**Introducerea în UML**

Pentru facilitarea dezvoltării sistemelor informatice și a software-ului s-a urmărit elaborarea de metodologii pentru analiza și proiectare, ca și de instrumente suport.

OMT este o metodologie orientata-obiect, apărută mult înaintea limbajului UML, ce va fi prezentat ulterior, din care acesta din urma a preluat elemente de reprezentare. Pâna la faza de implementare, metodologia OMT prevede etapele de:

**■ analiza;**

**■ proiectare de sistem;**

**■ proiectare obiectuala** (conține toate aspectele esențiale ale domeniului aplicatiei: obiectele din domeniul aplicatiei, inclusiv o descriere a proprietatilor obiectelor și a manifestarii lor foarte utile fiind diagramele obiectuale, fără a se ține cont de modul de implementare)

Metodologia OMT combină trei tipuri de modele.

■ Modelul obiectual reprezintă aspectele statice, structural, legate de “datele” unui sistem.

■ Modelul dinamic reprezintă aspectele temporale, comportamentale, de “control” ale unui sistem.

■ Modelul funcțional reprezintă aspectele transformaționale, legate de “funcțiile” unui sistem.

În general, se acceptă ca o tehnică de analiză și proiectare orientată-obiect trebuie să includă trei componente (cunoscute ca "triunghiul succesului”):

■ notația;

■ procesul;

■ instrumentul CASE.

Notația este componenta principală, constând în limbajul utilizat atât pentru modelare cât și pentru proiectare. Datorită importantei notației, s-a depus un mare efort pentru ajungerea la o acceptare universală a unei notații standard.

Rezultatul efortului de standardizare îl reprezintă limbajul UML, acesta fiind acceptat ca limbaj de modelare standard de către OMG (Object Management Group).

Evident, sistemele complexe au structuri complexe și manifestă un comportament complex. Pentru a cuprinde complexitatea, UML permite modelarea diferitelor aspecte ale unui sistem, în funcție de:

■ structura statică;

■ comportamentul dinamic;

■ relațiile dintre diferitele componente etc.

UML reflectă complexitatea modurilor de abordare prin prezența unei mulțimi de diagrame diferite pentru a reprezenta aspecte variate ale sistemului ce este modelat, oferind proiectantului un sprijin deosebit pe întreaga durat a procesului de modelare.

UML este acronimul pentru "Unified Modeling Language" (Limba Unificată de Modelare), o limbă standardizată utilizată în ingineria software pentru a descrie și documenta designul unui sistem software. UML furnizează o serie de diagrame grafice și notări semantice care permit dezvoltatorilor să modeleze diferite aspecte ale unui sistem software, precum structura, comportamentul și interacțiunile între componente.

**Istoria UML:**

* Începuturile UML-ului pot fi urmărite în anii '80 și '90, când obiectele și programareaorientată pe obiecte au devenit tot mai populare în comunitatea dezvoltatorilor software.
* În 1994, Grady Booch și Jim Rumbaugh au început să colaboreze la dezvoltarea unei limbide modelare orientate pe obiecte, care ulterior a devenit OMT (Object Modeling Technique) și Booch Method.
* În 1995, Rational Software Corporation, condusă de Grady Booch, a achiziționat OMT și aînceput să integreze tehnologiile de modelare într-un singur instrument.
* În același timp, Ivar Jacobson a contribuit cu lucrările sale asupra metodologiei de modelarea comportamentului, cunoscută sub numele de Use Case.
* În 1997, Rational a publicat prima versiune a limbajului UML, combinând contribuțiile luiBooch, Rumbaugh și Jacobson, cu aporturi suplimentare de la alți experți din domeniu.
* În 2005, UML a devenit un standard ISO (ISO/IEC 19501:2005).

### Diagramele UML

Diagrama UML (Unified Modeling Language) este un set de diagrame utilizate în ingineria software pentru a descrie modelele de design ale sistemelor software. UML este o limbaj grafic standardizat care permite dezvoltatorilor să comunice și să planifice designul software-ului întrun mod clar și precis. Diagrama UML poate fi utilizată pentru a reprezenta diverse aspecte ale sistemului software, cum ar fi comportamentul, structura, interacțiunile și procesele de afaceri.

La elaborarea proiectului de ansamblu s-a atins cerința de studiere a arhitecturii funcționale, în contextul căreia s-au conturat cele mai importante componente funcționale, împreună cu relațiile și interfețele dintre acestea. Așadar, prin alegerea temei, am pus bazele sistemelor, subsistemelor și operațiilor ce intră în alcătuirea aplicației web pentru promovare, continuând prin reprezentarea schemelor logice ale sistemului informatic, unde se regăsesc circuitele și fluxurile informaționale. În acest context, se introduce în analiză, termenul-cheie „UML”, adică „Unified Modeling Language”.

UML este un instrument modern de realizare a arhitecturilor software folosind diagrame semnificative pentru dezvoltatori: diagrama cazurilor de utilizare, diagrama claselor, diagrama stărilor, diagrama de secvență și diagrama de activitate.

Diagramele sunt reprezentări grafice ce permit ilustrarea proceselor, obiectelor, atributelor și a interacțiunilor implicate în interacțiunilor de demonstrat. Importanța lor se manifestă în mod special în etapa de proiectare a sistemelor, în contextul alegerii strategiei de realizare, împărțirii pe trepte a realizării componentelor.

## Tipuri de Diagrame UML

Limbajul UML (Unified Modeling Language) oferă un set de diagrame utilizate pentru a descrie modelele de design ale sistemelor software. Aceste diagrame sunt împărțite în două categorii principale: diagramele de structură și diagramele de comportament. (Fig1)

Diagramele de structură sunt utilizate pentru a evidenția componentele necesare în cadrul sistemului modelat și sunt utilizate în general pentru documentarea arhitecturii sistemelor software. Aceste diagrame sunt folosite pentru a reprezenta entitățile, clasele, obiectele și relațiile dintre acestea, precum și pentru a ilustra structura statică a sistemului.

Diagramele de comportament sunt utilizate pentru a evidenția ceea ce trebuie să se întâmple în cadrul sistemului modelat și pentru a ilustra comportamentul sistemului. Aceste diagrame sunt utilizate în general pentru a descrie funcționalitatea sistemului și pentru a reprezenta interacțiunile dintre componentele sale. Diagramele de comportament includ diagrame de secvență, diagrame de activitate, diagrame de stări și evenimente, precum și diagrame de interacțiune.

În total, există 14 tipuri de diagrame UML, fiecare cu un scop specific și utilizat pentru a reprezenta aspecte diferite ale sistemului software. Aceste diagrame sunt utile pentru a comunica și planifica designul software-ului într-un mod clar și precis, asigurând astfel o dezvoltare a sistemelor software mai eficientă și mai ușor de întreținut.

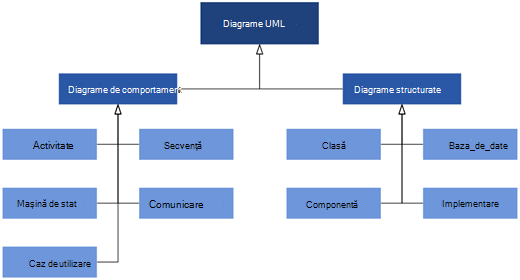


Fig. 1.Organizarea ierarhica a diagramelor UML

**Diagrama cazurilor de utilizare (Use Case Diagram)** descrie destinația funcțională a sistemului sau, cu alte cuvinte, descrie ceea ce sistemul va executa în procesul său de funcționare . În arhitectura proiectului propus, această diagramă ilustrează procesul de achiziție de către clienți si aprobarea comenzii de către administrator. Diagrama cazurilor de utilizare va face referință doar la o parte din modulele prezente în aplicația web pentru promovare, procesul de achiziție fiind cel mai important de ilustrat.

Acest tip de diagramă este compus din actori, cazuri de utilizare și relații de dependență, conturând foarte clar principiile proiectului propus.

Cazul de utilizare se notează cu o elipsă în interiorul căreia se conţine denumirea prescurtată sau numele în formă de verb cu cuvinte explicative (Fig. 2).

A close-up of a sign

Description automatically generated

Figura 2. Caz de utilizare.

Scopul cazului de utilizare constă în determinarea aspectului terminal sau fragmentului de comportare a unei entităţi fără desfăşurarea structurii interne a acestei intităţi. În calitate de aşa entitate poate fi un sistem iniţial sau un element al modelului care dispune de comportament propriu, precum este subsitemul sau clasa în modelul unui sistem.

Fiecare caz de utilizare corespunde unui serviciu aparte, care reprezintă o entitate modelată sau un sistem la cererea utilizatorului (actorului), *mai precis determină metoda aplicată către o anumită entitate.* Serviciul care este iniţializat la cererea utilizatorului reprezintă o succesiune terminată de acţiuni.

Cazurile de utilizare descriu nu numai colaborarea între utilizatori şi entităţi, dar şi reacţia entităţii la primirea anumitor mesaje de la utilizatori şi asupra percepţiei acestor mesaje în afară entităţii. Cazurile de utilizare pot include descrierea specificaţiilor modurilor de realizare a serviciului şi a diferitor situaţii excepţionale, aşa cum este prelucrarea corectă a erorilor unui sistem. Mulţimea cazurilor de utilizare în total poate determina toate aspecte posibile comportării aşteptate a unui sistem. Pentru comoditate mulţimea cazurilor de utilizare poate fi considerată ca un pachet aparte.

Din punct de vedere sistemo-analitic cazurile de utilizare pot fi folosite pentru specificarea cerinţelor externe către sistemul proiectat şi pentru specificarea comportării funcţionale a sitemului deja existent.

*Exemple de cazuri de utilizare pot fi acţiunile următoare: verificarea stării contului curent al clientului, intocmirea comenzii la procurarea mărfii, obţinerea informaţiei suplimentare despre solvabilitatea clientului, reprezentarea formei grafice la ecranul monitorului s.a.*

*Actorul r*eprezintă orice entitate externă sistemului modelat, care colaborează cu sistemul şi utilizează posibilităţile lui funcţionale pentru atingerea anumitor scopuri şi pentru rezolvarea problemelor particulare. Totodată actorii sunt utilizaţi pentru notarea mulţimii rolurilor coordonate ale utilizatorilor în procesul de colaborare cu sistemul proiectat. Fiecare actor poate fi considerat un anumit rol aparte relativ unui caz de utilizare concret. Notaţia grafică standardă a unui actor în diagramă este «omuleţul» sub care se indică numele actorului (Fig. 3).

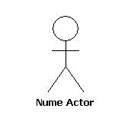


Figura 3 .- Actor

*Ca exemplu de actori pot fi: clientul unei bănci, angajatul unei bănci, vînzătorul unui magazin, managerul secţiei de vînzare, pasagerul unui avion, conductorul unui auto, administratorul unui hotel, celularul şi alte entităţi, care au legătură cu modelul conceptual care corespunde domeniului de lucru.*

*Legăturile în diagrama a cazurilor de utilizare.* Între componentele diagramei cazurilor de utilizare pot să existe diferite legături care desciu colaborarea exemplarelor unor actori şi cazurilor de utilizare cu exemplarele altor actori şi cazuri. Un anumit actor poate să colaboreze cu mai multe cazuri de utilizare. În acest caz actorul dat se adresează către cîteva servicii ale sistemului dat. La rîndul său un anumit caz de utilizare poate să colaboreze cu mai mulţi actori, pentru care el acordă serviciul său.

În limbajul UML sunt cîteva tipuri standarde de relaţii între actori şi cazuri de utilizare:

* relaţia de asociere (association relationship)
* relaţia de extindere (extend relationship)
* relaţia de generalizare (generalization relationship)
* relaţia de cuplare (include relationship)

*Relaţia de asociere.*Relaţia de asociere este o noţiune fundamentală în limbajul UML şi mai mult sau mai puţin se utilizează la crearea tuturor modelelor grafice în forma diagramelor canonice.

Cu privire la diagrama cazurilor de utilizare relaţia de asociere specifică rolul deosebit al actorului în cazul de utilizare aparte. Cu alte cuvinte, asocierea specifică particularitatea semantică de colaborare a actorilor şi cazurilor de utilizare în modelul grafic. În aşa mod această relaţie stabileşte ce rol joacă actorul la colaborarea cu exemplarul cazului de utilizare. În diagrama cazurilor de utilizare precum şi în alte diagrame relaţia de asociere se notează cu o linie neîntreruptă între actor şi cazul de utilizare. Această linie poate să aibă condiţii suplementare, de exemplu, numele şi multiplicitatea (Fig. 4).

A diagram of a customer

Description automatically generated

Figura 4.Relaţie de asociere între acotr şi caz de utilizare.

*Relaţia de extindere* defineşte interconexiunea exemplarelor cazului de utilizare cu cazul general, proprietăţile căruia sunt definite pe baza modului de uniune a exemplarelor date. În metamodelul relaţie de extindere este directă şi indică care condiţii concrete pot fi utilizate pentru unele exemple unui anumit caz de utilizare, definite pentru extinderea cazului de utilizare dat. Dacă are loc relaţie de extindere de la cazul de utilizare A la cazul de utilizare B, acest lucru înseamnă că proprietăţile exemplarului cazului de utilizare B pot fi adăugate datorită proprietăţilor extinse a cazului de utilizare A.

Relaţia de extindere între cazurile de utilizare reprezintă o linie punctată cu săgeată (cazul relaţiei de dependenţă), directă de la acel caz de utilizare, care reprezintă extinderea cazului de utilizare iniţial. Această linie cu săgeată este marcată cu cuvîntul „extend” („extinde”), Fig. 5.

A close-up of a text

Description automatically generated

Figura 5. Relaţia de extindere între cazurile de utilizare.

*Relaţia de generalizare* este folosită pentru indicarea faptului că care-va caz de utilizare A poate fi generalizat la cazul de utilizare B. În urma căruia, cazul A va fi cazul special cazului B. În urma căruia cazul B se numeşte părinte relativ A, iar cazul A – descendent relativ cazului de utilizare B.

Este nevoie de menţionat, că descendentul moşteneşte toate proprietăţile şi comportamentul părintelui său şi poate avea proprietăţile şi comportamentul său adăugător. Grafic relaţia dată este reprezentată cu linia întreagă cu săgeată în forma de triunghi nehaşurat, care indică cazul de utilizare părinte (Fig. 6). Această linie cu săgeată are un nume specific – săgeata „generalizare”.

A black and white line

Description automatically generated

Figura 6**.** Relaţia de generalizare între cazuri de utilizare.

*Relaţia de tip include* între două cazuri de utilizare indică un comportament stabilit pentru un caz de utilizare ce este inclus ca component compus în consecutivitatea comporatamentului a altui caz de utilizare.

Semantica acestei relaţii este definită în felul următor. Cînd exemplarul primului caz de utilizare în timpul executării ajunge la punctul de includere în consecutivitatea comporatamentului exemplarului al doilea a cazului de utilizare, exemplarul primului caz de utilizare execută consecutivitatea acţiunilor, care definesc comportamentul exemplarului al doilea a cazului de utilizare, după ce continuă executarea acţiunilor comportamentului său.

Cazul de utilizare ce este inclus poate fi independent de cazul de bază, anume el dă ultimului un comportament incapsulat, detalii realizaţiei căruia sunt ascunse şi pot fi liber împărţite între cîte-va cazuri de utilizare incluse.

Relaţia de tip include orientată de la cazul de utilizare A la cazul de utilizare B indică că fiecare exemplar al cazului A include proprietăţi funcţionale stabilite pentru cazul B. Aceste proprietăţi specializează comportamentul cazului respectiv A în diagrama dată.

Grafic relaţiile date sunt indicate cu linia punctată cu săgeată (cazul relaţiei de dependenţă), îndreptate de la cazul de utilizare de bază la cazul ce este inclus. În urma căruia linia cu săgeata este indicată cu cuvîntul-cheie „include”, Fig. 7).

A close-up of a sign

Description automatically generated

Fig. 7. Relaţie de tip include între cazuri de utilizare.

Diagram

Description automatically generated

Fig. 8 – Diagrama cazurilor de utilizare Client-Administrator

**Diagrama de stare** este una dintre cele mai bune modalități de reprezentare dinamică a unui sistem proiectat. Se conturează și se modelează astfel un comportament tipic proiectului reprezentat, specificând secvențele de stări prin care trece proiectul de-a lungul duratei sale de viață, ca un răspuns la evenimentele ce au loc asupra sa.

Importanța unei astfel de diagrame se ramifică în mod special în domeniul de automatizare al proceselor, unde prin intermediul programelor specifice, și cu ajutorul cunoașterii stării obiectului de studiat, se poate elimina necesitatea intervenției umane.

**Noţiuni preliminare**

Un ***eveniment*** reprezintă ceva (atomic) ce se întâmplă la un moment dat şi care are ataşată o locaţie în timp şi spaţiu. Evenimentele modelează apariţia unui stimul care poate conduce la efectuarea unei tranziţii între stări. Evenimentele nu au durată în timp.

Evenimentele pot fi clasificate în felul următor:

* sincrone sau asincrone
* externe sau interne

Evenimentele *externe* se produc între sistem şi actori (de exemplu apăsarea unui buton pentru întreruperea execuţiei programului).

Evenimentele *interne* se produc între obiectele ce alcătuiesc un sistem (de exemplu *overflow exception*).

Evenimentele pot include:

* ***semnale***; semnalul este un stimul asincron care are un nume şi care este

trimis de un obiect şi recepţionat de altul (ex: excepţii).

* ***apeluri de operaţii*** (de obicei sincrone): un obiect invoca o operaţie pe

un alt obiect; controlul este preluat de obiectul apelat, se efectuează operaţia, obiectul apelat poate trece într-o nouă stare, după care se redă controlul obiectului apelant.

* ***trecerea timpului***
* ***o schimbare a rezultatului evaluării unei condiţii***

O ***acţiune*** reprezintă execuţia atomică a unui calcul care are ca efect schimbarea stării sau returnarea unei valori. Acţiunile au o durată mică în timp, fiind tranzitorie (ex.: i++).

Prin ***activitate*** se înţelege execuţia neatomică a unor acţiuni. Activităţile au durată în timp, pot persista pe toată durata stării şi poate fi întreruptă (ex.: execuţia unei funcţii).

O diagramă de stări poate conţine *stări* şi *tranziţii*.

**Starea**

Prin *stare* se înţelege o condiţie sau situaţie din viaţa unui obiect în timpul căreia acesta:

* satisface anumite condiţii;
* efectuează o activitate;
* aşteaptă apariţia unui eveniment.

Notaţia grafică pentru stare este prezentată în figura 9.

A white rectangular object with black text

Description automatically generated

Figura 9. Notaţia grafică pentru stare

Există două stări particulare şi anume *starea iniţială* şi *starea finală*.

***Starea iniţială*** (Fig. 10a) este starea din care pleacă entitatea modelată.

***Starea finală*** (Fig. 10b) este starea în care entitatea modelată îşi încheie existenţa.

A white background with black lines

Description automatically generated

Figura 10: Notaţii grafice pentru starea iniţială (a) şi starea finală (b)

Elementele care caracterizează o stare sunt:

* ***Nume*** - identifică în mod unic o stare; numele este reprezentat de o succesiune de şiruri de caractere.
* ***Acţiuni de intrare/ieşire*** - sunt acţiuni ce se produc la intrarea, respectiv ieşirea din starea respectivă.
* ***Substări*** care pot fi
  + *concurente* (simultan active) – figura 11a
  + *disjuncte* (secvenţial active) –figura 11b

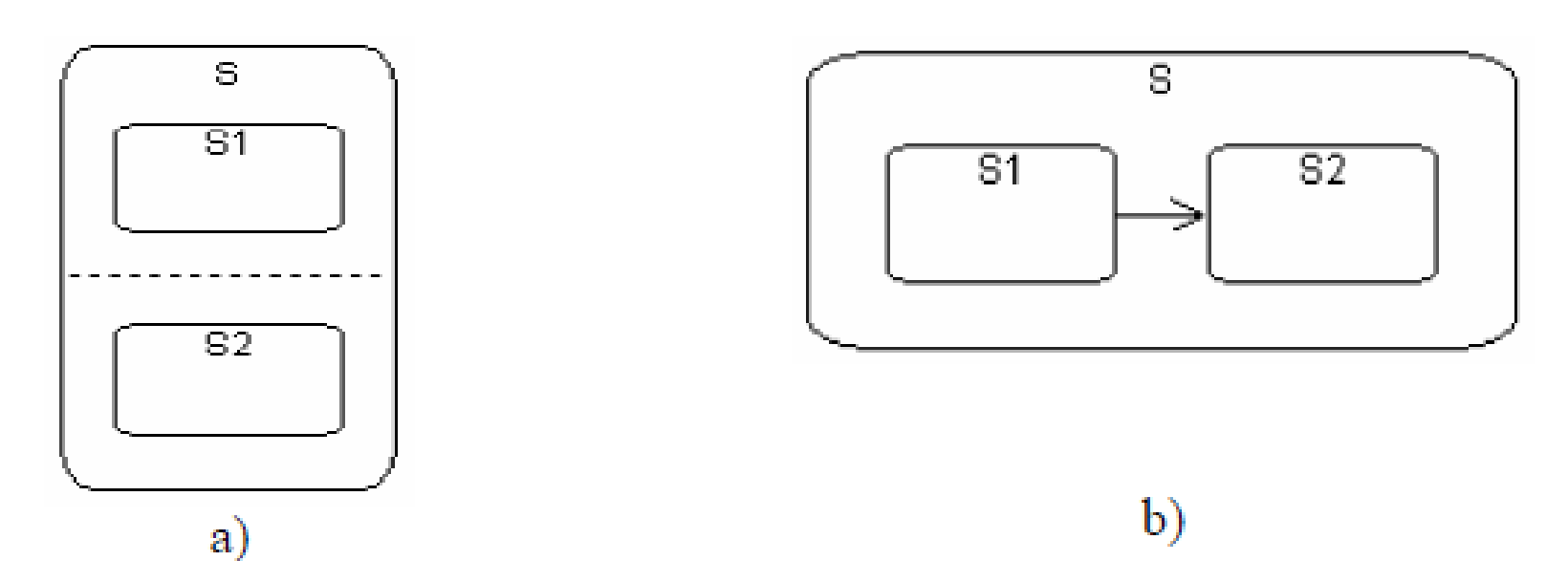


Figura 11: Notaţii grafice pentru substări concurente (a) şi disjuncte (b)

* ***Tranziţii interne*** - sunt acţiuni şi activităţi pe care obiectul le execută cât timp se află în acea stare; se produc între substări şi nu produc schimbarea stării obiectului.

A diagram of a diagram

Description automatically generated with medium confidence

Figura 12. Notaţia completă pentru stare

Forma generală a unei tranziţii interne este:

* nume\_eveniment(lista\_parametri)[condiţie\_gardă]/acţiuneunde:

nume\_eveniment– identifică circumstanţele în care acţiunea specificată se execută;

* condiţie\_gardă– condiţie booleană care se evaluează la fiecare apariţie a evenimentului specificat; acţiunea se execută doar când rezultatul evaluării este true; acţiunea – poate folosi atribute şi legături care sunt vizibile entităţii modelate.

După cum se poate observa din Fig. 12, două evenimente au notaţii speciale: *entry* şi *exit*. Aceste evenimente nu pot avea condiţii gardă deoarece se invocă implicit la intrarea, respectiv ieşirea din starea respectivă.

Activităţile sunt precedate de cuvântul cheie do.

Pentru a arăta că o stare conţine substări, se foloseşte cuvântul cheie include, urmat de numele diagramei substărilor (vezi exemplul din Fig. 13).

A diagram of a cell phone

Description automatically generated

Figura 13. Exemplu de stare

**Tranziţia.**

O *tranziţie* reprezintă o relaţie între două stări indicând faptul că un obiect aflat în prima stare va efectua nişte acţiuni şi apoi va intra în starea a doua atunci când un anumit eveniment se petrece.

Notaţia grafică pentru tranziţie se poate observa în Fig. 14.

A diagram of a system

Description automatically generated with medium confidence

Figura 14. Notaţia grafică pentru tranziţie *Starea sursă* reprezintă starea din care se pleacă.

*Eveniment* este evenimentul care declanşează tranziţia.

*Condiţie gardă* (guard condition) este o expresie booleană. Aceasta se evaluează la producerea evenimentului care declanşează tranziţia. Tranziţia poate avea loc numai dacă condiţia este satisfăcută.

*Acţiune* - opţional se poate specifica o acţiune care să se execute odată cu efectuarea tranziţiei.

*Starea destinaţie* reprezintă starea în care ajunge obiectul după efectuarea tranziţiei.

**Diagram

Description automatically generated**

Fig. 15 – Diagrama de stare