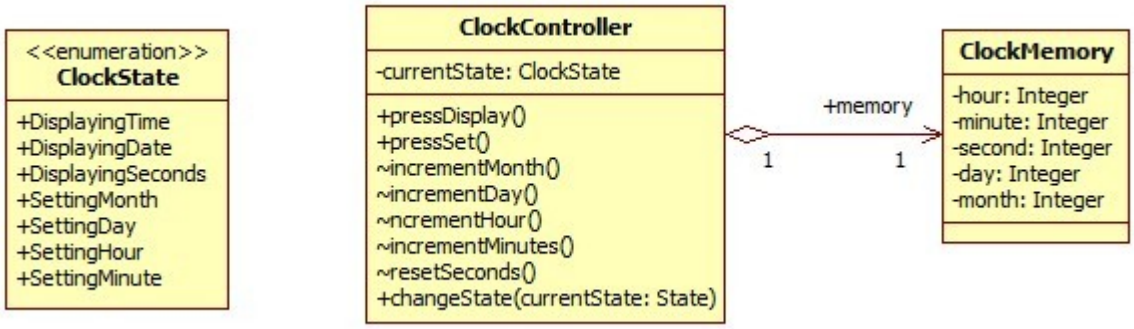
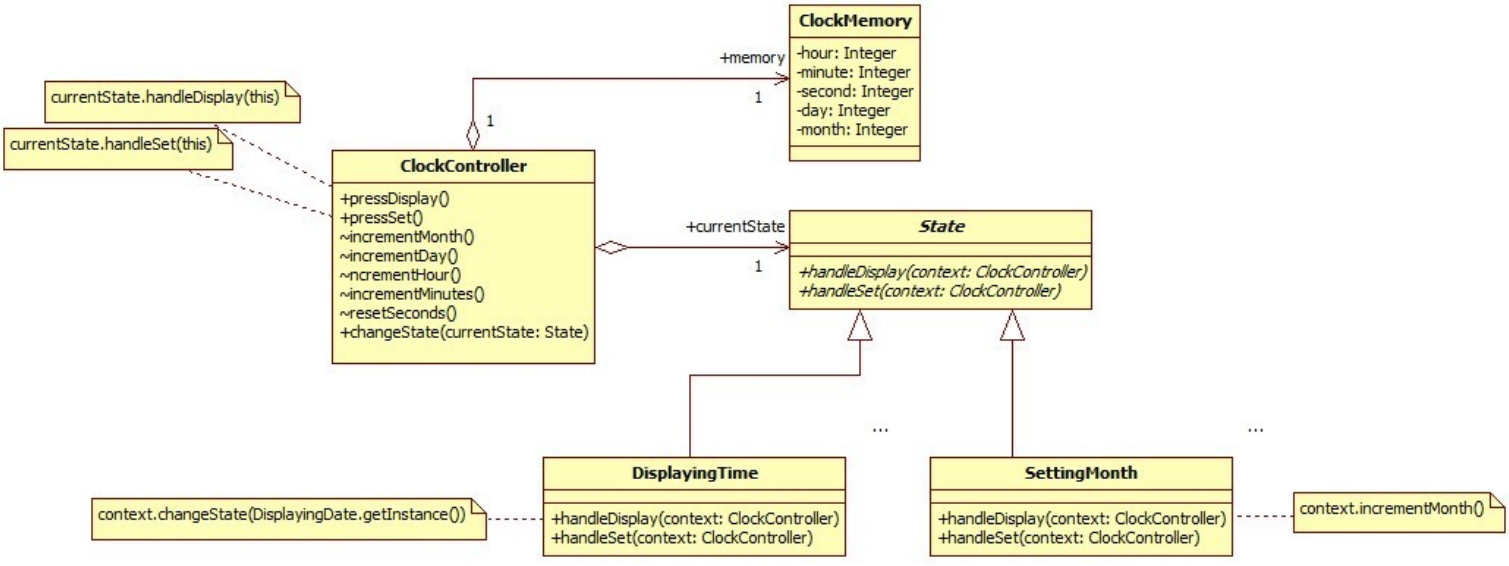


diagrama de tranzitie a starilor corespund unor operatii la nivelul clasei.
În acest caz, logica operatiilor pressDisplay() și pressSet() va folosi structuri conditionale (if/switch/case) pentru a gestiona schimbarile de stare și eventualele acțiuni asociate. Necesitatea de a lua în considerare o stare noua ulterior fnalizarii sistemului va determina modificari la nivelul acelor structuri conditionale, în toate metodele corespunzatoare evenimentelor.

? Ce pricipiu de proiectare enuntat la curs încalca aceasta abordare? Pentru bonus de seminar :), trimiteți răspunsul la aceasta întrebare și explicatiile aferente pe adresa vladi@cs.ubbcluj.ro, pana la finalul seminarului grupei de care apartineti.

2. ClockController reține o referinta catre obiectul stare asociat (instantiere a sablonului de proiectare State, vezi [3] pentru detalii). Obiectul control delega acestui obiect stare gestionarea evenimentelor (prin metodele de tip handler). Pentru a putea fi gestionata schimbarea de stare, obiectul context (ClockController) se transmite pe sine ca și parametru al metodelor handler.
În acest mod, logica de gestionare a evenimentelor este incapsulata la nivelul obiectelor stare concrete. Luarea în considerare a unei stari noi presupune crearea unui nou obiect stare concret și implementarea handlerelor corespunzatoare.



II. Specificarea constrangerilor pe modelele UML folosind limbajul OCL

Biblio:
1. Curs 9
2. *OMG OCL Spec 2.4* *https://www.omg.org/spec/OCL* (pentru aspecte punctuale)

Vezi cerintele și soluția din directorul *ocl* atașat.