Inginerie Software 2024
Laboratorul 7
Modelarea domeniului.*

Mădălina Erașcu, Alexandru Munteanu, Cristian Pal, Ionica Puiu

March 2024

1 Modelarea domeniului

Modelarea domeniului crează fundamentul componentei statice a modelului UML final al software-lui.

Când construim un model de domeniu începem cu încercări de a identifica abstractizările din lumea reală —adică principalele obiecte conceptuale care vor participa la sistem. Software-ul orientat obiect va fi organizat în jurul acestor obiecte din lumea reală, din spaţiul problemei. Aceasta se bazează pe teoria că lumea reală se schimbă mai rar decât cerinţele software.

Modelul domeniului este un model al acestor abstractizări din domeniul problemei.

Remarcă. *Modelul domeniului serveşte ca glosar de termeni folosiţi la scrierea detaliilor cazurilor de utilizare încă din primele etape ale acestui efort.*

Aşa cum identificăm obiectele din spaţiul problemei ca obiecte din lumea reală, va trebui să identificăm şi relaţiile dintre acestea. În această etapă sunt stabilite două tipuri importante de relaţii: generalizarea, care este relaţia de tip superclasă/subclasă, şi agregarea, care este o relaţie de tip parte/întreg. Există şi alte tipuri de relaţii între clase, dar *generalizarea şi agregarea sunt în mod special importante în modelarea domeniului.*

Pentru a reprezenta modelul domeniului se foloseşte (în general) diagrama de clase.

Clasele UML oferă un loc pentru capturarea atributelor (date) şi operaţiilor (funcţiile realizate de un obiect dat). Totuşi, în activitatea iniţială de modelare a domeniului nu se consumă prea mult timp cu capturarea atributelor şi operaţiilor. Acestea vor fi completate pe măsură ce se avansează în procesul de proiectare. Procesul de modelare a domeniului este centrat pe identificarea obiectelor şi a relaţiilor dintre acestea.

*Bazat pe resursele de laborator ale Conf. Dr. Cristina Mândruţă

Reutilizarea este unul din obiectivele importante ale construirii software-lui în jurul acestor abstractizări de entităţi din lumea reală, deoarece există sisteme software multiple care au în comun acelaşi domeniu al problemei. O modelare bună a domeniului problemei va revela aspectele reutilizable ale software-lui.

2 Metodologie

2.1 Etapele principale în dezvoltarea modelului domeniului

1. Identificarea conceptelor domeniului din descrierea problemei şi detaliile cazurilor de utilizare.
2. Promovarea conceptelor domeniului ca clase în modelul domeniului.
3. Identificarea şi analiza proprietăţilor semnificative la nivelul aplicaţiei, pentru obiectele fiecărei clase identificate anterior.
4. Identificarea conexiunilor conceptuale între concepte, pe baza descrierii problemei şi a detaliilor cazurilor de utilizare.
5. Promovarea conexiunilor dintre concepte ca asocieri între clasele corespunzătoare în modelul domeniului. Rafinarea lor prin identificarea relaţiilor de agregare.

2.2 Ghid practic

Cele mai bune surse pentru clasele domeniului sunt:

- descrierea generală a problemei
- cerinţele de detaliu
- cunoştinţele experţilor în spaţiul problemei

Primul pas:

- scrieţi cât mai multe propoziţii relevante din aceste surse
- încercuiţi/evidenţiaţi toate substantivele şi construcţiile substantivale.

În acest mod există şanse să găsiţi marea majoritate a obiectelor (claselor) domeniului. Ulterior, în cursul unui proces de rafinare:

- substantivele şi construcţiile substantivale devin obiecte şi atribute
- verbele şi construcţiile verbale devin operaţii şi asocieri
- frazele posesive indică faptul că substantivele trebuie să fie atribute, nu obiecte

Al doilea pas este trecerea în revistă a claselor candidat şi eliminarea elementelor nenecesare (redundante sau nerelevante) sau incorecte (prea vagi, concepte din afara zonei modelului, acţiuni exprimate prin substantive).

Pasul al treilea este construirea diagramei de clase.

În cursul construirii diagramei de clase se pot lua primele decizii referitoare la:

- relaţiile de generalizare dintre clase (chiar şi pe mai multe nivele)
- agregările de clase

În final, foarte asemănător cu o diagramă entitate-relaţie (ERD), modelul domeniului, actualizat pentru a arăta asocierile—relaţiile statice dintre clase pereche—trebuie să constituie o reprezentare corectă a spaţiului problemei, independentă de timp (adică statică).

3 Indicații

Diagrama claselor de domeniu

1. Evitaţi stabilirea multiplicităţilor în această etapă.
2. Nu faceţi o analiză prea exhaustivă a substantivelor şi verbelor.
3. Evitaţi asignarea de operaţii la clase. În această etapă a unui proiect nu există încă suficiente informaţii pentru a lua decizii de proiectare bune referitoare la operaţii. Aceste informaţii vor fi obţinute după modelarea interacţiunilor.
4. Parcurgeţi repede modelarea domeniului pentru a afla cât mai curând dacă aţi modelat ceea ce doresc clienţii.
5. Utilizaţi doar relaţia de agregare în cursul modelării domeniului. Diferenţa dintre agregare şi compoziţie sa va face ulterior, la proiectarea de detaliu.
6. Nu introduceţi elemente legate de tehnologii specifice (ex. baze de date relaţionale, un anume tip de server, o anumită platformă de execuţie). Acestea sunt probleme de implementare.
7. Modelul domeniului urmăreşte şi stabilirea unui acord al membrilor echipei de dezvoltare asupra numirii abstractizărilor cheie. De aceea numele claselor trebuie să fie cât mai sugestive. Nu se recomandă aici folosirea acronimelor.
8. Evitaţi referirea la construcţii de implementare. Acestea sunt relevante în spaţiul soluţiei, pe când modelul domeniului se referă la spaţiul problemei.

9. Dacă realizaţi reingineria unui sistem legacy care utilizează o bază de date relaţională, tabelele din baza de date sunt o sursă excelentă de nume pentru clasele domeniului. Totuşi nu toate pot fi utilizate în contextul unui model obiect.

10. În această etapă nu se gâdeşte încă în termeni de şabloane (patterns).

Exemplu (Librărie pe Internet)**.** *Lista cerinţelor pentru o librărie accesibilă pe Internet:*

- *Librăria va accepta ordine prin Internet.*
- *Librăria va menţine o listă de conturi pentru max. 1,000,000 clienţi.*
- *Librăria va oferi protecţie pe bază de parolă pentru fiecare cont.*
- *Librăria va oferi posibilitatea de a căuta într-un catalog central de cărţi.*
- *Librăria va oferi mai multe metode de căutare în catalog, incluzând căutare după autor, după titlu, după ISBN, şi după cuvânt cheie.*
- *Librăria va oferi un mijloc securizat de realizare a plăţilor cu card de credit de către clienţi.*
- *Librăria va oferi un mijloc securizat de plată via ordin de plată.*
- *Librăria va oferi link-uri electronice între Web şi baza de date şi sistemul de livrare.*
- *Librăria va oferi link-uri electronice între Web şi baza de date şi sistemul de gestiune a stocurilor.*
- *Librăria va păstra recenzii ale cărţilor şi va permite oricui să încarce recenzii pe site.*
- *Librăria va menţine clasamente ale cărţilor, bazate pe intrările de la clienţi.*

Următoarele exemple, luate din modelul domeniului pentru aplicaţia de librărie accesibilă pe Internet, au rolul de a ilustra o serie de greşeli tipice ce se fac în timpul modelării domeniului.

Explicaţii pentru exemplele din Figurile 1 și 2.

- Clasa **cBinaryTree** este o clasă parametrizată (clasă template). Aceasta defineşte o construcţie de implementare (arbore binar) care nu trebuie exprimată la acest stadiu al modelării.
- Clasa **cLoginMgr** are o operaţie numită `verificaParola`. Este prea devereme pentru decizii despre ce operaţii vor fi realizate de către care clase.
- Numele **cLoginMgr** nu este suficient de intuitiv.

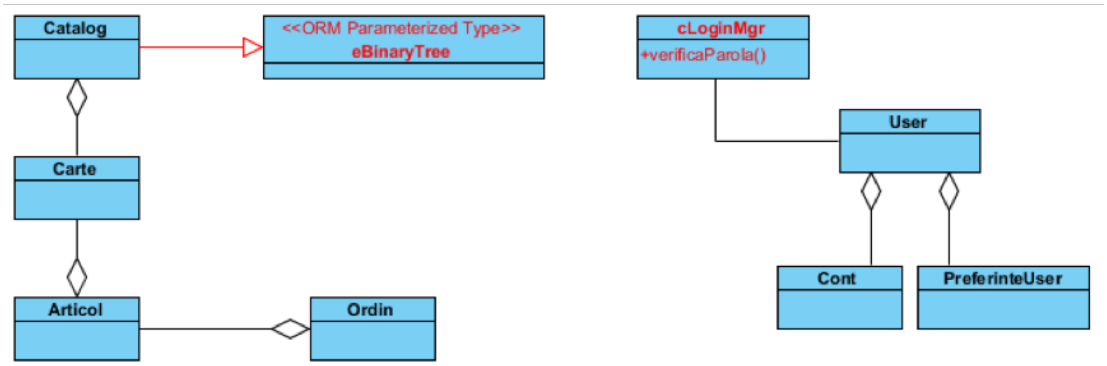

Figure 1: Exemplul 1 – greşeli

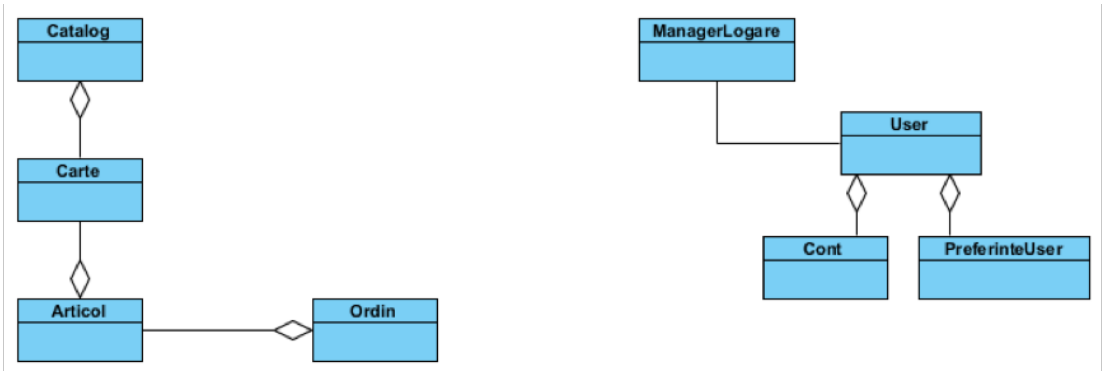

Figure 2: Exemplul 1 - corectare

Explicaţii pentru exemplele din Figurile 3 și 4.

- Numele clasei **cSessionBeanShpngCart** indică faptul că modelatorul a decis să reprezinte conceptul "cos de cumparaturi" utilizînd o componentă de tip EJB (Enterprise Java Bean). Explorarea modului în care clasele vor fi puse în corespondenţă cu tipuri de componente se face într-o etapă ulterioară modelării domeniului, în cursul procesului de proiectare de detaliu.
- Cel mai sugestiv nume pentru o clasă ce reprezintă un cos de cumparaturi este **CosCumparaturi**.

Explicaţii pentru exemplele din Figurile 5 și 6.

- Prezenţa atributului **foreignInventoryDBKey** indică faptul că modelatorul are în vedere folosirea unei baze de date relaţională. Mecanismul utilizat pentru persistenţa datelor va fi stabilit ulterior, în cursul proiectării.

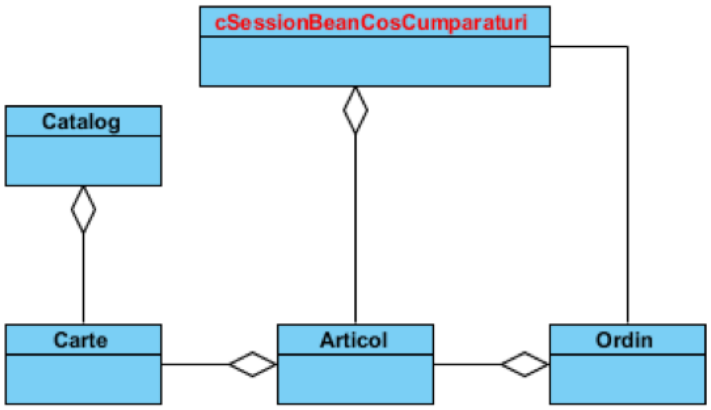

Figure 3: Exemplul 2 – greşeli

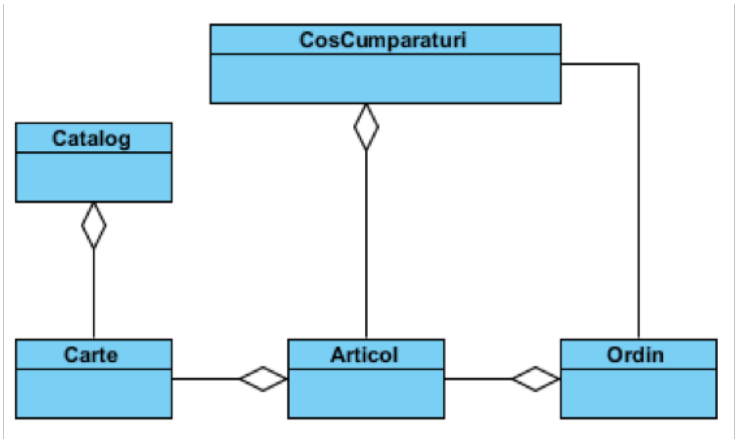

Figure 4: Exemplul 2 - corectare

- Clasa **Ordin** are asignate atribute şi operaţii. Clasele din modelul domeniului ar putea să conţină unele atribute dar în mod sigur nu trebuie să conţină operaţii.
- Asocierea dintre **Cont** şi **InfoFacturare** are multiplicitate. Multiplicitatea nu trebuie încă precizată.

Explicaţii pentru exemplele din Figurile 7 și 8.

- Prezenţa atributelor **pret**, **cantitate** şi **editor**, aparţinând probabil unor clase asociate, indică faptul că modelatorul a pus clasa Ordin în corespondenţă directă cu o tabelă Ordin existentă.

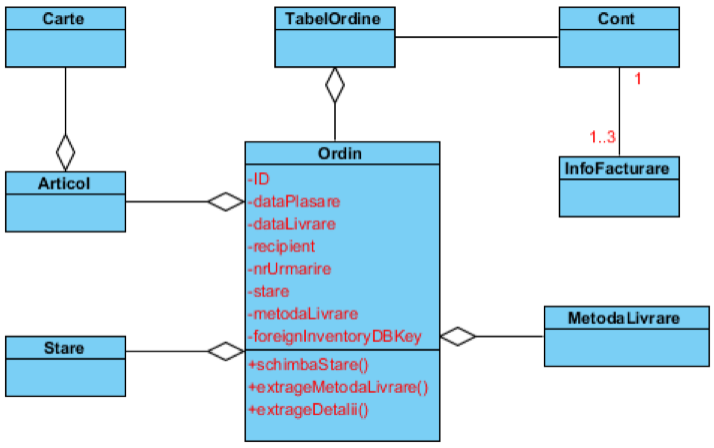

Figure 5: Exemplul 3 – greşeli

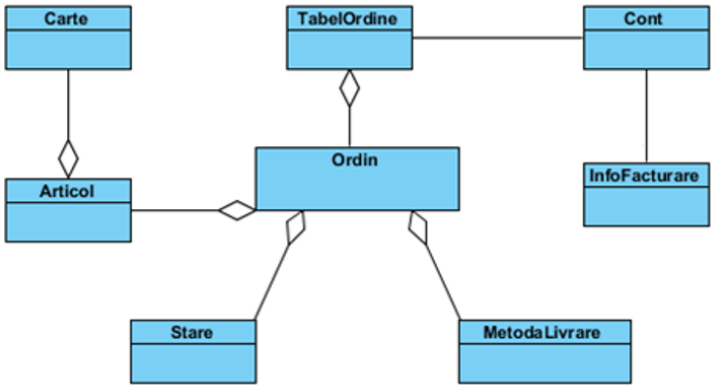

Figure 6: Exemplul 3 - corectare

- Clasa **OrdinCumparare** utilizează o construcţie Vector specifică unui limbaj particular (Java).
- Modelatorul a ales să utilizeze şablonul de proiectare Proxy; alegerea şabloanelor de proiectare nu este obiectiv al modelării domeniului...

Explicaţii pentru exemplele din Figurile 9 și 10.

- Asocierea dintre **Articol** şi **CosCumparaturi** este o compoziţie, dar este prea devreme să putem decide între compoziţie şi agregare.
- Stereotipurile claselor **Ordin** şi **OrdinCandidat** indică o decizie prematură de alocare a acestor clase la straturile arhitecturii multinivel a aplicaţiei.

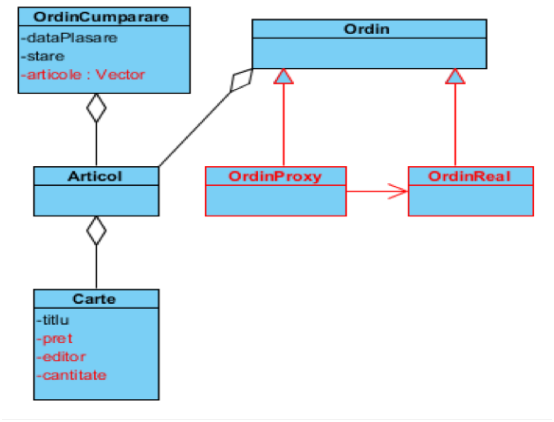

Figure 7: Exemplul 4 – greşeli

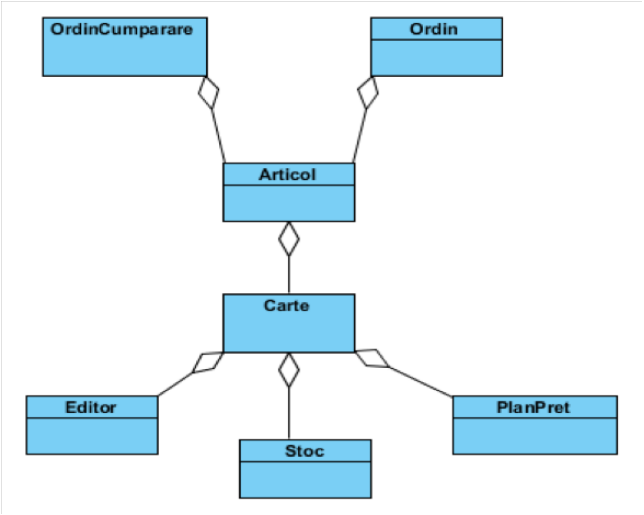

Figure 8: Exemplul 4 - corectare

Exemplu (Librărie pe Internet: Modelul complet al domeniului)**.** *Figura 11 conţine modelul complet al domeniului pentru aplicaţia Librarie pe Internet introdusă mai sus. Diagrama consolidează fragmentele prezentate în exemplele anterioare şi adaugă clase şi asocieri ce vor fi necesare în următoarele etape de modelare.*

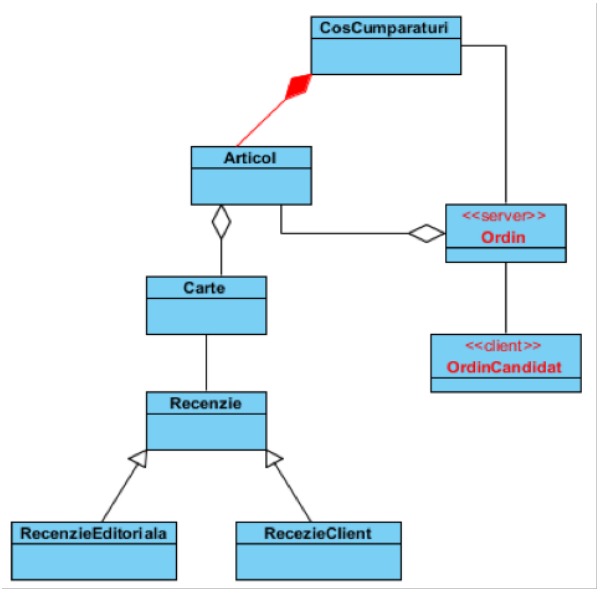

Figure 9: Exemplul 5 – greşeli

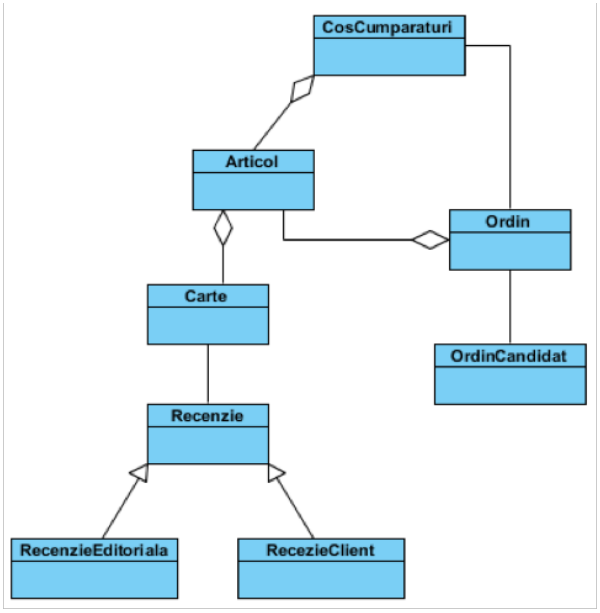

Figure 10: Exemplul 5 - corectare

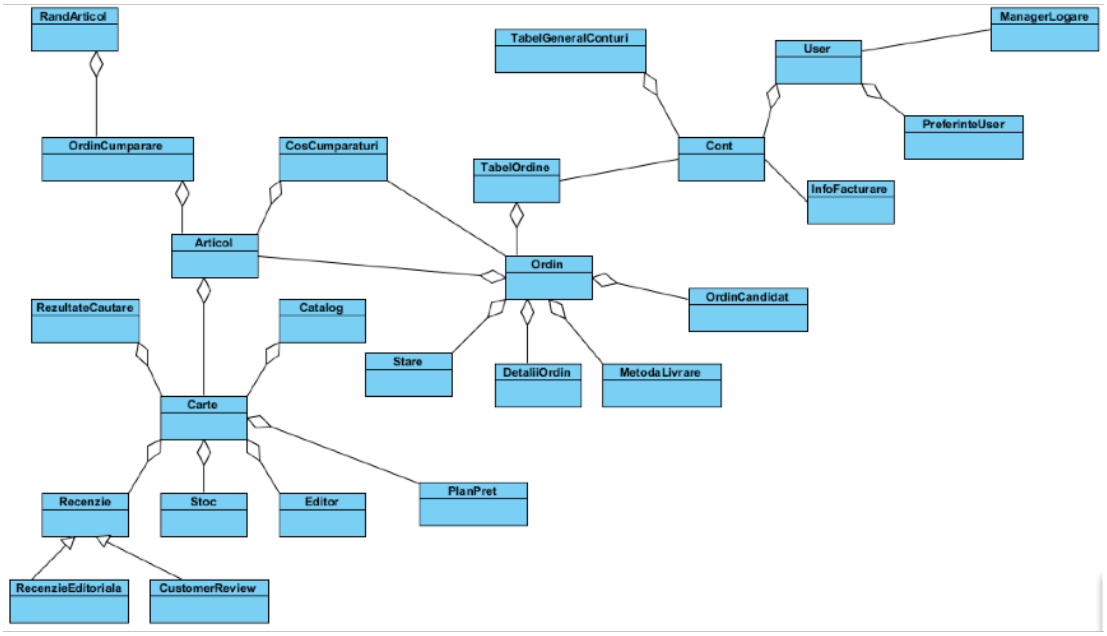

Figure 11: Modelul domeniului pentru Librărie pe Internet

4 Temă

1. Studiaţi la `https://www.visual-paradigm.com/support/documents/vpuserguide/94/1288_textualanaly.html` Part II. UML modeling Cap. 15 din Visual Paradigm Users Guide Textual Analysis - Facilitatea oferită de VP pentru analiza textuală.

2. Creaţi diagrama claselor de domeniu pe baza următoarelor specificaţii de cerinţe. Urmaţi etapele indicate în exerciţiu.

   Dezvoltăm un sistem cu care profesorii pot înregistra şi actualiza notele studenţilor. Profesorii trebuie să poată distribui rapoarte cu notele. Iată lista completă a cerinţelor sistemului:

   - Un profesor poate înregistra note. De câte ori sunt înregistrate note, acestea sunt salvate pe disc.
   - Un profesor poate actualiza note. De câte ori sunt actualizate note, nota existentă este încărcată. După modificare, noua notă este salvată pe disc.
   - Profesorul, secretara şi studentul pot vizualiza note.
   - Pentru a vizualiza note, solicitantul trebuie să fie autentificat.
   - Un student cu taxă este un tip de student.
   - O secretară poate genera rapoarte cu notele.

- Un profesor poate distribui rapoarte cu notele.

2.1. Identificaţi conceptele (i.e. acordaţi-le nume).

2.2. Promovaţi conceptele domeniului ca clase în diagrama de clase.

2.3. Există concepte care sunt specializări ale altor concepte mai generale? Dacă da, identificaţi care este conceptul generalizare şi care este conceptul specializare. Ce tip de relaţie trebuie să existe între conceptul generalizat şi conceptu specializat?

2.4. Există concepte ce pot fi grupate în colecţii de acelaşi tip? Dacă da, ce tip de relaţie trebuie folosit? Identificaţi toate conceptele care pot fi grupate în colecţii şi adăugaţi fiecare colecţie ca o nouă clasă.

2.5. Identificaţi conexiunile conceptuale dintre concepte.

2.6. Promovaţi conexiunile dintre concepte ca asocieri între clasele corespunzătoare din modelul domeniului.

3. Desenaţi diagrama claselor de domeniu pentru următoarea aplicaţie.

   - O bibliotecă are cărţi şi reviste. Trebuie dezvoltată o aplicaţie software pentru împrumut cărţi. Pentru a împrumuta o carte, clientul trebuie să fie membru al bibliotecii. Există o limită a numărului de cărţi ce pot fi simultan împrumutate de un membru al bibliotecii.
   - Biblioteca poate deţine mai multe exemplare ale unei anumite cărţi.
   - O carte se poate rezerva.
   - Unele cărţi pot fi împrumutate doar pe termen scurt. Alte cărţi pot fi împrumutate pentru 3 săptămâni. Utilizatorii pot extinde împrumuturile.

4. Creaţi modelul domeniului pentru *Sistem închiriere maşini*.