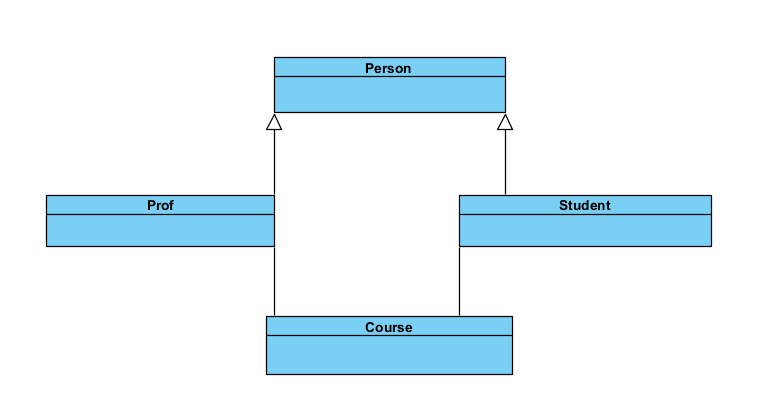
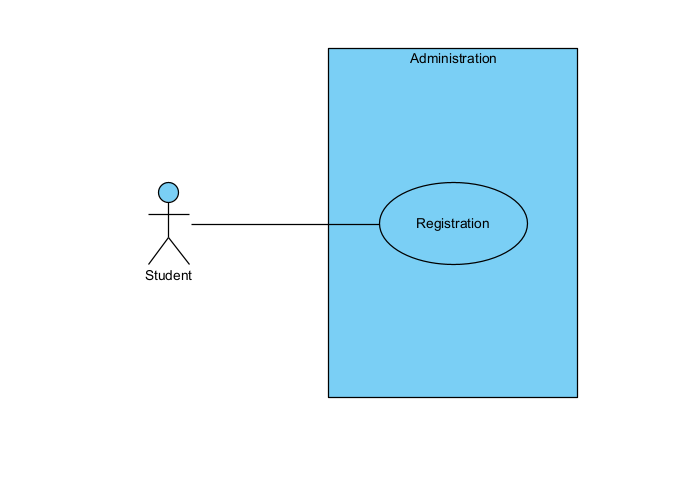
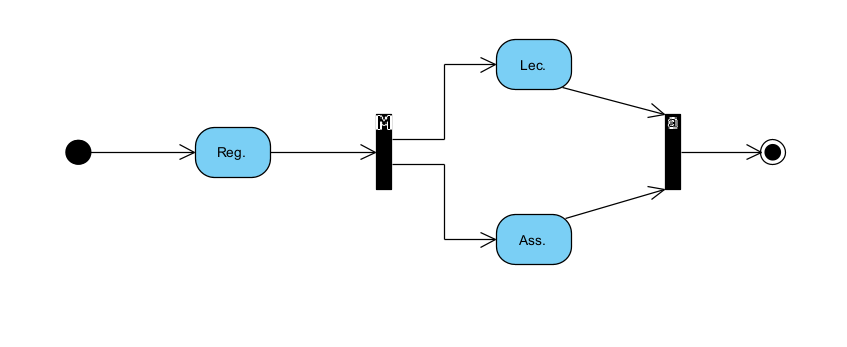
1. Ce este UML și pentru ce este utilizat?
   1. UML (Unified Modeling Language) este un limbaj de modelare grafic orientat-obiect. UML este potrivit pentru modelarea atât a relațiilor complexe între obiecte cât și a proceselor cu concurență. UML este un limbaj de modelare cu scop general, ceea ce înseamnă că utilizarea sa nu este restricționată la o anumită zonă de aplicație. Furnizează concepte de limbaj și de modelare, precum și o notare grafică intuitivă pentru modelarea diferitelor domenii de aplicație, permițând specificarea, proiectarea, vizualizarea și documentarea unui sistem software. Rezultatul modelării cu UML este un model grafic care oferă diferite viziuni ale unui sistem sub formă de diverse diagrame.
   2. UML nu este legat de un instrument de dezvoltare specific, de un limbaj de programare specific sau de o platformă țintă specifică pe care sistemul dezvoltat trebuie să fie utilizat. De asemenea, UML nu oferă un proces de dezvoltare software. În realitate, UML separă limbajul de modelare și metoda de modelare. Aceasta din urmă poate fi definită pe baza unui proiect sau a unei companii. Cu toate acestea, conceptele de limbaj ale UML favorizează un proces iterativ și incremental.
   3. UML poate fi utilizat în mod consecvent pe întregul proces de dezvoltare a software-ului. La toate etapele de dezvoltare, aceleași concepte de limbaj pot fi folosite în aceeași notație. Astfel, un model poate fi rafinat în etape. Nu este necesar ca un model să fie tradus într-un alt limbaj de modelare. Acest lucru permite un proces de dezvoltare software iterativ și incremental. UML este potrivit pentru diverse domenii de aplicație cu diferite cerințe referitoare la complexitate, volum de date, timp real, etc.
   4. Elementele modelelor UML și utilizarea corectă a acestora sunt specificate în metamodelul UML. Conceptele limbajului sunt definite într-un mod atât de general încât se obține o aplicabilitate largă și flexibilă. Pentru a evita restricționarea utilizării UML, standardul este (intenționat) vag în diferite puncte, permițând diferite interpretări sub formă de puncte de variație semantică. Cu toate acestea, acest lucru este o sabie cu două tăișuri; duce și la diferite implementări ale standardului de limbaj de către instrumentele de modelare, ceea ce, la rândul său, din nefericire, face dificilă schimbul de modele.
2. Ce sunt modelele și care este utilitatea lor?
   1. Modelele ne permit să descriem sistemele eficient și elegant. În 1973, Herbert Stachowiak a propus o teorie a modelelor care se deosebește prin trei caracteristici:
      1. Mapare: un model este întotdeauna o imagine (mapare) a ceva, o reprezentare a originalului natural sau artificial care poate fi în sine un model.
      2. Reducție: un model nu capturează toate atributele originalului, ci doar pe cele care par relevante pentru modelator sau utilizatorul modelului.
      3. Pragmatism: pragmatismul înseamnă orientarea către utilitate. Un model este asignat unui original pe baza următoarelor întrebări: Pentru cine? De ce? Pentru ce? Un model este folosit de către modelator sau utilizator în locul originalului într-un cadru de timp specific și pentru un scop specific.
   2. Modelele sprijină o reprezentare a unui sistem redus la esențial pentru a minimiza complexitatea sistemului până la aspectele gestionabile. De obicei, un sistem este descris nu printr-o singură vedere, ci printr-un număr de vederi care împreună produc o imagine generală unificată. Astfel, o vedere ar putea descrie obiectele implicate și relația lor unul cu celălalt; o altă vedere ar putea descrie comportamentul unui grup de obiecte sau prezenta interacțiunile dintre diferite obiecte.
3. Familiarizarea cu mediul de lucru mai sus amintit prin crearea anumitor diagrame din sectiunile 2.3.1 și 2.3.2 din [1]. Mai exact:
   1. Class diagram – diagrama de clasă



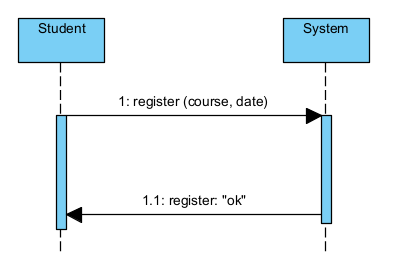
* 1. Use case diagram – diagrama de cazuri de utilizare



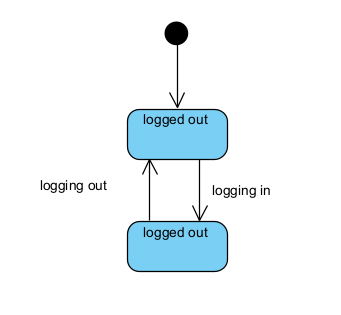
* 1. Activity diagram – diagrama de activitate



* 1. Sequence diagram – diagrama de secvente



* 1. State machine diagram – diagrama mașinii de stare



1. Ce tipuri de diagrame sunt prezentate în [1] – Capitolul 2? Care sunt asemănarile și deosebirile între acestea?
   1. Cartea ne prezintă în capitolul 2 cele mai importante și mai răspândite cinci diagrame UML, și anume diagrama de cazuri de utilizare, diagrama de clasă (inclusiv diagrama de obiect), diagrama de automat finit de stări, diagrama de secvențe și diagrama de activități.
   2. Diagrama de cazuri de utilizare, care specifică funcționalitatea de bază a unui sistem software.
   3. Diagrama de clasă definește apoi ce obiecte sau ce clase sunt implicate în realizarea acestei funcționalități.
   4. Diagrama de automat finit de stări definește comportamentul intra-obiect.
   5. Diagrama de secvențe specifică comportamentul inter-obiect.
   6. Diagrama de activități ne permite să definim acele procese care "implementează" cazurile de utilizare din diagrama de cazuri de utilizare.
   7. Diagrama de Cazuri de Utilizare (Use Case Diagram) și Diagrama de Clasă (Class Diagram):
      1. Asemănări:
         1. Ambele diagrame sunt utilizate în fazele de definire și proiectare a sistemelor software.
      2. Diferențe:
         1. Diagrama de cazuri de utilizare se concentrează pe interacțiunea între utilizatori și funcționalitățile sistemului, în timp ce diagrama de clasă se concentrează pe structurile de date și obiecte ale sistemului.
   8. Diagrama de Clasă (Class Diagram) și Diagrama de Obiect (Object Diagram):
      1. Asemănări:
         1. Ambele diagrame se bazează pe modele conceptuale de date și pe dezvoltarea software orientată pe obiecte.
      2. Diferențe:
         1. Diagrama de clasă definește structurile de clase și relațiile dintre ele, în timp ce diagrama de obiect oferă o imagine concretă a stării sistemului la un moment dat.
   9. Diagrama de Stare (State Machine Diagram) și Diagrama de Secvențe (Sequence Diagram):
      1. Asemănări:
         1. Ambele diagrame descriu comportamentul și interacțiunile între obiecte în cadrul sistemelor software.
      2. Diferențe:
         1. Diagrama de stare se concentrează pe modul în care obiectele trec prin diferite stări în timpul ciclului lor de viață, în timp ce diagrama de secvențe se concentrează pe ordinea cronologică a mesajelor schimbate între obiecte pentru a îndeplini o anumită sarcină.
   10. Diagrama de Activități (Activity Diagram) și Diagrama de Secvențe (Sequence Diagram):
       1. Asemănări:
          1. Ambele diagrame pot fi utilizate pentru a modela interacțiunile și fluxul de lucru în cadrul sistemelor software.
       2. Diferențe:
          1. Diagrama de activități se concentrează pe procesele și fluxurile de lucru, în timp ce diagrama de secvențe se concentrează pe interacțiunile specifice între obiecte pentru a îndeplini o anumită sarcină.
2. Caracterizați succint fiecare diagrama prezentată în [1] – Capitolul 2 (cele 5 menționate mai sus)
   1. Diagrama de Cazuri de Utilizare (Use Case Diagram):
      1. Această diagramă este utilizată pentru a defini cerințele pe care un sistem trebuie să le îndeplinească. Ea descrie funcționalitățile sistemului și utilizatorii care le utilizează, fără a intra în detalii specifice de implementare. Principalele entități de funcționalitate oferite de sistem sunt numite cazuri de utilizare și sunt exemplificate într-un sistem de administrare universitară prin funcționalitatea "Înregistrare", utilizată de studenți.
   2. Diagrama de Clasă (Class Diagram):
      1. Această diagramă este folosită pentru a specifica structurile de date și obiecte ale unui sistem. Ea se bazează pe concepte precum clasă, generalizare și asociere. De exemplu, într-o diagramă de clasă, pot fi modelate clasele Curs, Student și Profesor dintr-un sistem. Profesorii predau cursuri, iar studenții participă la cursuri. Profesorii și studenții au proprietăți comune, exprimate prin relații de generalizare.
   3. Diagrama de Obiect (Object Diagram):
      1. Această diagramă oferă o imagine concretă a stării unui sistem la un moment dat. Bazată pe definițiile diagramelor de clasă, ea prezintă o imagine instantanee a obiectelor și relațiilor lor la un moment specific. De exemplu, o diagramă de obiect ar putea arăta că profesorul Henry Foster predă cursurile "Modelare Orientată pe Obiecte" și "Programare Orientată pe Obiecte".
   4. Diagrama de Stare (State Machine Diagram):
      1. Această diagramă descrie comportamentul obiectelor în cadrul ciclului lor de viață, evidențiind tranzițiile între diferite stări. De exemplu, o persoană poate fi în starea "neconectat" când vizitează pentru prima dată un site web, iar starea se schimbă în "conectat" după ce introduce cu succes numele de utilizator și parola. Tranzițiile între stări sunt declanșate de evenimente precum conectarea sau deconectarea.
   5. Diagrama de Secvențe (Sequence Diagram):
      1. Această diagramă descrie interacțiunile între obiecte pentru a îndeplini o anumită sarcină, precum înregistrarea pentru un examen într-un sistem de administrare universitară. Ea se concentrează pe ordinea cronologică a mesajelor schimbate între partenerii de interacțiune, oferind diverse mecanisme pentru controlul acestei ordini și modularizarea interacțiunilor complexe.
   6. Diagrama de Activități (Activity Diagram):
      1. Această diagramă este utilizată pentru a modela procese de orice fel, inclusiv procese de afaceri și procese software. Ea oferă mecanisme pentru coordonarea acțiunilor care alcătuiesc un proces, inclusiv fluxul de control și fluxul de date. De exemplu, o diagramă de activități poate arăta acțiunile necesare pentru ca un student să participe la o lectură și la o temă.
3. Folosind Visual Paradigm, realizați diagramele de clasă din Figurile 1 și 2.

