**GHID DE UTILIZARE**

**Enterprise Architect (EA)**

Elaborat: st.gr. TI-183, Procopenco Natalia

# **3. Descrierea principiului de lucru al EA**

Enterprise Architect (EA) cu logo-ul în figura 11 nu este o simplă aplicație pentru reprezentarea diagramelor și desenarea acestora, este un instrument care depozitează toate datele din diagrame, este o aplicație care poate fi utilizată doar pentru 30 de zile gratuit după care necesită cumpărarea licenței.

Figura 11 – Logo-ul EA

Aici o entitate poate să fie utilizată în mai multe diagrame, adică nu este necesară crerea elementelor noi de fiecare dată, dacă acestea deja există undeva și modificarea în una din diagrame face modificarea în toate celelalte unde a fost utilizată. Relațiile dintre entități nu sunt simple reprezentări ale săgeților ci conectivitatea acestora și sensul funcționării lor.

Fereastra Browser Window este o reprezentare a modelului în sine (descrierea detaliată a acesteia este descrisă în cadrul capitolului 3). Toolbox-ul este o paletă cu reprezentarea tuturor entităților și relațiilor specifice tipului de diagramă ales (și descrierea lui este atașată mai jos).

EA există în mai multe versiuni care au fost actualizate de-a lungul timpului. Prima versiune apărută pe internet a fost 1.1. Pe parcurs aceasta a fost modificată dar printre cele mai vechi versiuni se enumeră 2.x, 3.x, 4.x, 5.0 ș.a dar ultimile sunt 15.0, 15.1 și 15.2. Sparx Systems au adăugat din ce în ce mai multe funcționalități instrumentului de modelare UML. Ultima versiune este bazată pe UML 2.5.

**Important!**

După descărcarea EA și deschiderea acestuia trebuie creat sau deschis un proiect, interfața ce apare inițial este reprezentată în figura 12. Din momentul în care prima dată este utilizat EA, trebuie creat un repozitoriu pentru salvarea datelor proiectului creat. Din start se poate crea o mapă și deja în ea să fie salvate datele.

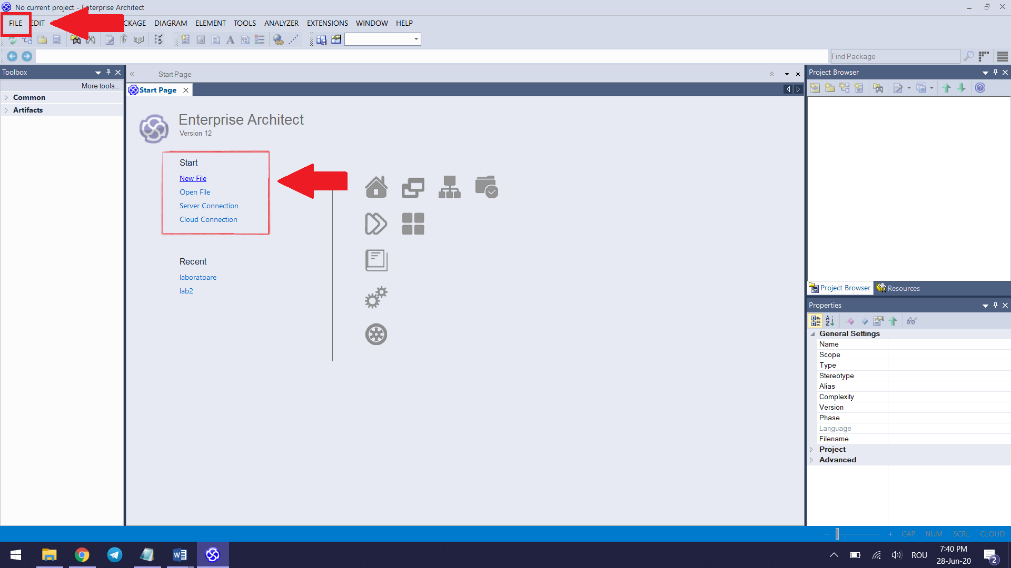


Figura 12 – Interfața EA

Crearea unui proiect se face prin selectarea New File de pe Start Page a Enterprise Architect, urmat de indicarea unui nume al proiectului și selectarea repozitoriului. O altă modalitate ar fi selectarea butonului File/New Project/ Nume + Repozitoriu. O a treia variantă este hotkey Ctrl+N urmat de aceeași pași. Butoanele ce trebuie apăsate pentru creare sunt indicate cu pătrate roșii și săgeți în figura 12.

După creare proiectului, EA îl afișează pe acesta (gol) pentru ca ulterior să poată fi început lucrul. La pasul dat se afișează fereastra “Select Mode” din figura 13, în care trebuie selectate modelele diagramelor care o să fie create.

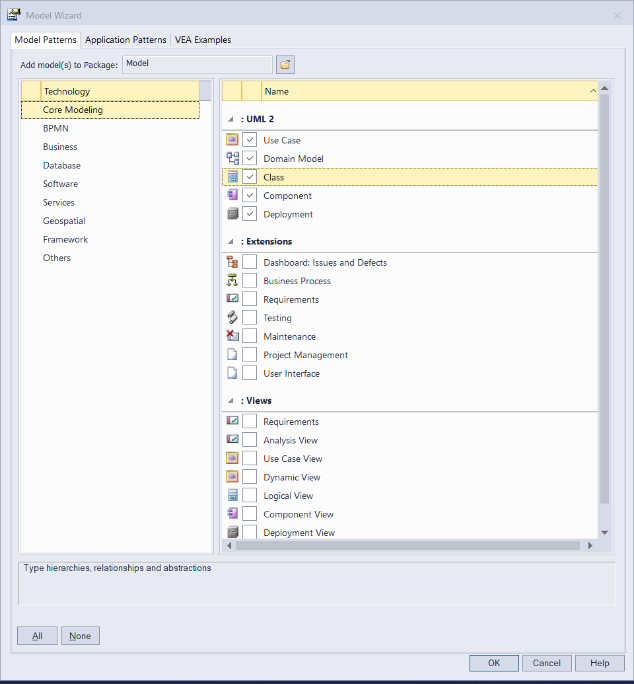


Figura 13 – Fereastra Select Mode

După selectare câtorva dintre tipurile de modele pe care le oferă EA (ulterior celelalte se creează manual ca pachete separate), în fereastra ”Project Browser” din figura 14 sunt afișate toate pachetele create. De asemenea în această ferestră întotdeauna sunt afișate toate diagramele și componentele acesteia.

Crearea manuală a unui pachet se face prin accesarea butonului în formă de o mapă indicat în figura de mai jos, partea dreaptă, după care este afișată ferestra din partea stângă a figurii 14, în care trebuie indicat numele acestui pachet (în mod obligatoriu) și tastarea ok.

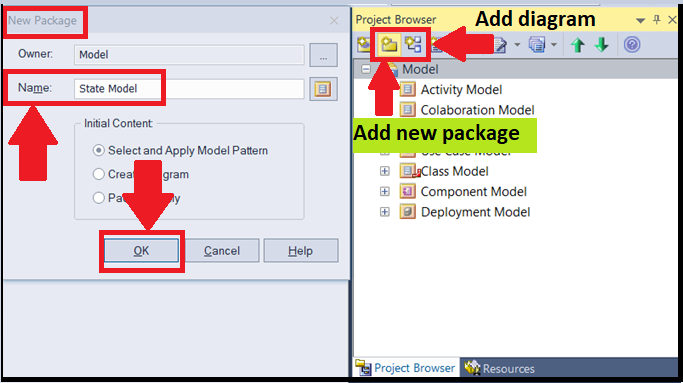


Figura 14 – Project Browser/Creare pachet

După selectarea ”ok” se deschide fereastra ”Select Mode” din nou, însă de această dată se accesează ”ok” fără nici o selectare pentru a avea acest pachet gol.

În acest pachet gol urmează adăugarea unui tip de diagramă. Pentru a efectua acest procedeu, înițial trebuie selectat pachetul în care o să fie salvată, apoi apăsarea butonului de adăugare a diagramei indicat cu roșu în figura 14. După accesarea acesteia urmează ferestra în care se enumeră tipurile de diagrame pe care le putem selecta (figura 15).

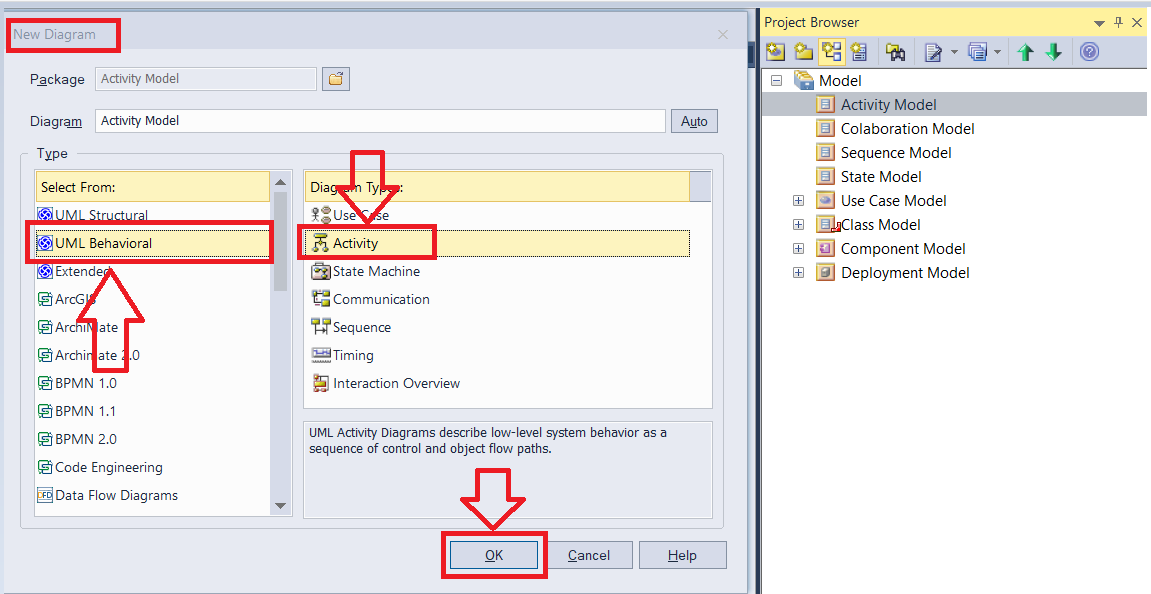


Figura 15 – Creare diagramă în pachet gol

Din fereastra pentru selectarea tipului de diagramă din figura 15 din stânga, se selectează din ce tip face parte ea, apoi diagrama însăși în confomitate cu ce trebuie creat. Pașii ce trebuie urmați sunt reprezentați în figura de mai sus

Fiecare pachet, diagramă sau proiect poate fi modificat. Pentru modificarea unui detaliu al diagramei sau a unei entități se face dând dublu click pe ea, sau o alternativă ar fi dând click dreapta pe ceea ce dorim să modificăm și să fie selectat butonul ”Properties” din fereastra afișată.

Ștergerea unei diagrame sau pachet are loc prin accesarea butonului ”delete” de pe tastatura calculatorului sau dând click dreapta și selectând ”delete x-element”.

Selectând din bara standard de instrumente butonul ”Edit” (figura16), sunt afișate hotkeys-urile pentru o navigare mai ușoară în cadrul proiectului.

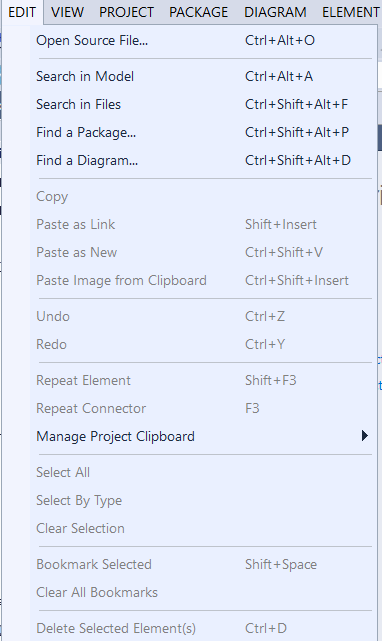
****

Figura 16 – Posibilitățile butonului ”Edit”

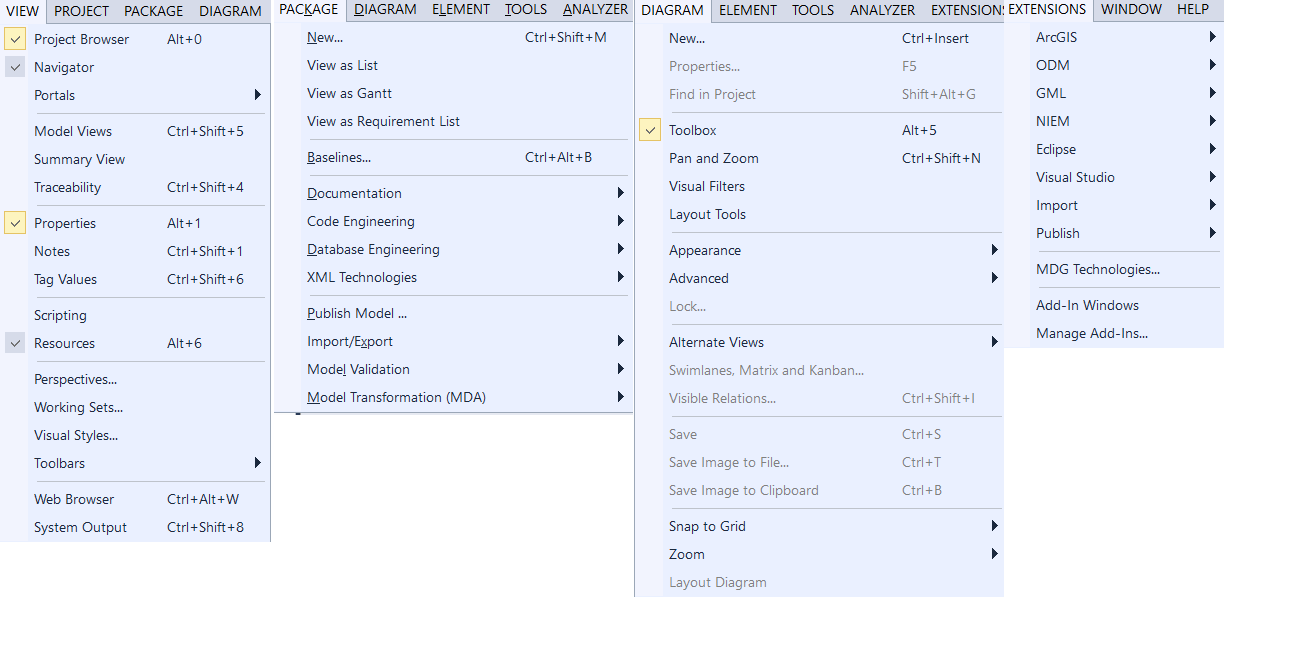
Selectând ”Diagram” cu reprezentarea în figura 17 observăm funcționalitățile pe care le oferă și anaume crearea diagramei, păstrarea toolbox-ului pe ecran, zoom-ul dar și hotkeys-urile pentru o coordonare a acțiunilor mai ușoară. De asemenea ceea ce oferă ****butoanele ”View”, ”Package”, ”Extension”.

Figura 17 – Butoane din Bara standard

Diagrama cazurilor de utilizare face parte din ”Use Case Model” și este alcătuită din usecase-uri, actori și relațiile de asociere, generalizare, dependență. Un exemplu al unei diagrame este în figura 18.

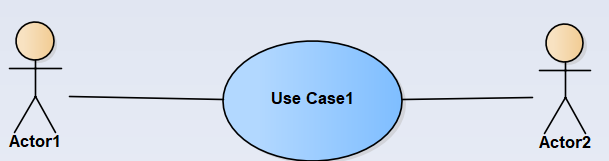
****

Figura 18 – Use case

Diagrama de secvențe este alcătuită din actori, obiecte, relații sincrone, asincrone, de răspuns, linie de viață și o reprezentare ale entităților este în figura 19.

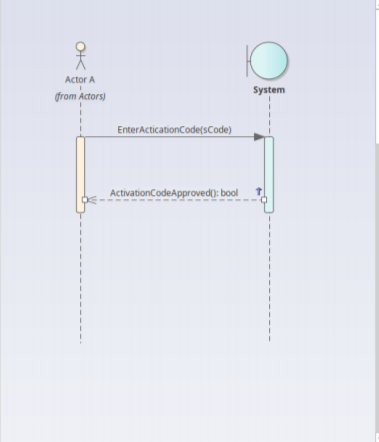


Figura 19 – Diagrama de secvențe

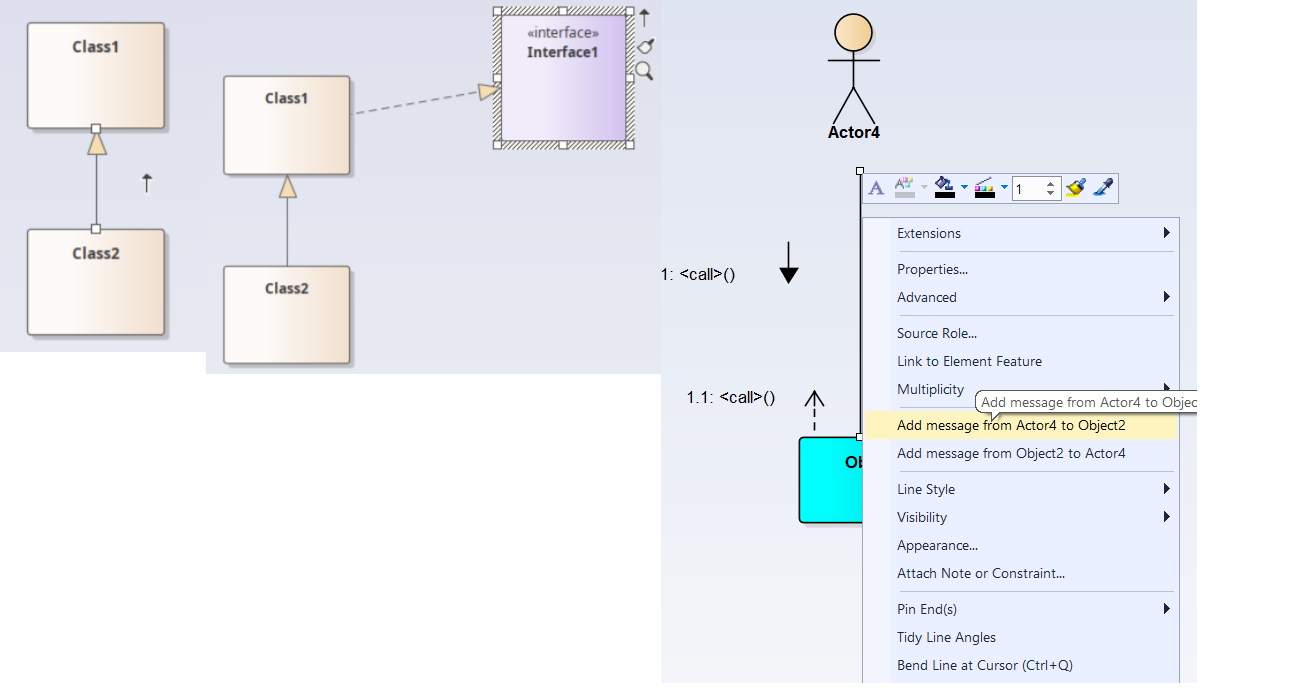
**** Pentru diagramele de colaborare și clase exemplul de diagramă este anexat în figura 20. Diagrama de colaborare conține relații sincrone, asincrone și de răspuns între obiect și actor sau obiect și obiect. Cea de clasă relații de realizare cu interfața, de generalizare, dependență și cazurile particulare ale asocierii, agregarea și compoziția în cazul relațiilor dintre clasă și clasă.

Figura 20 – Diagrama de colaborare și clase

În diagrama de colaborare call-urile sunt reprezentate pe relația de asociere dintre entități. Pentru a adăuga un call trebuie tastat click dreapta și selectat de la cine spre cine va fi acesta. Acest moment este reprezentat în figura 20 partea draptă și anume după tastare, opțiunile posibile.

În figura 21 este reprezentată fereastra cu proprietăți ale unei clase. Pentru a adăuga câmpurile cu atributele și metodele clasei trebuie selectat butonul ”details” urmat de ”attributes”. Se deschide apoi o fereastră în care trebuie întroduse datele despre variabilele și funcțiile clasei resprective.

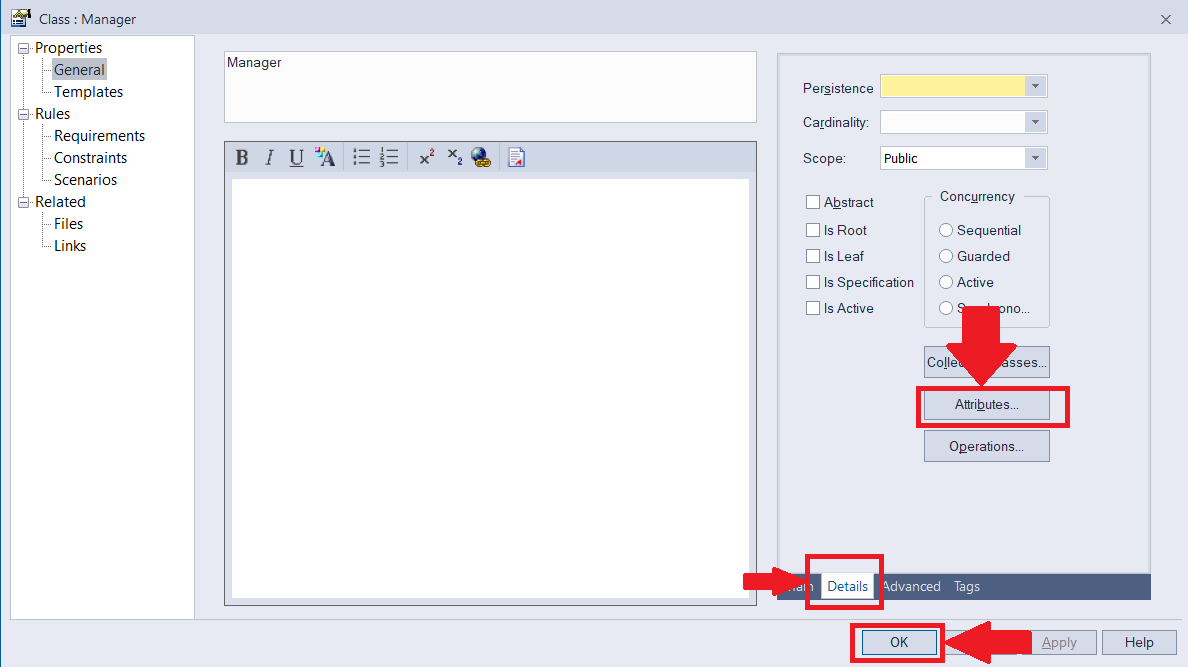
****

Figura 21 – Adăugare atribute în clasă

După tastarea ”ok” atributele și metodele își ocupă poziile în câmpurile sale și clasa arată ca și în figura 20 pe stânga.

Pentru a genera codul unei clase, trebuie selectată clasa propriu-zisă. Dând click dreapta pe ea se deschid opțiunile din figura 22. Aici există butonul ”Code Engineering” care la rândul său conține butonul de generare a codului. Tastând acest buton, se deschide fereastra în care trebuie indicat repozitoriul și de selectat în care limbaj o să fie generat acesta.

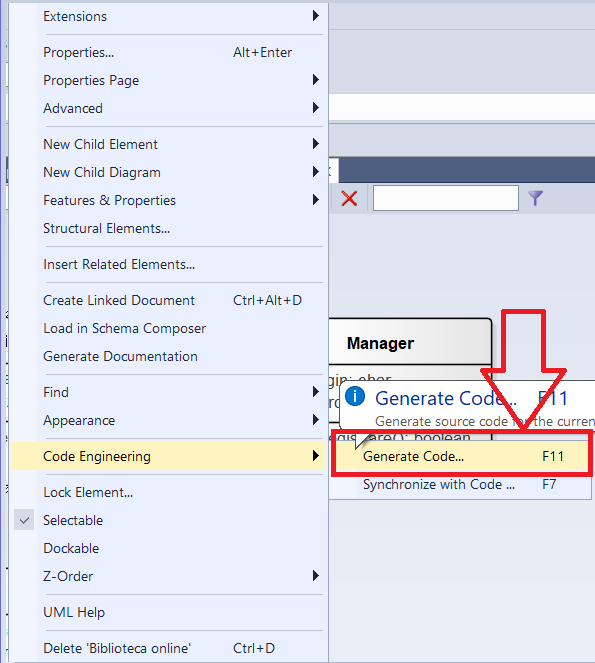


Figura 22 – Generare cod clasă

Mai sus a fost explicat principiul de lucru al instrumentului EA. Funcționalitățile de bază, exemple ale diagramelor, crearea acestora, a unui proiect, a unui pachet.Pe un exemplu concret vor fi explicate fiecare tip de diagramă în capitolul 4, cu explicarea fiecărei entități.

# **4. Demonstarea principiului de funcționare**

A fost realizat un exemplu de proiect cu ajutorul instrumentului Enterprise Architect și modelat în cadrul acestuia o bibliotecă online. Prima diagramă și anume use case este diagrama care descrie funcționalitățile de bază ale sistemului.

**Cazul de utilizare** implică interacţiunile dintre actori şi sistem reprezentat în figura 23. Numerotate sunt entitățile important din cadrul diagramei! Actorii (5) pot fi oameni sau sisteme automate reprezentate în diagramă în formă de omuleț cu un nume scris obilagtoriu cu majusculă. Grafic, un caz de utilizare se reprezintă printr-o elipsă. Fiecare caz de utilizare trebuie să aibă un nume distinct de al celorlalte cazuri de utilizare. Numele este format dintr-un şir de caractere și reprezintă o acțiune.

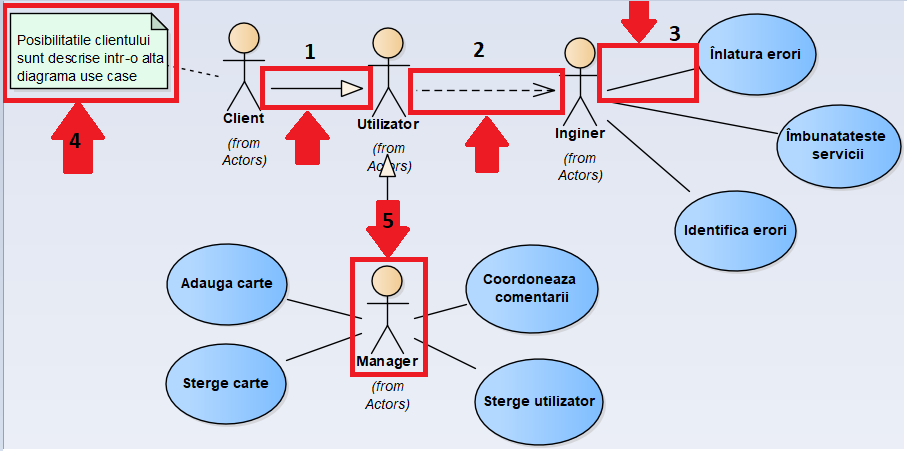


Figura 23 – Use case

Relațiile reprezentate în figura 23 care pot conecta entitățile în diagrama use case sunt asocierea (3) în formă de linie dreaptă și care poate fi unidirecțională cand la un capăt este reprezentată săgeata și bidirecțională când nu are săgeți, dependența (2) care este o linie întreruptă cu săgeată în margine, indică de cine sau ce depinde entitatea X și generalizarea sau moștenirea (1) care este o linie continuă cu un triunghi colorat în vârf, indică faptul că o entitate este părinte iar altele două sau mai multe entități o pot înlocui. Notița (4) este un simplu simbol care reprezintă comentariile sau restricțiile.

**Important!**

1. Asociere dintre doi actori sau două cazuri de utilizare nu există.
2. Pentru ca să existe relație de asociere dintre doi actori trebuie sa fie un use case la mijloc.
3. Pentru descrierea funcționalității unui use case prin altele se utilizează relația de dependență cu steriotipurile ”include” care presupune că e un pas următor obligatoriu în cadrul procesului și ”extend” care presupune că pasul poate fi executat dar și poate să nu fie.
4. Relația de generalizare poate avea loc și între actori și între use case-uri fără restricții.

**Diagrama de secvențe** este una din cele 2 diagrame care fac parte din diagrama de interacțiune și este reprezentată în figura 24. Ea transformă evenimentele identificate în scenariile cazurilor de utilizare într-o reprezentare grafică a utilizărilor sistemelor de către actor. Diagrama de secvență descrie cronologic interacțiunea obiectelor, identificînd mesajele schimbate între obiecte ca răspuns la un eveniment, împreună cu secvența mesajelor.

Numerotat cu (1) este obiectul, entitate înscrisă într-un dreptunghi și a cărei nume are forma: Nume\_obiect:Tip\_clasă::Tip\_pachet. Linia de viață (2) este reprezentată grafic printr-o linie întreruptă verticală care indică faptul că obiectul există și interacționează. În momentul în care obiectul devine activ, pe linia de viață apare un dreptunghi vertical subțire care este numit ”focus control” (3) și el este într-atât de lung, cât are loc acțiunea în care participă obiectul dat.

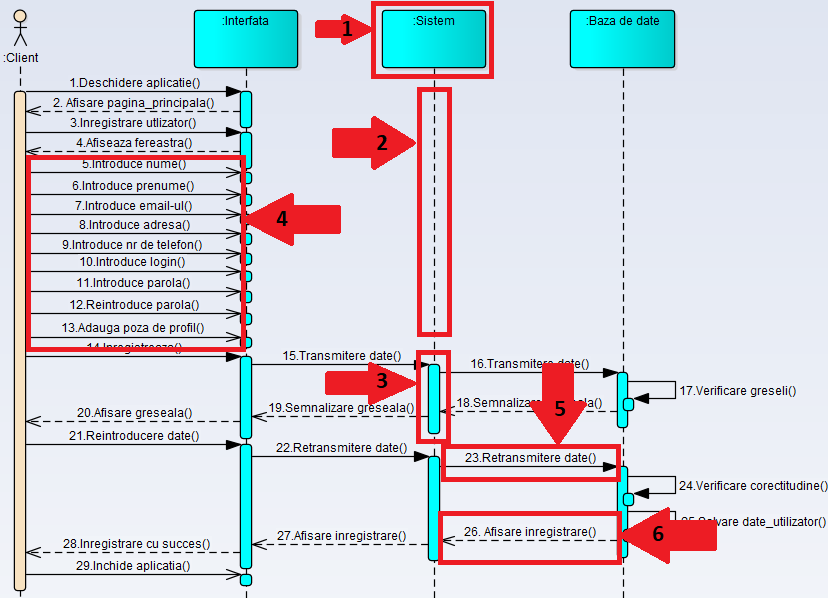


Figura 24 – Diagrama de secvențe

Ce ține de relațiile din cadrul acestui tip de diagramă, păi deosebim 3 tipuri și anume relația asincronă (4) care nu necesită răspuns, este o acțiune în urma căreia nu se întâmplă nimic fără continuare a proceselor, sincronă (5), care realizează acțiune propriu-zisă și de răspuns, care există numai după cererea celei sincrone.

Grafic, relația asincronă este reprezentată printr-o linie continuă cu săgeată în vârf, cea sincronă printr-o linie continuă cu un triunghi negru colorat în vârf iar cea de răspuns printr-o linie întreruptă cu săgeată în vârf.

**Diagrama de colaborare** este la fel o diagramă de interacţiuni din figura 25 care pune accentul pe organizarea structurală a obiectelor care participă la interacţiune. Diagramele de colaborare au multe asemănări cu diagramele de secvenţă, exprimând aceleaşi informaţii dar într-un alt format. Pot fi create la nivele diverse de detaliu şi în diferite stadii de dezvoltare a procesului software. Deoarece au un conţinut similar, pot fi folosite pentru generarea diagramelor de secvenţă şi viceversa.

Diferenţa semnificativă faţă de diagrama de secvenţă este aceea că diagrama de colaborare arată explicit legăturile dintre obiecte. De asemenea, la diagrama de colaborare timpul nu are o dimensiune explicită. Din acest motiv, ordinea în care sunt trimise mesajele este reprezentată prin numere de secvenţă.

Mesajele dintr-o diagramă de colaborare sunt reprezentate de un set de simboluri care sunt asemănătoare celor utilizate în diagrama de secvenţă, dar cu câteva elemente adiţionale pentru a marca secvenţierea şi recurenţa.

În diagrama de colaborare la fel sunt utilizate relațiile descrise în diagrama de secvențe și anume cea sincronă (1), asincronă și de răspuns (3).

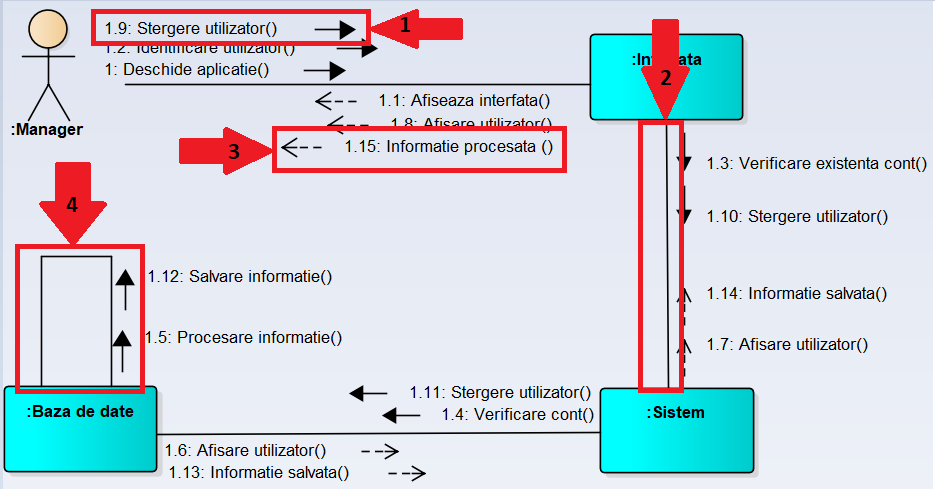
****

Figura 25 – Diagrama de colaborare-nivel de exemple

Relația de asociere (2) dintre obiect/obiect sau actor/obiect demonstrează legătura dintre ele și posibilitatea de a menționa ce acțiuni pot avea loc. Bucla (4) este utilizată atunci când e necesară procesarea și salvarea informației. Informația din call-ul care vine, intră în buclă, are loc acțiunea și vine un sigur mesaj de răspuns.

Diagrama de colaborare este în 2 nivele și anume nivel de exemple și de specificare. În figura 25 este reprezentat nivelul de exemple cu explicațiile de rigoare. Nivelul de specificare este reprezentat în figura 26.

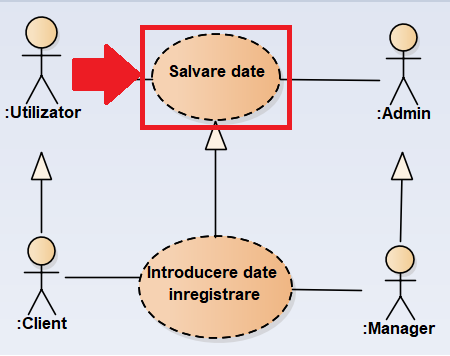


Figura 26 – Nivel de specificare

Aici apare noțiunea de colaborare, reprezentată grafic printr-un oval cu liniile întrerupte. Numele trebuie să reprezinte o acțiune. Colaborarea este o interacțiune ce reprezintă o totalitate de roluri cu efect corporativ.

**Diagrama de clase** UML modelează componentele (entităţile) de interes ale unui sistem și este reprezentată în figura 27. Clasa are instanţe, sau realizări. Aceste instanţe sunt obiectele clasei. Prin conceptul de clasă se descriu structura şi comportarea obiectelor clasei. Structura conţine atributele fiecărui obiect din clasă.

Clasele dunt entități de strucutură de bază al limbajului UML. Sunt reprezentate prin pătrate, delimitate în 3 părți. Clasa este alcătuită din numele, atributele și metodele sale.

Atributele și metodele au în față nivele de acces și anume +, -, # care se descifrează ca public, privat și protected în ordinea scrisă a simbourilor.

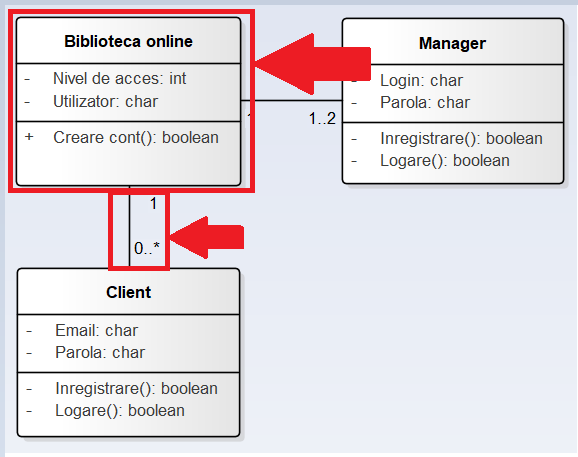


Figura 27 – Digrama de clase

Conform figurii 27 se observă un detaliu important și anume multiplicitatea. Notaţia pentru multiplicitate în cazul asocierilor UML este:

* 0 ... \* Se referă la zero sau mai multe obiecte;
* 0 ... 1 Se referă la niciun obiect sau la cel mult un obiect;
* 1 ... \* Se referă la cel puţin un obiect;
* 1, sau 1 ... 1 Se referă la exact un obiect;

Relațiile posibile dintre clase sunt cea de asociere, ca și în figura 27, dar și cazul paricular al său, agregarea, iar caz particular ar agregării fiind compoziția. De asemenea dependența și generalizarea.

Relații care nu au fost descrise până acum sunt agregarea care este reprezentată grafic printr-o linie continuă cu un romb la vârf necolorat, iar compoziția se diferențiază grafic prin faptul că acest romb este colorat în negru.

**Diagramele de stare** sunt grafuri, în care nodurile sunt stări și arcele direcționate sunt tranziții, avînd ca etichete numele evenimentelor care le-au provocat. Se precizează acțiunile rezultate din aceste schimbări.

Diagramele de stare sunt create numai pentru clasele cu un comportament dinamic semnificativ,care se pot determine prin studierea diagramelor de interacțiune, determinate la rîndul lor din descrierea diagramelor cazurilor de utilizare. Astfel, comportamentul dinamic al sistemului este caracterizat printr-o mulțime de obiecte concurente, fiecare cu propria diagramă de stare.

În figura 28 este reprezentată o diagramă de stare. Prin intermediul pătratelor roșii sunt indicate particularități importante ale acesteia.

**Important!**

1. Fiecare diagramă de stare începe cu un punct inițial (1) colorat, negru.
2. O stare (2) este reprezentată grafic printr-un dreptunghi cu margini rotungite. Numele sătrii trebuie să fie obligatoriu o acțiune.
3. Stările sunt conectate prin săgeți (3) numite tranziții.
4. Condiția gardă (4) apare atunci când dintr-o stare ies mai multe tranziții. Aceste tranziții sunt acțiuni care se scriu în paranteze pătrate “[ ]”. Condițiile date presupun faptul că numai una din ele se va efectua.
5. În secțiunea ”Properties” pe stânda, butonul ”guard” conține câmpul pentru condiția propriu-zisă.
6. Punctul final (5) este punctul de stopare al procesului și este reprezentat printr-un punct verde colorat cu unul negru în mijlocul acestuia.

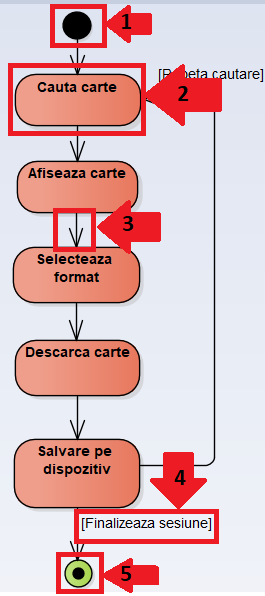


Figura 28 – Diagrama de stare

**Diagramele de activitate** sunt cazuri particulare ale diagramelor de stare care nu descriu însă un flux de control bazat pe evenimente, ci unul procedural, în care toate sau majoriatea tranzițiilor se efectuează automat, la terminarea acțiunilor efectuate în interiorul stărilor. Diagramele de activități se pot reprezenta pentru a descrie un caz de utilizare, un pachet sau implementarea unei operații. În figura 29 este reprezentat un exemplu de diagramă de activitate.

**Important!**

1. O activitate (1) este reprezentată grafic ca și o stare, aceasta fiind un dreptunghi cu margini rontugite și cu numele fiind o acțiune.
2. Fork (2) are o tranziție de intrare și mai multe de ieșire.
3. Join (3) are mai multe tranziții de intrare și numai una de ieșire.
4. Fork și join sunt utilizate împreună din caiză că acestea presupun faptul că anumite acțiuni au loc paralel.
5. Blocul de decizie (4) este reprezentat printr-un romb colorat care reprezintă un stop în diagramă. Prin intermediul său este luată o decizie, prin care tranziție va continu procesul în dependență de condițiile care trebuie respectate.
6. La punctul (5) este indicată acțiunea ce are loc prin o anumită tranziție. Este o simplă informație care nu este obligatorie.
7. Activitatea din cadrul diagramei de activitate poate avea numai o singură tranziție de ieșire și numai una de intrare. De aceea numai șa acest tip de diagramă utilizăm blocurile de decizie (4).

Despre condițiile de gardă, punctul inițial și cel final a fost scris mai sus la diagrama de stare. Aceste elemente le observăm și aici dar au aceeași însemnătate.

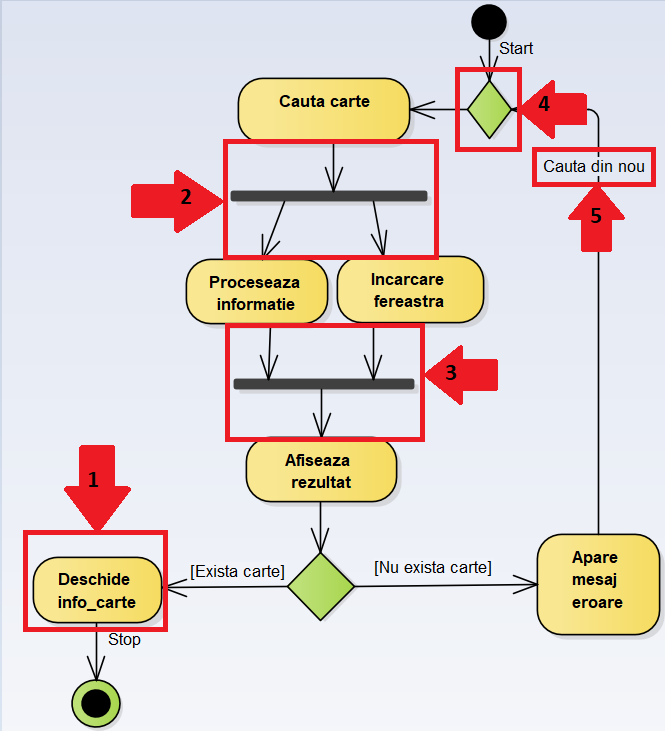
****

Figura 29 – Diagrama de activitate

**Diagrama de componente** modelează sistemul software real în mediul de implementare, evidențiază probleme de configurare prin relațiile de dependență, reprezintă o imagine a sistemului existent, înainte de a fi modificat, pot evidenția probleme de implementare fără a fi necesar să se citească tot codul sursă.

Diagrama de componente este o diagramă utilă folosită de arhitectul de sistem în fazele iniţiale de proiectare a sistemului pentru a evidenţia principalele componente software ale acestuia şi modul în care acestea colaborează.

Astfel, o componentă poate fi: cod binar, document, fișier conținând cod sursă sau date, tabelă a unei baze de date. O componentă binară este o parte fizică și substituibilă a unui sistem, care realizează și este în conformitate cu un set de interfețe.

În figura 30 este reprezentată un exemplu al diagramei de componente și evidențiate părțile importante ale acesteia, la ce trebuie atras atenția.

**Important!**

1. O componentă (1) este reprezentată grafic printr-un dreptunghi cu margini rotungite și un desen în colțul drept.
2. Între componente putem avea relația de dependență care este o relație semantică dintre două entități unde modificarea unei dintre ele – independente, poate provoca modificarea celei de a doua – dependente. Despre relația de dependență s-a vorbit la începutul capitolului!
3. A doua relație posibilă este cea de asamblare care este relația dintre 2 părți care definește faptul ca una, sau mai multe părți furnizează serviciile pe care alte părți le utilizează. Această relație este reprpezentată printr-o linie continuă pe mijlocul căreia avem un cerc alături de o semilună. Poziția semilunii indică care componentă utilizează datele furnizate de alta.

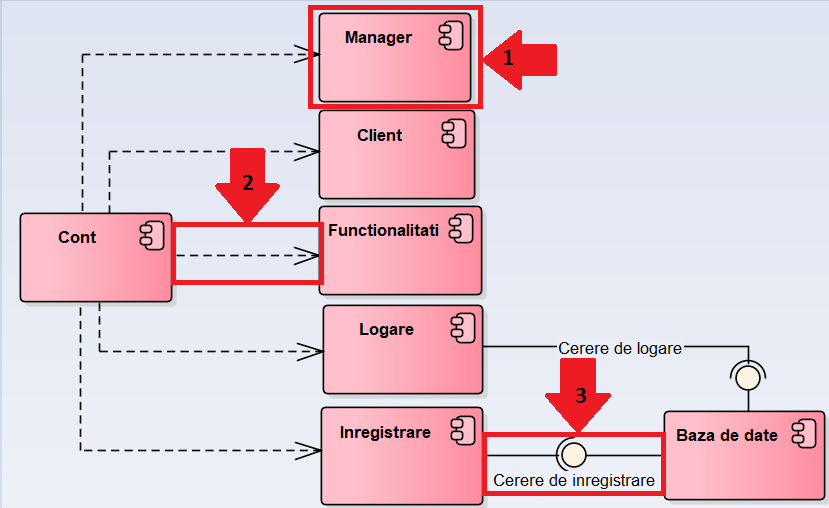


Figura 30 – Diagrama de componente

**Diagrama de plasare** modelează echipamentele mediului de implementare. Nodul este o resursă de calcul pe care artefactele sunt implementate pentru executare. Un nod este un lucru fizic care poate executa unul sau mai multe artefacte. Nodul este un element esențial UML care descrie execuția codului și comunicarea între diverse entități ale unui sistem.

Nodurile sunt abstracții ale echipamentelor, asa cum clasele sunt abstracții ale obiectelor. Fiecare nod reprezintă un tip de echipament, nu un echipament particular. Figura 31 descrie o diagramă de plasare și elementele acesteia.

**Important!**

1. Un nod se reprezintă printr-un cub în care este înscris numele său.
2. Numele se scrie cu majusculă.
3. Relația dintre noduri este numai de asociere și trebuie indicat tipul legăturii (prin ce este legatura, dacă este sau nu protejată în cazul bazei de date). Pe când între componentele din cadrul lor poate fi dependența. Iar numai între componente și interfață este relație de realizare.
4. Artefactele (4).

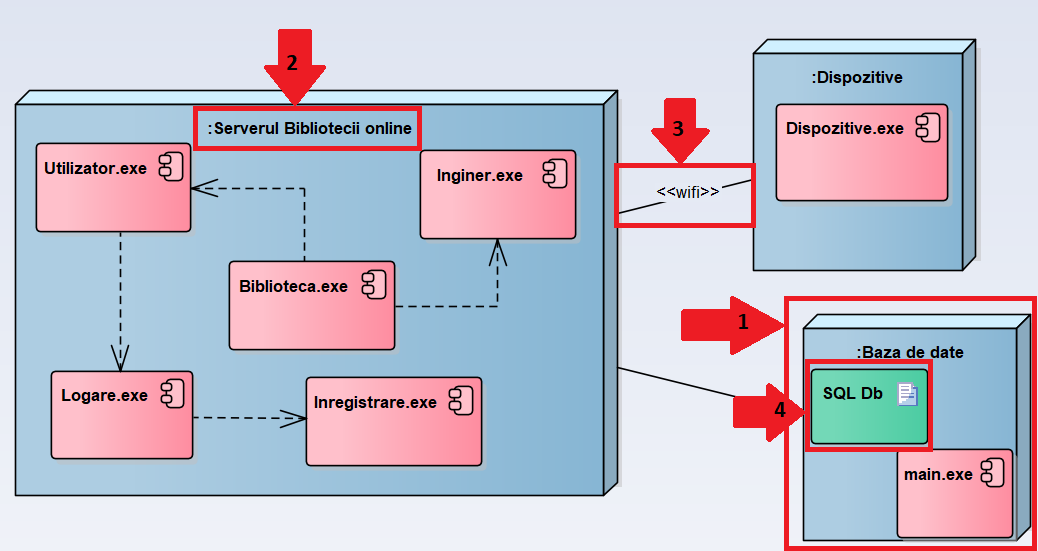


Figura 31 – Diagrama de plasare

# **Bibliografie:**

1. Rational Rose [resursă electronică] – Regim de acces:

<https://www.kpms.ru/Automatization/Rational_Rose.htm>

1. Microsoft visio [resursă electronică] – Regim de acces:

<https://en.wikipedia.org/wiki/Microsoft_Visio>

1. ArgoUML [resursă electronică] – Regim de acces: <https://ru.wikipedia.org/wiki/ArgoUML>
2. StarUML [resursă electronică] – Regim de acces: <https://en.wikipedia.org/wiki/StarUML>
3. PowerDesigner [resursă electronică] – Regim de acces:

<https://en.wikipedia.org/wiki/PowerDesigner>

1. Diagrama use case [resursă electronică] – Regim de acces:

<http://cadredidactice.ub.ro/sorinpopa/files/2018/10/L1_diagrame_use_case.pdf>

1. Diagrama de secvențe [resursă electronică] – Regim de acces:

<https://sites.google.com/site/uml4students/diagramele-de-secventa>

1. Diagrama de colaborare [resursă electronică] – Regim de acces:

<https://sites.google.com/site/uml4students/diagrama-de-colaborare>

1. Diagrama de activitate [resursă electronică] – Regim de acces:

<https://sites.google.com/site/uml4students/diagrama-de-activitate>

1. Diagrama de stare [resursă electronică] – Regim de acces:

<https://sites.google.com/site/uml4students/diagrama-de-stare>

1. Diagrama de componente [resursă electronică] – Regim de acces:

<https://support.microsoft.com/ro-ro/office/crearea-unei-diagrame-de-componente-uml-aa924ecb-e4d2-4172-976e-a78fa157b074?ui=ro-ro&rs=ro-ro&ad=ro>