UNIVERSITATEA "DANUBIUS" GALAȚI
DEPARTAMENTUL DE STUDII PENTRU ÎNVĂȚĂMÂNT LA DISTANȚĂ

GABRIELA VÎRLAN
Cristina Maria Enache
Cristina Gabriela Zamfir

Proiectarea sistemelor informatice de gestiune

C S I D

EDITURA FUNDAȚIEI ACADEMICE "DANUBIUS" GALA□I
2008

Cuprins

Capitolul 1
Noiuni fundamentale

În informatică, la fel ca în orice ştiinţă se operează cu noţiuni, concepte, mărimi primare şi derivate. Aceste concepte, noţiuni sunt corelate prin legi pentru descrierea diferitelor fenomene şi aplicaţii care îi sunt proprii. Determinarea şi definirea acestor concepte şi legi în mod riguros şi clar, permite şi uşurează abordarea, înţelegerea şi însuşirea informaticii ca ştiinţă şi ca meserie.

Legea care se are în vedere înainte de toate, este legea potrivit căreia calitatea informaţiilor de ieşire dintr-un sistem este în funcţie de calitatea informaţiilor la intrarea în sistem.

Informatica a apărut ca ştiinţă în al cincilea deceniu al secolului nostru, în strânsă legătură cu apariţia şi dezvoltarea calculatoarelor electronice. Din acest motiv informatica a fost un timp denumită *„ştiinţa calculatoarelor*".

1.1. Noţiunea de informaţie

Rezolvarea corespunzătoare a problemelor din ce în ce mai complexe, în toate activităţile vieţii sociale şi economice, reclamă ridicarea calităţii activităţii de conducere la toate nivelele.

Perfecţionarea acestor activităţi are ca premiză **informaţia**. În sens primar, termenul de informaţie înseamnă o înştiinţare, în general „un mesaj despre anumite lucruri sau evenimente care au avut, au, sau vor avea loc".

Prin informaţie se înţelege *orice mesaj care măreşte gradul de cunoaştere al unei fiinţe umane în raport cu mediul înconjurător* (altfel spus, informaţia reprezintă cantitatea de noutate adusă de un mesaj din lumea reală înconjurătoare), fiind elementul cel mai dinamic al sistemelor informaţionale, respectiv informatice, fiind elementul care dă viaţă acestor sisteme. În activitatea de conducere informaţia este „materia primă" cu care se lucrează în vederea luării deciziilor. Informaţia reprezintă obiectul prelucrării şi totodată, principala categorie de resurse utilizate în sistemele informaţionale, respectiv informatice, alături de celelalte categorii de resurse: personalul de specialitate, echipamentele de prelucrare şi instrumentele de utilizare a acestor echipamente (metode şi tehnici de analiză şi proiectare, limbaje de programare, pachete standard de programe de aplicaţii, sisteme de operare, metode şi tehnici de organizare şi prelucrare a informaţiei etc.).

Cu cât informaţia este mai clară, mai concisă cu atât deciziile luate sunt mai corecte. Ea este cea care stă la baza formulării deciziilor, a acţiunilor şi a îndeplinirii obligaţiilor. La baza informaţiei stă noţiunea de **dată**. Prin dată se înţelege *orice mesaj primit de un receptor, sub o anumită formă*. Nu orice dată poate fi o informaţie. Relaţia dintre date şi informaţii este ilustrată în figura 1.1.

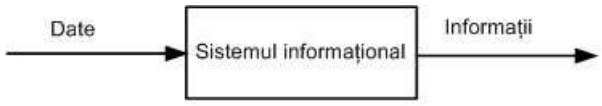


Fig.1.1. – Relaţia dintre informaţii şi date

O definiţie foarte concretă asupra diferenţei dintre informaţie şi dată a fost dată de **Peter Ferdinand Drucker** (1909-): *„Informaţia este o dată plină de scop şi de înţeles*".

Toate datele care intră în sistem sunt analizate, triate şi numai o parte dintre ele vor alcătui informaţiile necesare luării unei decizii.

1.2. Noţiunea de sistem

Conceptul de sistem apare în forma embrionară încă din filozofia antică greacă. Afirmând că întregul este mai mult decât suma părţilor, **Aristotel** dă o primă definiţie noţiunii de sistem, care se va dezvolta şi va evolua pentru a ajunge la forma actuală de abia la începutul secolului XX.

Cel care pune bazele unei teorii închegate privind teoria sistemele (este considerat fondatorul teoriei generale a sistemelor) este biologul german **Ludwig von Bertalanffy** (1901-1972) care între anii 1928 şi 1950 publică o serie de lucrări reprezentând începuturile teoriei generale a sistemelor şi a sistemelor deschise. Astfel, el a dedus principiile universale care sunt valide pentru sisteme în general. În acest sens L. von Bertalanffy a reformulat mai întâi conceptul clasic al sistemului şi l-a determinat drept o categorie prin care se cunosc relaţiile dintre obiecte şi fenomene [Bertalanffy]. Conceptul nou dat sistemului reprezintă un set de componente interdependente, o entitate complexă în care spaţiul-timp arată asemănările structurale.

În general prin **sistem** se înţelege orice secţiune a realităţii în care se poate identifica un ansamblu de elemente materiale sau nemateriale (echipamente, metode, tehnici, fenomene, obiecte, procese, concepte, personal etc) interconectate printr-o mulţime de relaţii reciproce, precum şi cu mediul înconjurător şi care acţionează în comun în vederea realizării unor obiective bine definite.

În conformitate cu Open University (1980), un sistem este un ansamblu cu un scop (obiectiv), care are componente (părţi) ce coexistă în vederea deservirii unui interes uman particular, dar care se schimbă la părăsirea sistemului.

Pe baza celor spuse mai înainte se poate spune ca prin *sistem* se înţelege un ansamblu de elemente, structurat cu ajutorul relaţiilor, ale cărei părţi se află într-o puternică interdependenţă şi independenţă faţă de mediul înconjurător, atât elemente, cât şi relaţiile endogene (relaţiile dintre elementele sistemului) sau exogene (relaţiile dintre elementele sistemului şi mediul înconjurător) având un caracter dinamic, iar existenţa şi funcţionarea să fiind subordonată unui scop.

Pentru realizarea funcţiilor unui sistem sunt necesare informaţii. Fiecare sistem are ataşat un anumit număr de informaţii, care poartă amprenta sistemului respectiv. Figura următoare ilustrează corelaţiile dintre un sistem, diferite subsisteme, precum şi funcţiile acestuia.

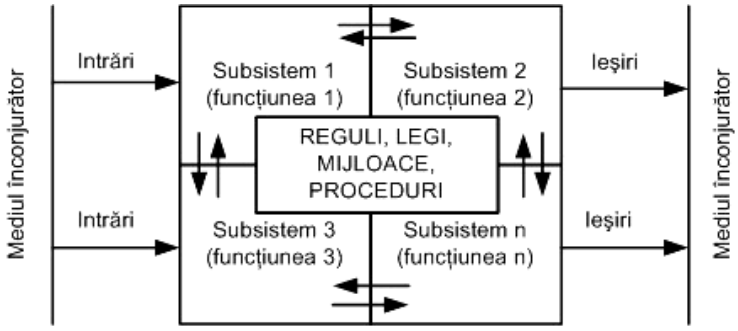


Fig. 1.2. – Relaţiile dintre sistem şi mediul înconjurător

Pentru a caracteriza noţiunea de sistem este necesar să se pună în evidenţă următoarele cinci caracteristici (care reies din definiţia sistemului):

1. mulţimea de elemente;

2. relațiile (endogene) între elemente sistemului;
3. relațiile cu mediul înconjurător (exogene), intrările şi ieşirile în/şi din sistem;
4. caracterul variabil în timp al elementelor şi relațiilor;
5. scopul, finalitatea sistemului.

Societatea comercială (firma) ca exemplu de sistem satisface în totalitate cele cinci laturi ale definiției unui sistem:

1. mulțimea de elemente este alcătuită din: salariații, elementele materiale, clădiri, materii prime, produse, mijloace financiare, informaționale etc.;
2. , 3. şi 4. Relațiile între aceste elemente, cât şi cu mediul înconjurător, au un caracter dinamic, complex;
5. scopul este de a fabrica produse şi a presta servicii, obținând beneficii.

Se poate rezuma că un sistem este un ansamblu de elemente; fiecare element al sistemului poate fi un subsistem, care la rândul lui poate fi considerat sistem format din elemente. Mulțimea relațiilor între componentele unui sistem, precum şi a relațiilor între componente şi ansamblu formează **structura sistemului**. Modul în care elementele unui sistem sunt dispuse între ele reprezintă *structura statică* a sistemului. Dacă legătura dintre aceste elemente este de compoziție, de apartenență, de utilizare, de vizibilitate a unui element asupra altuia, se spune că există o *relație statică* între elemente; de asemenea există şi o *interfață*, care reprezintă schimbul dinamic între două elemente (un flux de informații). Mulțimea caracteristicilor unui sistem, la un moment dat, determină **starea sistemului**. Un sistem îşi adaptează comportamentul la cerințele mediului său. De capacitatea de adaptare a sistemului şi de viteza de reacție va depinde durata să de supraviețuire. Reacția sistemului la un stimul trebuie să fie cât mai rapidă şi cât mai adecvată cu putință. [Vătui, 2000].

Sistemele sunt de mai multe tipuri, iar în funcție de criteriul de clasificare utilizat acestea sunt:

- după natura lor:
  - sisteme naturale (organismele vii);
  - sisteme elaborate (tehnice, economice, conceptuale – aici se regăsesc sistemele informaționale);
- după modul de funcționare:
  - deschise (ieşirile nu influențează intrările);
  - închise (intrările sunt influențate de către ieşiri);
- după comportament:
  - deterministe (se cunosc intrările, ieşirile, şi regulile care leagă intrările de ieşiri);
  - probabilistice (nondeterministe, incerte).

1.3. Noțiunea de sistem cibernetic

Pentru analiza comportamentului sistemelor, în ansamblul lor, s-a propus conceptul de „cutie neagră" care reprezintă sistemul privit ca un tot, făcând abstracție de procesele sale interne. Cutia neagră primeşte impulsuri din partea mediului înconjurător (*intrările* în sistem) şi după ce preia aceste impulsuri, le transformă în acțiuni asupra mediului (*ieşirile* din sistem).

Acest sistem devine **sistem cibernetic**, atunci când apare fenomenul de reglare (numită conexiune inversă sau *feed-back*), şi care poate fi caracterizat ca în figura 1.3.

Sintetizând, se poate spune că noțiunea de sistem cibernetic a apărut ca o necesitate de a privi un anumit sistem izolat din punct de vedere informațional. Aceasta presupune că în acest sistem cantitatea de informație este finită, că orice intrare de informație din mediu în sistem (intrare informațională) şi orice ieşire de informație din sistem în mediu (ieşire informațională) sunt controlabile sau observabile.
Prin **sistem cibernetic** se înțelege deci un sistem având cel puțin o buclă de reglaj (feed-back) prin care se aplică de la ieşirea sistemului un semnal la intrarea acestuia, unde un mecanism de comparație

permite ca rezultatul compunerii semnalului de ieşire cu cel de intrare să fie transmis blocului de decizie.

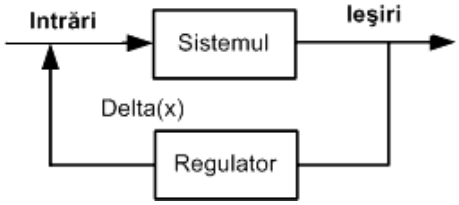


Fig. 1.3. – Legătura feed-back într-un sistem cibernetic

Sistemele cibernetice constituie o clasă importantă de sisteme, iar – după cum se va vedea – sistemele informaționale şi informatice sunt sisteme cibernetice. Schema clasică a unui sistem cibernetic cu o buclă de reglaj este arătată în figura următoare.

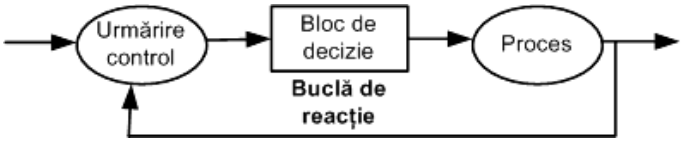


Fig. 1.4. – Legătura feed-back într-un sistem cibernetic

Un exemplu simplu ne poate arăta diferența între un sistem şi un sistem cibernetic: de pildă, un autoturism este un sistem mecano-electric, dar nu este un sistem cibernetic, un autoturism împreună cu conducătorul auto constituie un sistem cibernetic.

1.4. Noțiunea de sistem informațional

Un sistem poate fi controlat printr-un alt sistem, denumit *sistem de conducere / comandă* sau *de pilotaj*. Acest sistem de conducere intră în legătură cu un *sistem operant* ce asigură, la rândul său, transformarea unor fluxuri de intrare în fluxuri de ieşire.

Sistemul de conducere are la bază anumite obiective determinate de specificul domeniului analizat, cadrul legislativ în materie şi cerințele strategice ale domeniului. Rezultă că sunt fixate obiective clare şi directe pentru procesul de management, ce trebuie îndeplinite prin interacțiunea dintre sistemul de conducere şi cel operant.

Între sistemul de conducere şi cel operant intervine un *sistem informațional* (un sistem de interfa□ă între sistemul de conducere □i cel operant), definit ca un set finit de concepte, metode, tehnici, procedee, modele, instrumente şi procese utilizate pentru prelucrarea informațiilor şi a interacțiunilor provenite de la sistemul operant, în vederea transformării lor în date ce pot fi furnizate sistemului de conducere, în condiții de eficiență economică acceptabilă, într-un context operațional controlabil, în limitele cadrului legal al domeniului supus analizei, în scopul realizării funcțiilor organismului respectiv şi a atributelor conducerii acestuia.

Potrivit teoriei sistemelor orice organism conține aceste trei subsisteme: de conducere, informațional şi operant. Fiecare dintre aceste subsisteme au un rol bine definit, şi anume:

- ***sistemul operant (condus)*** este acela pe care se bazează activitatea organismului analizat, şi care are rolul de a transforma fluxurile primare şi / sau informaționale în fluxuri secundare sau de prelucrare;
- ***sistemul de conducere (de pilotaj)*** asigură declanşarea activității decizionale aplicată întregului organism analizat. Acest subsistem permite monitorizarea, reglarea, coordonarea şi controlul activității respectivului organism în mediul său specific. El decide asupra organizării, evoluției şi

interconexiunilor cu subsistemul operant şi informaţional. În acest context, sistemul de conducere este implicat fundamental în asigurarea eficienţei activităţii organismului;

- ***subsistemul informaţional*** îndeplineşte rolul de prelucrare manuală / automată a informaţiilor transmise de către sistemul operant, în raport cu cerinţele cadrului legislativ şi cu particularităţile organizării şi funcţionării domeniului analizat, în scopul furnizării datelor necesare controlului activităţii globale asigurate de către sistemul de conducere. Practic, subsistemul informaţional asigură, în general, stocarea, prelucrarea şi difuzarea datelor necesare subsistemelor de interfaţă: de conducere şi operant. Putem spune că subsistemul informaţional are următoarele funcţii:
  - cunoaşterea şi specificul prelucrărilor realizate la nivelul subsistemului operant;
  - furnizarea de date pertinente, exacte şi operative subsistemului de conducere;
  - implementarea funcţiilor esenţiale relative la informaţiile cu specific caracteristic domeniului analizat:
    - generarea de informaţii;
    - memorarea acestor informaţii;
    - prelucrarea acestor informaţii;
    - comunicarea acestor informaţii.

Se observă că sistemul informaţional se află în *relaţie de subordonare* faţă de sistemul de conducere al sistemului. Relaţia de subordonare este dată de faptul că sistemul informaţional trebuie să producă acele informaţii care sunt utile procesului de conducere al sistemului. Aceasta se concretizează prin cerinţele şi deciziile emise de acesta faţă de sistemul informaţional.

Tot prin *sistem informaţional* mai înţelegem şi partea dintr-un sistem care răspunde sensului restrâns al analizei, adică sistemul privit exclusiv din punct de vedere al culegerii, transmiterii, prelucrării şi stocării informaţiilor. O definiţie completă a noţiunii de sistem informaţional a fost dată de Buckingham (1987): "*Sistem informaţional este un sistem care asamblează, memorează, prelucrează şi obţine informaţii relevante pentru o organizaţie (sau pentru o societate) astfel încât informaţiile să fie accesibile şi utile celor care doresc să le utilizeze şi anume: manageri, staf, clienţi şi cetăţeni. Sistemul informaţional este un sistem care presupune activitate umană (socială) care poate implica sau nu prelucrarea pe calculator*".

Sistemul informaţional reprezintă ansamblul informaţiilor, surselor şi nivelurilor consumatoare, canalelor de circulaţie, procedurilor şi mijloacelor de tratare a informaţiilor din cadrul sistemului căruia îi este ataşat. Sistemul informaţional se mai poate defini drept ansamblul de resurse, circuite şi procese informaţionale. Orice activitate specifică, sau organism specific, are un sistem informaţional specific.

În societatea umană orice activitate este definită de acte normative, care implicit determină şi delimitează şi sistemul informaţional aferent acestei activităţi.

Orice sistem informaţional trebuie să asigure organismului căruia îi aparţine informaţii complete, în cantităţi suficiente, corecte şi la nivelul de operativitate cerut de nivelele consumatoare. Orice firmă are propriul său sistem informaţional, care ajută conducerea firmei, prin informaţiile care le furnizează, să ia deciziile optime pentru conducerea acesteia.

Sistemul informaţional are un <u>*scop propriu*</u> şi anume acela de a deservi activitatea de conducere cu informaţii de fundamentare a deciziilor şi are şi <u>*metode proprii*</u> de realizare a scopului.

În cadrul sistemului informaţional economic se disting două subsisteme care se află în relaţii de interdependenţă asigurând în acelaşi timp şi existenţa întregului sistem într-o stare de echilibru (figura 1.5.). Cele două subsisteme sunt:

- sistemul conducător al procesului informaţional;
- procesul informaţional al sistemului economic.

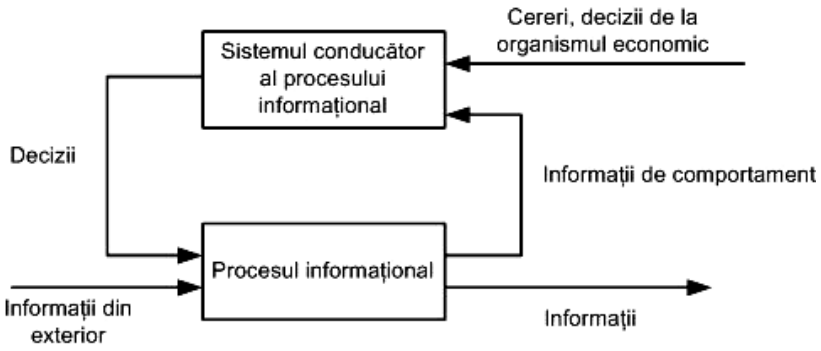


Fig. 1.5. – Relațiile dintre cele două subsisteme ale sistemului informațional

În cadrul **procesului informațional** au loc o serie de transformări prin intermediul cărora datele despre sistemul condus (real) şi informațiile primite din afară sunt transformate în informații utile procesului de conducere a sistemului economic. Procesul informațional poate fi separat pe mai multe etape şi anume:

1. etapa de *culegere a informațiilor*, care face legătura directă cu sistemul real din care rezultă datele;
2. o etapă de *prelucrare a informațiilor* în care datele intră în componența diferitelor modele, suferă o serie de procese de grupare, selectare, astfel încât se obțin informații necesare procesului de conducere.

În interiorul şi între aceste etape, informația circulă prin aşa numitul *proces de transmitere a informației.*

Sistemul conducător al procesului informațional *primeşte* de la procesul informațional informații referitoare la comportamentul acestuia; de asemenea mai *primeşte* de la sistemul conducător al organismului economic, cerințe şi decizii privitoare la nevoile de informare ale acestuia. Pe baza acestor cerințe sistemul conducător al procesului informațional *întocmeşte* programe prin intermediul cărora procesul informațional să poată furniza sistemului conducător al organismului economic informațiile cerute. De asemenea, se pot *întocmi* programe privitoare la înlăturarea unor informații considerate inutile, îmbunătățirea calității informației conform constatărilor în procesul utilizării lor, modificarea termenelor de livrare etc.

Sistemul informațional se află în relație de subordonare față de sistemul de conducere al sistemului economic. Relația de subordonare este dată de faptul că sistemul informațional trebuie să producă acele informații care sunt utile (cerute) procesului de conducere al sistemului economic. Aceasta se concretizează prin cerințele şi deciziilor emise de acesta față de sistemul informațional.

1.5. Noțiunea de sistem informatic

Elementul revoluționar al mutațiilor care au determinat saltul calitativ al sistemelor informaționale este axat pe dezvoltarea şi perfecționarea mijloacelor tehnice şi a procedurilor de prelucrare a informațiilor în scopul automatizării prelucrării datelor. Automatizarea prelucrării datelor cu ajutorul calculatorului electronic prezintă saltul calitativ major de automatizare a unor părți ale sistemului informațional.

Apariția calculatoarelor electronice şi constituirea ***informaticii*** ca ştiință, reprezintă al treilea moment important în istoria informațională a omenirii, după limbaj şi tipar. Calculatorul electronic a produs o revoluție în societatea umană în ansamblul ei, cu implicații deosebite în sfera informațională (prin modificarea activității intelectuale a omului), cu preponderență în activitatea de conducere (management).

Dezvoltarea echipamentelor de calcul şi îndeosebi a calculatoarelor a contribuit la perfecționarea modelelor economice, ceea ce a făcut posibilă apariția sistemelor informatice şi de conducere. Astfel, informatica se integrează organic în activitatea economică şi socială, iar domeniile de utilizare a acestora se diversifică în ritm rapid, prin includerea unor noi activități umane.

Ținând cont de cele spuse mai înainte se poate afirma că partea automatizată cu ajutorul calculatorului, din cadrul sistemului informațional al unui sistem se numeşte sistem informatic (prima dată aceste sisteme erau cunoscute sub denumirea de *sisteme de prelucrare automată a datelor*, **SPAD**).

O definiție primă defini□ie a unui sistem informatic "prin *sistem informatic* se înțelege ansamblul de componente hardware/software, proceduri şi oameni reunite şi organizate pentru a prelucra date, în vederea îndeplinirii anumitor sarcini şi realizării unor performanțe măsurabile prin criterii stabilite".

O defini□ie completă: **sistemul informatic** este format dintr-un set finit de metode, tehnici, procedee, modele, strategii, instrumente, sisteme de tehnică de calcul, sisteme de comunicație de date, personal specializat în informatică, sisteme organizatorice, restricții şi facilități legislative în materie, utilizate pentru generarea, transmiterea, prelucrarea algoritmică, difuzarea şi interpretarea rezultatelor în vederea îndeplinirii funcțiilor organismului şi a atributelor sistemului de gestiune (monitorizarea, reglarea, coordonarea, controlul). Toate aceste elemente au rolul de a asigura o funcționare optimă şi o reglare de tip conexiune inversă (feed-back) a întregului sistem aferent unui organism , în condiții de eficiență economică şi rentabilitate financiară acceptabile.

Sistemul informatic poate avea caracter informațional-decizional sau numai strict informațional, după cum este sau nu orientat spre rezolvarea, alături de problemele pur informaționale, şi a celor decizionale.

Din definiția sistemului informatic rezultă că între acesta şi sistemul informațional, căruia îi este subordonat, există anumite raporturi, care în principal se referă la:

1. existența unui raport de compoziție, sau apartenență prin care sistemul informatic este o parte a sistemului informațional;
2. orice sistem informatic există numai în cadrul unui sistem informațional care-l cuprinde şi-l subordonează funcțional, utilizându-l ca infrastructura să tehnică;
3. sistemul informatic poartă amprenta organismului pe care îl reprezintă.

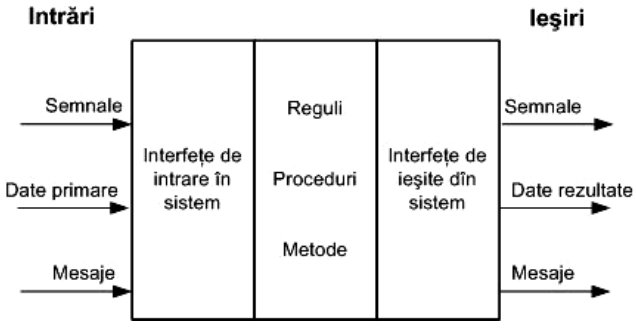


Fig.1.6. – Sructura unui sistem informatic

Sistemul informatic se poate prezenta sub forma unei „***cutii negre***", caracterizată de intrări, ieşiri, reguli, proceduri, mijloace şi metode.

Sistemul informatic primeşte datele de la sistemul operant şi, prin intermediul unei baze de proceduri asociate, asigură prelucrarea multiplă a acestora, în conformitate cu un sistem procedural bazat pe algoritmi de prelucrare şi de calcul, în vederea obținerii unor date de ieşire sub formă de rapoarte, indicatori sintetici, grafice, alte ieşiri sub formă mixtă şi / sau ieşiri către alte sisteme informatice.

În mod practic, sistemul informatic foloseşte colecţii de date organizate sub formă de:

- baze de date gestionate prin sisteme de gestiune a bazelor de date (SGBD);
- baze de tabele gestionate prin sisteme de tip LOTUS sau derivate din acestea (EXCEL, QUATTRO).

Intrările sunt asigurate prin operaţiunile externe generate de activitatea desfăşurată la nivelul sistemului (subsistemului) operant, operaţii ce generează un flux de date care vor determina operaţii de actualizare asupra bazelor de date. Deci, intrarea unui sistem informatic este informa□ia care se înregistrează, sub forma datelor de intrare, în sistemul informatic respectiv, cu scopul de a fi prelucrată. Intrările unui sistem informatic reprezintă totalitatea datelor introduse în sistem cu scopul de a fi prelucrate, în conformitate cu regulile func□ionale □i / sau organizatorice specifice sistemului economic care-l utilizează □i cu legisla□ia în vigoare.

Intrările unui sistem informatic pot fi ob□inute prin:

- Introducerea datelor din documente (tastatură, scanare).
- Transfer electronic de date (re□ea locală, re□ea de arie întinsă).

Prelucrările unui sistem informatic reprezintă totalitatea opera□iilor efectuate asupra datelor înregistrate în şi respectiv pentru ob□inerea informa□iilor necesare func□ionării unui sistem economic.

Ie□irile unui sistem informatic reprezintă totalitatea rezultatelor prelucrărilor efectuate asupra datelor înregistrate în şi respectiv, prin interpretarea cărora se pot ob□ine informa□iile care fundamentează deciziile manageriale pe toate nivelele organizatorice ale sistemului economic care îl utilizează. Ele reprezintă:

- Datele de pe documentele întocmite de sistem.
- Datele exportate către alte sisteme informatice.

Reiese faptul că ieşirile sunt obţinute ca urmare a prelucrării datelor de intrare, fiind concretizate în următoarele tipuri de rapoarte:

- *rapoarte – liste – situaţii*: conţin indicatori sintetici şi / sau analitici sub formă tabelară, obţinuţi prin diferite procese de prelucrare aplicate asupra colecţiilor de date, fiind necesare luări de decizii cu caracter tactic, strategic şi operativ;
- *indicatori sintetici*: conţin elemente sintetice / analitice cu un grad maxim de prelucrare şi reprezentative, afişabile pe ecran, şi sunt folosite pentru informarea şi luarea operativă a deciziilor de către manageri sau personalul de specialitate;
- *grafice*: arată tendinţa unor fenomene sau procese pe o perioadă de timp sau ponderea unor elemente într-un asemenea fenomen;
- *mixte*: se prezintă sun forma unor documente elaborate pentru factorii de decizie şi conţin text explicativ, repoarte, indicatori sintetici şi grafice, inclusiv combinaţii ale acestor tipuri de ieşiri;
- *ieşiri către alte sisteme informatice*: sunt utilizate pentru transmiterea on-line de date între diverse organisme, fiind necesare informării reciproce sau raportării cu privire la fenomene şi procesele actuale, trecute şi / sau viitoare.

Principalele **obiective** ale unui sistem informatic sunt:

1. cunoaşterea stării sistemului;
2. cuantificarea rezultatelor sistemului;
3. creşterea operativităţii în activitatea managerială a sistemului;
4. utilizarea eficientă a resurselor sistemului.

Fiecare organism (societate, întreprindere, firmă) are de îndeplinit mai multe atribuţii pentru a-şi atinge scopul pentru care a fost creat (pentru care, de fapt, funcţionează). Pentru ca un sistem informatic să fie integrat unui organism el trebuie să fie ataşat în mod intim tuturor funcţiunilor

automatizabile din cadrul acestuia. Deci fiecărei funcțiuni a organismului, îi va corespunde un subsistem informatic, care toate împreună vor alcătui sistemul informatic al acestuia.
Încă de la începutul utilizării tehnicii de calcul în prelucrarea datelor la nivelul unei organizații s-a impus necesitatea ca sistemul informatic să fie împărțit în mai multe subsisteme. Partea sistemului informatic asociată unei funcțiuni se numeşte **subsistem** şi este caracterizată de un ansamblu de operațiuni similare care realizează obiectivele sistemului pentru partea automatizată din funcțiunea respectivă.

Împăr□irea pe subsisteme a fost determinată de o serie de factori precum:

1. tipul de producție şi ramura din care face parte întreprinderea;
2. gradul de complexitate al sistemului informatic;
3. experiența şi numărul membrilor echipei de proiectare;
4. condițiile existente în întreprindere şi caracteristicile sistemului informațional existent;
5. gradul de utilizare al echipamentelor;
6. situația codificării şi calitatea documentației;
7. gradul de pregătire al documentației de proiectare şi tehnologice etc.

Astfel s-au constituit „subsisteme" pe:

1. funcțiunile întreprinderii (producție, comercial, financiar-contabil, personal etc.);
2. pe grupuri de activități de conducere (planificare şi urmărire, gestiunea stocurilor, programarea, lansarea şi urmărirea fabricației, gestiunea livrărilor etc.);
3. subsistemul care conține modelele pentru pregătirea deciziilor;
4. subsistemul care asigură gestiunea bazei de date etc.

Modul de constituire al subsistemelor a fost în general influențat de gîndirea „realizării de aplicații", care a premers gîndirii sistemice; care consideră că împărțirea pe subsisteme trebuie subordonată în principal necesităților implementării sistemului informatic, sau mai corect al diferitelor părți ale sistemului informatic şi, deasemenea, trebuie avut în vedere în mod consecvent caracterul cibernetic al oricărui subsistem al sistemului informatic (structurat ca sistem cibernetic).

Deoarece în cadrul oricărui ansamblu de operațiuni se pot regrupa subansambluri de operațiuni de acelaşi tip, tot aşa în cadrul unui subsistem se pot decupa subsisteme**,** denumite şi aplicații, tratate pe calculator ca lucrări independente. Aplicația, sau subsistemul, stă în raport cu subsistemul cum stă partea față de întreg.

Cele mai importante subsisteme, din cadrul unui sistem al firmei, sunt:

1. subsistemul de resurse umane;
2. subsistemul de producție;
3. subsistemul financiar –contabil;
4. subsistemul de marketing.

În cazul societăților industriale sistemul informatic se structurează de obicei în următoarele subsisteme:

- planificarea tehnico-economică;
- pregătirea fabricației;
- programarea, lansarea şi urmărirea producției;
- marketing;
- personal-salarizare;
- financiar – contabil.

Capitolul 2
Sisteme informatice

Pornind de la defini□ia sistemului informatic, sistemul informatic este format dintr-un set finit de metode, tehnici, procedee, modele, strategii, instrumente, *sisteme de tehnică de calcul*, *sisteme de comunicație de date*, *personal* specializat în informatică, *sisteme organizatorice*, restricții şi facilități legislative în materie, utilizate pentru generarea, transmiterea, prelucrarea algoritmică, difuzarea şi interpretarea rezultatelor în vederea îndeplinirii funcțiilor organismului şi a atributelor sistemului de gestiune (monitorizarea, reglarea, coordonarea, controlul), se poate defini arhitectura unui sistem informatic, care este alcătuit din următoarele componente:

- sistemul de tehnică de calcul este *acel sistem care permite realizarea anumitor activități, având la bază prelucrarea informației, alcătuit din două subsisteme principale: hardware şi software*, este deci alcătuit din calculatoarele, împreună cu produsele software utilizate, care sunt puse la dispozi□ia utilizatorilor sistemului informatic;
- sistemul de comunica□ie de date este reprezentat de *ansamblul de mijloace tehnice interdependente care asigură transferul informa□iilor între două sisteme de calcul oarecare, aflate la o anumită distan□ă unul fa□ă de altul*. Un sistem de comunica□ie leagă între ele două sau mai multe calculatoare cu scopul de a asigura transferul de date între ele, prin intermediul unei re□ele de calculatoare. În func□ie de tipul re□elei de calculatoare (re□ea locală, metropolitană, de arie întinsă etc.) aceste sisteme de comunica□ie sunt mai mult sau mai pu□in complexe. Datorită faptului că în ultimii ani Internet-ul a început să fie utilizat ca suport pentru afacerile firmelor dezvoltarea acestor sisteme de comunica□ie capătă o amploare din ce în ce mai mare.
- personalul specializat în informatică □i utilizatorii, sau sistemul de resurse umane, este format din *mul□imea de persoane cu pregătire diferite care interac□ionează cu sistemul informatic în vederea îndeplinirii func□iilor acestuia*. În func□ie de sarcinile □i / sau func□iile pe care le ocupă fiecare persoană în raport cu sistemul informatic de dezvoltat, sau utilizat, raport care este stabilit în func□ie de nivelul de pregătire a fiecărei persoane în parte, aceasta va face parte dintr-o anumită categorie de personal: personal de specialitate □i personalul de exploatare.
- sistemul organizatoric reprezintă *totalitatea metodelor □i tehnicilor organizatorice utilizate în vederea atingerii func□iei sistemului informatic*.

Sistemele informatice depind în mod esențial de resursele software care ajută utilizatorii să lucreze cu echipamentul hardware şi să acceseze rețelele de calculatoare. Astfel software-ul asigură introducerea, prelucrare, ieşirea, datelor precum şi controlul sistemelor informatice.

2.1. Categorii de produse software

Software-ul se clasifică în două mari categorii de programe:

A. *Software-ul de aplica□ii* este reprezentat de programele care acționează direct asupra unui domeniu de utilizare particular pentru a asigura procesarea informațiilor necesare utilizatorilor finali. Acest tip de programe se regăse□te într-o multitudine de variante, versiuni, platforme de lucru, producători şi preț. Software-ul de aplicații are o zonă determinată de utilizare (de exemplu, domeniul economic) iar în cadrul ei, zona poate fi şi mai specifică (de exemplu, calcul tabelar). În prezent se remarcă folosirea cu precădere de pachete integrate (*suite*) pentru activitățile de birou care au un rol important în creşterea producțivității. Acestea cuprind: procesor de texte, calcul tabelar, baze de date, programe de prezentare şi de lucru pe Internet. Cele mai importante aplicații de acest tip sunt: MS Office, Corel Word Perfect Office, Lotus Smart Suite şi Sun Star Office.

O clasificare a acestora pentru domeniul economic poate fi:

1. *Programe de procesare de text.*
2. *Programe de calcul tabelar.*
3. *Programe de management al bazelor de date.*
4. *Programe de prezentări şi grafică.*
5. *Programe de lucru în Internet şi poştă electronic.*

Avantajele acestor grupuri de aplicaţii sunt:

- determină creşterea productivităţii, facilitează comunicaţia în interiorul organizaţiei, permit lucrul în reţea şi integrează aplicaţiile Internet;
- includ toate tipurile de programe necesare activităţii de birou;
- costul achiziţiei unui pachet integrat este mai mic decât al aplicaţiilor cumpărate individual;
- toate aplicaţiile din pachet arată aproape identic ceea ce le face mai uşor de învăţat şi de lucrat;
- au facilităţi importante de transfer al informaţiilor dintr-o aplicaţie în alta.

Dezavantaje:

- multe din facilităţile oferite de aceste programe nu sunt folosite de utilizatori;
- au nevoie de resurse mari pentru a funcţiona corespunzător (spaţiu pe disc, memorie);
- pot afecta viteza de lucru şi puterea sistemului.

Tot în această categorie producătorii de software au dezvoltat o categorie de pachete integrate de nivel mediu care au preluat o serie din caracteristicile celor de mai sus. Acestea combină funcţiile mai multor tipuri de programe în unul singur. Cele mai cunoscute sunt: MS Works, Lotus Works, Claris Works. Ca avantaje ar putea fi resursele modeste pe care le utilizează şi costul sub 100 US $, iar dezavantajul este că nu au performanţele programelor individuale.

B. ***Software-ul de sistem*** sunt programele care gestionează şi manipulează resursele şi activităţile unui computer fiind o interfaţă între computer şi software-ul de aplicaţii. În această categorie se includ următoarele programe:

1. *Sisteme de operare* – asigură interfa□a dintre calculator □i sofyware-ul de aplica□ii. Exemple: MS DOS,Windows, Linux, Mac OS, OS2 etc.
   Sistemul de operare este un program principal, permanent stocat în memorie, lansat în execuţie la pornirea calculatorului care îndeplineşte funcţii de coordonare şi control asupra resurselor fizice ale calculatorului şi care intermediază dialogul om-calculator. Deci, el asigură şi alte facilităţi ca vizualizarea de fişiere, ştergere, copieri de fişiere. Mai are rolul de a înmagazina şi regăsi datele pe disc, organizează intrările de date de la tastatură, trimite datele către imprimantă şi verifică dacă tipărirea decurge normal, controlează monitorul. Există diverse sisteme de operare, diferite între ele în funcţie de arhitectura calculatorului, codurile maşină, instrucţiuni etc.
   Scopul unui sistem de operare este de a face comenzile maşinii „transparente" utilizatorului. Acelaşi sistem de operare poate fi implementat pe o varietate largă de maşini, cu viteze şi capacităţi diferite. Cele mai multe softuri sunt făcute ca lucrul cu ele să fie intermediat de sistemul de operare. Datorită faptului că sistemul de operare este acelaşi, programele pot rula pe diverse tipuri de computer fără sau cu mici modificări. Acest element de portabilitate a ajutat ca softul să se diversifice.
2. *Programe de reţea* – care asigură comunicaţia dintre calculatoare. De exemplu: Windows NT, 2000 sau Novell.
3. *Programe utilitare* – ce efectuează diverse activităţi precum: diagnoza utilizării sistemului şi resurselor, protecţia antivirus, arhivarea documentelor, optimizarea sistemului etc.

2.1.1. Procesoarele de text

A da o definiţie exactă unui procesor de texte este o activitate hazardată, sortită eşecului, datorită complexităţii şi diferenţelor specifice între atât de diversele pachete numite *procesoare de texte*. Cert este că un procesor de texte este un pachet de programe ce lucrează asupra unui text în vederea

imprimării acestuia, chiar dacă textul va fi transmis prin fax sau poştă electronică, va fi stocat în fişiere text, ce vor reprezenta surse ale unui program sau documentaţii on-line etc.
Evoluând de la minieditoare de texte, cu performanţe scăzute, ca *Edit* pentru sistemul de operare MS-DOS, *Write* pentru sistemul de operare Windows şi ajungând până la procesoare profesionale ca *Word for Windows*, *AmiPro*, *JustWrite*, procesoarele de texte realizează acum toate operaţiile necesare pentru editare, machetare, formatare, vizualizare, tipărire, verificare şi multe alte operaţii ce dau unui text un aspect foarte plăcut şi atractiv. Răspândirea interfeţelor grafice, de tipul **WYSIWYG** (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et – *ceea ce vezi* (*pe ecran*) *este ceea ce vei avea* (*la imprimantă*)), au dus la apropierea din ce în ce mai mare a tipografului, tehnoredactorului, secretarei şi nu în ultimul rând, al oricărui om ce vrea să scrie un text, de calculator prin procesorul de texte şi îndepărtarea să de maşina de scris clasică.

Integrarea aplicaţiilor. Procesoarele de texte nu lucrează în general singure, ci cooperează cu alte aplicaţii ce realizează operaţii complexe. Orice program care se respectă ştie să se descurce cu câteva formate grafice, spreadsheet-uri, cu formatele celor mai populare baze de date şi mai ales cu diverse formate de text. Pe lângă procesorul de texte se găsesc de obicei, un editor de ecuaţii, un editor de grafică, un editor de stil şi artă şi multe altele.

Mediul *Windows*, prin multitasking (reprezintă o modalitate de lucru oferită de un sistem de operare, prin care un calculator poate executa mai multe task-uri în acelaşi timp; un ***task*** este o aplicaţie sau program care este executat la un moment dat), oferă procesoarelor de texte posibilitatea de a folosi în comun alte aplicaţii ale sistemului. Astfel, există posibilitatea de a ataşa documente mesajelor trimise prin fax sau poştă electronică. De asemenea se pot crea legături dinamice (**DDE** – **D**ynamic **D**ata **E**xchange, realizează un transfer al datelor dintr-o aplicaţie Windows în alta, sau **OLE** – **O**bject **L**inking and **E**mbedding, permite inserarea unui obiect creat printr-un anumit program în interiorul altui obiect creat printr-un program diferit; legătura între cele două obiecte fiind gestionată de sistemul de operare) cu datele unei aplicaţii *Windows*, astfel încât orice actualizare a datelor originale se reflectă în document, dacă aplicaţiile sunt deschise simultan (DDE) sau la cerere (OLE).

Se poate sintetiza că programele de procesat text permit crearea, modificarea şi tipărirea de documente aflate în formă digitală. Avantaje utilizării lor:

- capacitatea de a edita broşuri, manuale etc.;
- utilizarea şi convertirea de documente din/în formatul HTML pentru Internet;
- corectarea textului, gramaticii şi traducere.

Programele **DTP** (**D**esk**T** **P**ublishing) sunt programe mai specializate în producerea de cărţi, manuale, broşuri şi care oferă mai multe facilităţi în acest domeniu (grile, formatări, stiluri).

2.1.2. Programe de calcul tabelar

Sunt programe utilizate pentru analize de business, planificare şi modelare. Acestea oferă un mod electronic de înlocuire a tabelelor de hârtie, creionului şi a calculatorului de buzunar. Programele se prezintă sub forma unui tabel cu un număr foarte mare de rânduri şi coloane, permit combinarea acestora, calcule, grafice, analize etc. Totodată permit accesarea şi interogarea bazelor de date de pe Internet pentru al le prelucra.

Trebuie amintit faptul că locul unde s-au cerut pentru prima dată calculatoare PC foarte puternice a fost *Wall Street*. Domeniul economic este, se pare, unul primordial pentru producţia de hardware şi software. Dacă din punct de vedere al hardware-ului problemele erau în cea mai mare parte rezolvate, din păcate software-ul era mai puţin dezvoltat şi devenise apanajul unor specialişti. Aceasta a determinat necesitatea unor programe care să fie foarte uşor accesibile şi „novicilor”, nemaifăcând necesară intermedierea prin specialişti (analişti progrmatori).

Cum într-o economie de piață apariția unei nevoi duce la apariția mijlocului prin care este satisfăcută, pe fondul dezvoltării hardware-ului, la începutul anilor 1980 au apărut primele programe de calcul tabelar numite şi *spreadsheets*. Programele de calcul tabelar sunt pachete de programe (dacă sună prea pretenţios ele se numesc mai simplu programe) care permit manipularea datelor aranjate sub formă de tabel. Mai simplu spus pentru cei ce au tangență cu activitatea de birou, ace□tia trebuie să se gândească la un document cumulativ dar foarte foarte mare.

Avantajele utilizării acestor programe constă în faptul că odată stabilite formulele, relațiile între celule şi introduse datele, imediat rezultatele calculelor vor fi afişate şi mai mult, la modificarea datelor sau o altă introducere ulterioară, rezultatele calculelor vor fi refăcute imediat sesizându-se modificarea intervenită.

Posibilitatea de a crea modele fiind foarte dezvoltată se pot dezvolta modele extrem de complexe, se pot elabora şi diverse analize şi calcule financiare (când ne-am referit la faptul că spreadsheet-urile acoperă domeniul economic vom mai menționa că există o multitudine de funcții financiare, contabile, statistice, matematice şi chiar inginereşti, care vin să uşureze şi mai mult lucrul cu aceste programe). Aceste analize se pot face uşurință folosind comenzile *what-if* (ce s-ar întâmpla dacă ...), *solve for* (rezolvă pentru ...) sau optimizări.

2.1.3. Software pentru baze de date

Fiecare organizație lucrează cu un număr mai mic sau mai mare de documente şi date. Unele date se află în arhive, altele în circulație, unele sunt create în interior, altele în exterior. Indiferent de forma, conținutul sau proveniența lor, ele formează un volum de informație care este vital pentru buna administrare şi succesul activității respectivei organizații. Toate aceste colecții de date împreună cu aplicațiile ce utilizează datele respective, formează o bază de date, mai exact o *bază de date* şi un sistem de gestiune a bazelor de date (SGBD), care de fapt reprezintă software-ul utilizat pentru dezvoltarea aplica□iilor cu baze de date.

Deci bazele de date cuprind colecții structurate de date, sistemul de gestiune a acestora cu interogări, machete, rapoarte, aplicații precum şi mediile de interfațare şi dezvoltare a aplicațiilor referitoare la colecțiile de date.

Gestiunea datelor în cadrul unor organizații mari sau a celor cu un flux rapid de date devine o problemă care consumă timpul, energia şi nervii unui număr considerabil de funcționari. Consumă deci resurse, bani. Căci indiferent dacă este vorba de facturi sau de datele secrete ale unui nou produs, există situații în care afaceri extraordinare sau simpla imagine publică depind de eficiența cu care sunt manipulate aceste date. Calculatoarele încearcă să facă ordine în milioanele de date de diverse tipuri ce tind să sufoce organiza□iile. Dar în spatele lor trebuie mereu să se afle cineva care să le conducă şi să înțeleagă tot fluxul acestora. Folosirea sistemelor de gestiune a bazelor de date în mod eficient înseamnă economie de timp, de muncă, de bani şi nu în ultimul rând conferă competitivitate şi stil.

Trăsături standard

În orice sistem de gestiune a bazelor de date primul lucru care trebuie făcut este stabilirea structurii (organizării) datelor. Structura relațională a reuşit să se impună în domeniul SGBD-urilor, astfel încât marea majoritate a acestora sunt relaționale, de aceea se vor trata numai acest tip de SGBD.

O **bază de date** reprezintă o colecție de date organizate sub formă de tabele (în terminologia bazelor de date, un fişier se numeşte tabelă), date între care există anumite legături (numite relații logice), şi care permite căutarea rapidă şi regăsirea informațiilor utilizând calculatorul. Tabelele sunt organizate pe linii □i coloane. Coloanele se mai numesc *câmpuri* (*fields*) iar liniile se mai numesc *înregistrări* (*records*).

Utilizarea SBBD-urilor este dată de anumite facilită□i, printre care se enumeră:

- Una din facilitățile care se va găsi la majoritatea SGBD-urilor, dar diferit realizată de la program la program, este *scrierea aplicațiilor* (*programarea*). Având la dispoziție limbaje de programare specifice bazelor de date, se pot dezvolta diverse proceduri ce personalizează cu adevărat aplicația respectivă. Programele pot fi scrise instrucțiune cu instrucțiune sau descrise prin evenimente, pot exista structuri de gen WHILE, FOR, CASE sau nu. Se pot genera în unele SGBD-uri şi fişierele sau codurile executabile .EXE, ce pot fi de sine stătătoare (*stand alone*) sau minimale, adică folosesc biblioteci externe cu care se pot lega dinamic.
- O altă facilitate foarte importantă este s*ecuritatea datelor*. Se pot proteja astfel, prin parole, înregistrări sau fişiere, grupuri de fişiere şi aplicații. Există diverse nivele de protecție, stabilite în general, de administratorul bazei de date. În cazul lucrului în rețea, securitatea datelor este foarte importantă. Sunt probleme create de accesul la date partajat sau accesul la date numai pentru vizualizare şi nu pentru modificare sau restructurare. Poate fi protejată chiar şi intrarea în SGBD, prin parole pe două sau mai multe nivele.
- Orice aplicație mai serioasă trebuie prezentată într-o formă cât mai accesibilă. Acest lucru se realizează de obicei, prin ceea ce se numeşte interfața aplicației sau *meniurile*. Meniurile sunt dialoguri la nivel utilizator prin care se pot selecta diverse căi de urmat în aplicația respectivă. Există în general un generator de meniuri ce scoate în final cod-program, care poate fi legat de restul părților din aplicație. Se pot folosi meniuri *Pop-up, Pull-down*, butoane radio, de bifare, de apăsare etc., ce pot fi lansate printr-o combinație de taste (*shortcut*) sau selectate cu mouse-ul şi care dau aplicației o față mult mai prietenoasă şi mai ales mai accesibilă.
- *Relații între fişiere* (*join*) sunt necesare în cazul în care nu se dore□te sau nu se poate să se păstreze informații complete despre un obiect. Descrierea obiectului, aflat în altă tabelă, poate fi accesată prin stabilirea unei relații cu această descriere, evitându-se astfel multiplicarea aceleiaşi informații şi inconsistența informației. În bazele de date relaționale această trimitere se realizează printr-o adresă, un cod. Pot fi stabilite relații de tipul unu-la-unu (*one to one*) sau unu-la-mulți (*one to many*). Ele pot fi şterse sau modificate în mod interactiv sau descriptiv.
- O altă problemă realizată în mod diferit de diversele SGBD-uri este cea a importului şi exportului de fişiere sau date, adică a *compatibilității*. În lumea bazelor de date sunt încetățenite câteva formate standard de tip *xBase*, *ISAM (Indexed Sequential Access Method)*, ceea ce face ca SGBD-urile să poată lucra nu numai cu baze de date proprii ci şi cu altele. Astfel se pot porta între ele baze de date *dBASE* cu *FoxPro*, *Paradox*, *Access*, *Informix*; ba chiar există un SGBD numit *Magic* care nu are format propriu ci foloseşte drivere specifice pentru celelalte baze de date.
- În fine, o altă facilitate a SGBD-urilor este dată de existen□a *Help*-urilor interactive sau obişnuite, ce pot scoate utilizatorul din multe încurcături. Bazele de date sub *Windows* oferă suport pentru DDE şi OLE (grafică, imagine) ce permite elaborarea unor aplicații cu imagine, sunet şi animație - *multimedia*.

2.1.4. Programe de prezentare

Programele de prezentare sunt utilizate pentru a converti diverse informații în elemente grafice. Aceste programe oferă capacități multimedia (utilizarea de fotografii, animații, sunete şi secvențe video) dar şi posibilități de a face prezentări pentru Internet.

Avantaje:

- asigură o comunicare simplă şi sugestivă;
- construiesc prezentări eficiente;
- oferă posibilități de interactivitate.

Programele de grafică s-au dezvoltat uimitor după apariția unei interfețe grafice şi a unor plăci grafice puternice, cu posibilitatea definirii unui număr mare de culori şi forme. Folosesc, în general, mouse-ul, creioanele optice, uneltele care sunt la dispoziția utilizatorilor □i sunt afişate sub forma unor butoane

pe ecran (creioane, foarfecă, gumă, pensule, rolluri, lupe etc.). Dispun de palete de culori prin care se poate selecta o culoare predefinită sau defini o culoare utilizator (*custom*). Generează formate grafice specifice, unele de tip *bitmap* (orientate pe puncte), altele de tip *vectorial* (orientate pe curbe, linii).

Programele de grafică actuale sunt programe orientate pe obiecte grafice, dar mai rezistă încă şi programe de grafică orientate ecran. Există în această lume a graficii, *programe de grafică utilitară sau de afaceri* (*Bussines Graphics*) care sunt extrem de pretenţioase din punct de vedere hardware dar sunt suficiente de simple în utilizare. Ele sunt dotate cu seturi ample de desene prefabricate (*Clip Art*) şi au în general formate vectoriale. Dintre acestea putem aminti: *Powerpoint (Microsoft), Freelance Graphics (Lotus), Corel Presentations (Corel), Persuasion (Aldus), Charisma (Micrografx).*

O a doua categorie o constituie ***procesoarele de imagini*** ce sunt utilizate pentru pictură, afişe, colaje, retuşuri fotografice, animaţie, prelucrare video. Prelucrările oferite de astfel de programe merg de la reglaje de contraste şi luminozitate, efecte speciale de vizualizare şi transformare a imaginii, deformări plane şi spaţiale, efecte de posterizare, până la colorarea imaginilor alb-negru, capturi ale ecranelor, adăugare de perspective şi multe altele.

Din această categorie se poate semnala câteva, menţionând că toate sunt sub *Windows* sau sub *Mac*:

- *PhotoShop (Adobe)*;
- *PhotoStyler (Aldus)*;
- *Picture Publisher*, *PhotoMagic (Micrografx)*;
- *Corel Draw (Corel)*;
- *Painter (Fractal Design)*;
- *Image Wizard (Image Ware)*;
- *Image Pals (ULead).*

Programele de prezentare sunt programe de grafică, sunet şi animaţie menite să realizeze mult mai şocant şi diversificat prezentări necesare lansării unui produs nou, reclame publicitare, clipuri, instruire asistată de calculator etc. Paleta lor este foarte largă, mergând de la simple programe de grafică şi ajungând la programe multimedia, adică programe ce pot lucra cu imagini video, muzică, animaţie.

2.1.5. Programe de lucru pe Internet

Între cele mai importante produse software utilizate în prezent, aflate într-o dezvoltare permanentă se află *web browser –ele*. Netscape Navigator, Internet Explorer sau Opera sunt cele mai răspândite programe care lucrează în Internet şi oferă accesul la resursele acestuia. Domeniile de utilizare a lor sunt:

- navigare în Internet;
- căutare de informaţii;
- poştă electronică;
- transferul de fişiere;
- grupuri de discuţi;
- alte aplicaţii (chat, video-chat, fax).

Poşta electronică (e-mail) a schimbat modul de lucru al oamenilor şi posibilităţile de comunicare. Acest sistem permite transmisia şi recepţia de mesaje în formă electronică (digitală) prin Internet sau alte reţele. Facilităţile po□tei electronice sunt:

- transmitere/primirea de mesaje de la/către unul sau mai mulţi utilizatori, folosirea de *mailing list*;
- confidenţialitate şi securitate;
- posibilitatea răspunsului automat;
- crearea de liste de subscripţie;
- acces la cutia poştală din mai multe locuri;
- transfer de fişiere (text sau multimedia);

- filtrarea şi sortarea mesajelor primite;
- utilizarea de agenţi inteligenţi.

2.2. Alte tipuri de produse software

Evident software-ul pentru sistemele informatice nu se opreşte aici. Există un număr mare de programe ce vor fi prezentate în continuare.

2.2.1. Programe utilitare

Programul utilitar este un program mai mic ce aduce îmbunătăţiri faţă de lucrul în sistem de operare, realizând diverse operaţii simple sau mai complexe asupra suporturilor de memorie sau datelor aflate pe aceste suporturi. Operaţiile pe care le fac programele utilitare pot fi realizate şi direct din sistemul de operare de către specialişti, lucrând cu întreruperile, gestionând memoria mai bine, însă ele se realizează foarte greu de către un nespecialist. De altfel, apariţia programelor utilitare a dus la dezvoltarea şi perfecţionarea sistemelor de operare, astfel încât aceste sisteme au integrat treptat şi multe programe utilitare (gen *Notepad, Wordpad, Paint* etc.).

Facilităţile pe care le-au adus programele utilitare referitor la hard disc au fost legate în primul rând de posibilitatea de afişare în regim text sau grafic a structurii pe foldere şi subfoldere a acestuia. Afişarea sub formă arborescentă (*tree*) cu ramificaţiile detaliate sau nu, cu relaţiile dintre fişiere şi structură, afişarea în ferestre cu posibilitatea rapidă de copiere, mutare dintr-o fereastră în alta, selectarea fişierelor mult mai rapidă, afişarea conţinutului unui fişier cu posibilitatea editării lui, dacă dimensiunea acestuia nu este prea mare, vizualizarea fişierelor şi directoarelor ordonate după diverse criterii: nume, extensie, lungime, dată sunt numai câteva facilităţi pe care le-au adus programele utilitare. Tot cu aceste programe se mai pot crea meniuri proprii în care pot fi introduse comenzi întâlnite frecvent, executarea lor făcându-se prin selectarea opţiunii din meniu sau prin scurtături (*shortcuts*). Sunt de fapt primul pas făcut spre sistemele de operare vizuale, cu interfeţe prietenoase ca *WindowsNT, Macintosh, Next.*

Din categoria acestor programe utilitare amintim:

- *Norton Desktop (Symantec)*;
- *Norton Antivirus (Symantec)*;
- *Dashboard (Hewlett-Packard)* etc.

Ele aduc şi alte facilităţi referitoare la date, fişiere cum ar fi: protejarea prin parole a fişierelor, folderelor, discurilor, compactarea sau arhivarea datelor pentru a mări spaţiul pe disc, copierea şi ştergerea fişierelor automat (*backup*) la un anumit moment din zi curăţarea discului şi a memoriei de eventualii viruşi. Unele programe utilitare pot reface informaţii şterse (fişiere, foldere) sau pierdute prin formatare accidentală, facilităţi numite *undelete* şi *unformat*. *Defragmentarea* este o altă operaţiune prin care se grupează spaţiul liber de pe disc în mod compact (la început sau la sfârşit) astfel încât accesul la un fişier sau folder să fie mult mai rapid ca şi căutarea unui spaţiu liber pentru crearea unui fişier sau folder. Gestionarea memoriei mult mai eficient se poate face prin programe utilitare, ce pot „mări" memoria convenţională de 640 Kb prin utilizarea memoriei de la 640 K în sus (*expanded* & *extended memory*), astfel încât cantitatea de date ce poate fi prelucrată nemaifăcându-se acces la hard-disk creşte substanţial şi odată cu aceasta timpul de execuţie se micşorează. Tot pentru mărirea vitezei există utilitare ce pot anticipa ce parte a discului va fi folosită şi pot încărca în memorie informaţiile din această zonă (*chaching*). Probleme de comunicaţii pot fi rezolvate şi ele cu ajutorul programelor utilitare. Astfel de utilitare sunt:

- *Norton Utilities*;
- *PCAnywhere (Symantec)*;
- *After Dark (Berkeley Sys.)*;
- *QEMM (Quarterdeck)*;

- *Fastback (Fifth Generation).*

Odată cu dezvoltarea platformei *Windows* au apărut utilitare ce pot crea iconuri, cursoare de mouse, meniuri, dialoguri, acceleratoare etc.

2.2.2. Pachete integrate

Folosirea eficientă a unui calculator necesită cel puțin folosirea unui procesor de texte, unui program de calcul tabelar, a unei baze de date şi eventual a unui procesor de imagine. Cum majoritatea utilizatorilor au nevoie de câte un program din fiecare, multe firme oferă pachete conținând fie o colecție de programe care sunt vândute şi separat, fie un produs integrat care oferă facilitățile de procesare de texte, calcul tabelar, grafică şi baze de date simple.

Acestea se numesc *pachete integrate* şi au avantajul comunicabilității între produse, uniformității stilistice şi nu în ultimul rând al prețului mai scăzut. Dintre acestea se poate aminti:

- *MS Office* al firmei *Microsoft*;
- *Smart Suite* al firmei *Lotus*;
- *WordPerfect Office* al firmei *Corel*;
- *Star Office* al firmei *Sun*;
- *Open Office* al *Open Office Org*;

Aproape toate pachetele integrate oferă şi programe de comunicații de poştă electronică, organizatoare etc. şi folosesc Windows-ul, ceea ce asigură o portabilitate şi compatibilitate a aplicațiilor dîntr-un produs în altul.

2.2.3. Proiectare asistată pe calculator

Programele de proiectare asistată de calculator **CAD** (**C**omputer **A**ided **D**esign) sunt programe de grafică, mai pretențioase, în care precizia desenării joacă un rol foarte important. Se pot proiecta obiecte tridimensionale asupra cărora se pot aplica secționări, scalări, rotații, compuneri şi descompuneri de obiecte.

Vizualizarea obiectelor poate fi făcută în cele mai mici amănunte şi din diverse poziții. Produsele din această categorie dispun de bogate biblioteci grafice, cu multe exemple, cu imagini ce pot fi adăugate în bibliotecă, cu stiluri de linii, cu pattern-uri pentru haşuri şi suprafețe. Ustensilele de care dispun sunt uşor accesibile cu ajutorul mouse-ului, se poate folosi cu uşurință compasul, linia, guma, lupa şi multe altele, printr-o simplă apăsare a unui buton dintr-o tabelă de instrumente (*tools*).

Posibilitatea inserării textului în imagine precum şi importul de imagini de tip raster sunt alte facilități ce fac din programele de proiectare o soluție producțivă pentru crearea rapidă de desene, rapoarte, schițe, documente.

Dintre aceste programe de proiectare asistată se remarcă:

- *Auto CAD for Windows, Generic CAD, Auto Sketch (AutoDesk)*
- *Drafix CAD (Foresight)*;
- *Design CAD 2D/3D (American Design)*;
- *Toolbox Professional*;
- *Auto Architect* etc.

2.2.4. Programe de gestionare a documetelor i de birou

Programele **DMS** (**D**ocument **M**anagement **S**ystem) sunt programe specializate în organizarea şi stocarea documentelor de tip text sau imagine. Există sisteme DMS destinate în princip al prelucrării

de texte prin stocarea lor într-o arhivă indexată şi sisteme DMS destinate prelucrării de imagini cu stocarea acestora pe discuri optice.
Într-un sistem DMS se definesc mai întâi profilurile documentelor, adică informaţii ce definesc conţinutul unui document, tipul acestuia, cine l-a creat şi când, unde este stocat, legături cu alte documente. Indexarea se poate face după aceste profile sau după cuvinte cheie existente în documente. Funcţia principală a unui DMS este *regăsirea rapidă a documentelor şi vizualizarea acestora*. Dacă specificaţiile nu sunt complete atunci se furnizează lista documentelor ce satisfac condiţiile de interogare, pe baza indexului profilelor sau chiar cuvintelor din documente. Un sistem DMS este destinat lucrului în reţea, deoarece la un document trebuie să aibă acces, eventual în acelaşi timp, mai multe persoane precum şi faptului că un document poate fi realizat de către mai multe persoane. Apar astfel probleme de actualizare a versiunilor documentelor, modificări ireconciliabile de către persoane diferite, securitate şi control asupra accesului la un document, ce sunt rezolvate în mod diferit de la program la program.

Calea către biroul fără hârtii este deschisă prin aceste DMS, dar ele necesită în general un scanner, un PC puternic, o imprimantă rapidă, discuri optice în cazul prelucrării de imagini, ceea ce poate duce la un cost destul de ridicat, deocamdată! De altfel probleme legate de recunoaşterea optică a caracterelor (OCR), arhivare, dezarhivare rapidă sunt probleme de actualitate în domeniul DMS şi care dau o imagine fantastică viitorului.

Din familia sistemelor DMS se pot remarca:

- *Auto EDMS* al firmei *ACS Telecom*;
- *Notes* 3.0 al firmei *Lotus*;
- *Apple Search* al firmei *Apple*;
- *Acrobat* al firmei *Adobe*.

Produsele **PIM** (**P**ersonal **I**nformation **M**anager) includ un planificator de activităţi corelat cu un calendar, o agendă de adrese, cu posibilitatea de conectare la telefon prin placă de fax sau modem, notesuri pentru diverse documente, dicţionare etc.

Un PIM este un produs uşor de folosit, rapid în utilizare şi nu necesită un spaţiu mare pe disc. Este un program ce stă resident în memorie, adică poate fi accesat oricând se dore□te, chiar dacă este activ un alt program. Manipularea produsului se face de obicei cu ajutorul mouse-ului ca de exemplu răsfoirea unor pagini ce apar pe ecran, selectarea unor litere din repertoar. În prezent multe din funcţiile acestora se regăsesc în programele de e-mail (Outlook, de exemplu). Acestea pot importa date dintr-o bază de date sau spreadsheets sau interschimba date, informaţii cu un alt utilizator de PIM. Mai poate conţine agende cu cărţi de vizită, cu organizarea timpului pe zile şi ore, având posibilitatea realizării unor legături între persoane şi notesul cu activităţi sau întâlniri, anunţării programate în cazul unor evenimente ca aniversări, activităţi importante.

Dintre aceste produse se pot remarca:

- *Act* al firmei *Symantec (CSI)*;
- *Organizer* al firmei *Lotus*.

Posibilitatea listării la imprimantă, în diverse formate standard sau definite de utilizator, a oricărei componente a PIM-ului, introducerii parolelor de acces, personalizării agendei prin notaţii proprii, inserării de poze şi imagini în agendă sunt şi alte facilităţi care fac din PIM instrumentul ce poate elibera un birou de teancul de hârtii, agende, cărţi de vizită etc.

2.2.5. Shareware □i public domain

Oferta *shareware* constă din programe diverse testate sau netestate, disponibile la preţuri foarte mici, cu posibilitatea încercării lor înainte de a le cumpăra. Dacă programul se dovedeşte a fi util atunci este obligatorie virarea unei sume modice în contul autorului produsului, nu al vânzătorului, putând primi

astfel documentații şi versiuni îmbunătățite ale programului respectiv. Piața produselor *shareware* cuprinde practic toate domeniile, cum ar fi: programe utilitare, jocuri, programe educaționale, aplicații grafice, limbaje de programare, spreadsheets, procesoare de texte, baze de date etc.

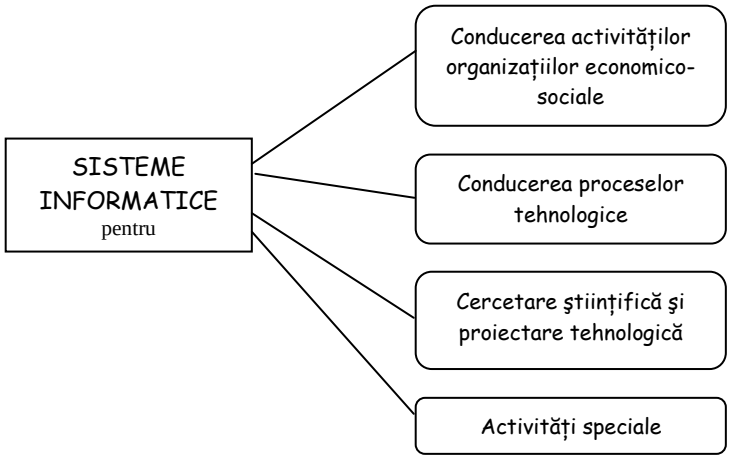
Oferta *public domain* constă din produse software la prețuri foarte mici pe care cumpărătorul le poate da prietenilor sau chiar le poate revinde fără nici o restricție.

Ce se urmăreşte prin produsele *shareware* şi *public domain*? În primul rând intrarea în „lumea bună" a informaticii, de asemenea promovarea produselor, câştigarea unui număr mare de clienți, virtuali cumpărători şi ai unor alte produse ale aceleaşi firme (nu obligatoriu tot produse shareware).

2.3. Clasificarea sistemelor informatice

Clasificarea sistemelor informatice se face în funcție de anumite criterii, şi anume: [Lungu&alt, 1995], [Oprea, 1999]

1. <u>În funcție de domeniul de utilizare</u>, acestea se clasifică în patru grupe, care sunt prezentate în următoarea figură.



   a. Specific **sistemelor informatice pentru conducerea activităților organizațiilor economico-sociale** este faptul că datele de intrare, de regulă, sunt fumizate prin documente întocmite de om, iar datele de ieşire sunt fumizate de către sistem tot sub formă de documente (liste, rapoarte etc.) pentru perceperea acestora de către om.
   b. Spre deosebire de acestea, **sistemele informatice pentru conducerea proceselor tehnologice** se caracterizează prin aceea că datele de intrare sunt asigurate prin intermediul unor dispozitive automate care transmit sub formă de semnale (impulsuri electronice) informații despre diverşi parametri ai procesului tehnologic (presiune, temperatură, umiditate, nivel), iar datele de ieşire se transmit, de asemenea, sub formă de semnale unor organe de execuție, regulatoare, care modifică automat parametrii procesului tehnologic. Se execută în acest fel controlul şi comanda automată a procesului tehnologic. Astfel de sisteme sunt folosite în locurile în care este periclitată intervenția în mod direct a factorului uman. Exemple de asemenea sisteme sunt cele pentru laminarea oțelului, pentru procesele din petrochimie, pentru fabricarea cimentului, a hârtiei, centrale nucleare etc. În mod firesc apar diferențe între obiectivele celor două categorii de sisteme, cele pentru conducerea proceselor tehnologice având ca obiective îmbunătățirea randamentului agregatelor, urmărirea siguranței în funcționare, creşterea indicatorilor de calitate a produselor, îmbunătățirea altor indicatori tehnico-economici.

c. **Sisteme informatice pentru activitatea de cercetare ştiinţifică şi proiectare tehnologică** asigură automatizarea calculelor tehnico-inginereşti, proiectarea asistată de calculator şi alte facilităţi neesare specialiştilor din domeniile respective.
d. **Sistemele informatice speciale** sunt destinate unor domenii specifice de activitate, ca exemplu: informare şi documentare, tehnico-ştiinţifică, medicină etc.

2. Un alt criteriu de clasificare al sistemelor informatice economice este în funcţie de nivelul ierarhic ocupat de sistemul economic în structura organizatorică a organizaţiei, conform căruia avem următoarea clasificare:
   a. **Sisteme informatice pentru conducerea activităţii la nivelul organizaţiilor economice**. Acestea pot fi descompuse în subsisteme informatice asociate funcţiunilor organizaţiilor economico-sociale sau chiar unor activităţi.
   b. **Sisteme informatice pentru conducerea activităţii la nivelul organizaţiilor economico-sociale cu structură de grup**. În această categorie sunt incluse sistemele informatice la nivelul regiilor autonome.
   c. **Sisteme informatice teritoriale**. Sunt constituite la nivelul unităţilor administrativ-teritoriale şi servesc la fundamentarea deciziilor adoptate de către organele locale de conducere.
   d. **Sisteme informatice pentru conducerea ramurilor, subramurilor şi activităţilor la nivelul economiei naţionale**. Se constituie la nivelul ramurilor, subramurilor şi activităţilor individualizate în virtutea diviziunii sociale a muncii şi specificate în clasificarea economiei naţionale. Sunt elaborate şi administrate de ministerele, departamentele sau organele care au prin lege sarcina de a coordona metodologic grupele respective de activităţi. Principala lor funcţie constă în fundamentarea şi reglarea echilibrului dezvoltării economico-sociale în profil de ramură.
      Aceste sisteme vor trebui să realizeze elaborarea de variante a proiectului de plan în profil de ramură, încărcarea optimă a capacităţilor de producţie, folosirea intensivă a maşinilor, utilajelor şi instalaţiilor, urmărirea şi controlul realizării sarcinilor de plan şi a celor privind calitatea producţiei, perfecţionarea produselor şi a tehnologiilor, înnoirea producţiei şi asigurarea de noi produse, utilizarea superioară a potenţialului material şi uman din ramura respectivă.
   e. **Sisteme informatice funcţionale generale** ce au ca atribut principal faptul că intersectează toate ramurile şi activităţile ce au loc în spaţiul economiei naţionale, furnizând informaţiile necesare coordonării de ansamblu şi sincronizării lor în procesul reproducţiei din cadrul economiei de piaţă. În această categorie sunt cuprinse sistemele pentru planificare, statistică, financiar-bancar etc.

3. Un alt criteriu de clasificare al sistemelor informatice este acela după aportul acestuia în actul decizional.

   Decidentul dintr-o unitate are prin sistemul informatic un puternic suport pentru fundamentarea deciziilor sale. Acest suport implementează modele matematico-economice din domeniul specific de activitate sau cu caracter general. Este situaţia clasică de realizare a sistemelor informatice ca asistent al decidentului. Acestea execută o mică parte din activitatea decidentului, rolul lor important fiind de culegere şi prelucrare automată a datelor. Este perioada de până în jurul anului 1970, când două discipline au venit în sprijinul ştiinţific al sistemului informatic-decizional: cercetările operaţionale şi teoria deciziei. În această perioadă apar şi primele **sisteme suport de decizie** (SSD) ca sisteme pentru prelucrarea automată a datelor împreună cu sistemele de luare a deciziilor.
   Anii 1970 au însemnat o creştere puternică a fluxului informaţional în toate domeniile de activitate, a bazelor de date şi a teleprelucrării. Acestea au permis prelucrarea unui volum mai mare de date şi o comunicaţie mai rapidă şi mai eficientă. Rolul sistemului informatic creşte în raport cu decidentul, ajungând să fie un colaborator al acestuia. De multe ori, aceste sisteme

informatice execută o parte însemnată din activitatea decidentului evoluând, astfel spre sisteme suport de decizie.

Începând cu anii 1970, bazele de date au evoluat spre relațional şi distribuit, iar rețelele de calculatoare locale şi generale au devenit curente. Informația care se prelucrează se diversifică foarte mult, volumul de date este tot mai mare, iar complexitatea prelucrărilor de asemenea. Sistemele informatice încep să execute mare parte din activitatea de rezolvare a problemelor de decizie, devenind experte în domeniu, evoluând astfel spre **sisteme expert** (SE). Volumul de date mare şi complexitatea deosebită a datelor care circulă pe magistralele (rețelele) informaționale internaționale în momentul de față tind să sufoce sistemele informatice bazate pe relațional. Abordarea orientată obiect, precum şi realizarea de baze de cunoştințe, pe maşini tot mai puternice, tind să rezolve această problemă.

Sistemele expert, precum şi sistemele suport de decizie sunt de fapt sisteme informatice dedicate. Iată câteva aspecte comune şi deosebiri dintre ele:

a. Tehnologia de realizare se păstrează în mare parte pentru toate cele trei tipuri de sisteme. Pe de o parte, SSD şi SE au preluat în metodologia lor de realizare majoritatea activităților din metodologia de realizare a SI, adoptând o parte din ele. Pe de altă parte, metodologia de realizare a şi a evoluat mult odată cu apariția SSD şi SE, preluând o serie de elemente de simplitate, flexibilitate, precum şi stilul de lucru în paşi mărunți şi reluări succesive. Ideea ca un sistem informatic, ca dealtfel orice produs informatic, se realizează "la cheie" prin etape care odată realizate nu se mai pot relua, nu mai este agreată. Stilul de lucru de la sistemele expert care presupune realizarea unei versiuni care nu este nici ultima, nici cea mai bună, urmând apoi să se realizeze versiuni succesive pentru perfecționare şi dezvoltare, este tot mai mult utilizat şi în realizarea sistemelor informatice.
b. Toate folosesc abordarea sistemică pentru studierea şi rezolvarea problemelor. Aceasta este o cale eficientă pentru învingerea complexității şi păstrarea coerenței.
   Abordarea sistemică presupune o serie de caracteristici în procesul de cunoaştere, caracteristici care se regăsesc la realizarea tuturor celor trei tipuri de sisteme. Aceste caracteristici sunt:
   - extragerea sistemului studiat se face din mediul înconjurător;
   - definirea problemei şi descrierea ei se face cantitativ şi/sau calitativ;
   - se definesc mijloacele posibile pentru rezolvarea problemei;
   - se formulează diferite variante de rezolvare a problemei;
   - se compară variantele şi se alege cea mai bună (cea care satisface cel mai bine cerințele).
c. Modul de rezolvare al problemelor păstrează direcții comune care caracterizează sistemul uman de prelucrare şi evaluare a informației. Acest lucru este firesc în SSD şi SE, şi se accentuează în şi prin abordarea orientată obiect. În acest sens, se îmbină aspectele descriptive cu cele imperative, neprocedurale cu cele procedurale, în funcție de sistem punându-se accentul pe unul sau altul dintre aceste aspecte. Modulul rezolutiv se bazează în special pe raționamente, dar şi pe algoritmi în SE şi se bazează în special pe algoritmi, date şi raționamente în SSD şi SI. Raționamentul se bazează pe modelul logic şi nu pe cel fizic, ceea ce înseamnă că primează relevanța şi mai puțin precizia. Acest lucru este valabil atât în mecanismul de inferență din SE, cât şi în procesul decizional din SSD. În SI, în modelul prelucrativ, contează mai mult precizia şi mai puțin relevanța. Aplicațiile cu baze de cunoştințe sunt în ultimă instanță aplicații informatice care permit rezolvarea de probleme dificile prin simularea raționamentului uman asupra unor cunoştințe specifice unui domeniu dat.
d. Cele trei sisteme, deşi au arhitecturi diferite, păstrează şi elemente comune. Toate au colecții de date care sunt fişiere sau baze de date în SI, baze de cunoştințe în SSD (baza de date şi baza de modele) şi SE (baza de cunoştințe şi modele). În plus față de SI, SSD conțin o bază de module care este de fapt o bibliotecă de module permanente sau de uz temporar. Acestea pot fi ale utilizatorului sau realizate de firme specializate. Modulele operative, tactice sau strategice, de calcul sau analiză etc. Dimensiunile acestor module pot fi de la o singură relație până la foarte multe. Legat de această bază de module, SSD va conține un mecanism de

construire sau generare a modulelor, va avea posibilitatea să restructureze un modul, să-l actualizeze şi să opereze asupra modulelor pentru a obține rapoarte de ieşire. În loc de colecțiile de date din SI, SE conțin o bază de cunoştințe în care se descriu obiectele din lumea reală. Ea conține fapte (axiome) şi reguli (care pot descrie şi modele). Atât SSD, cât şi SE au componente pentru învățare care achizionează noi cunoştințe. Această componentă lipseşte ca atare în SI, deşi sunt încercării în acest sens de a fi inclusă.

De asemenea, toate sistemele conțin interfețe cu utilizatorul care tind să devină tot mai prietenoase, uşor de folosit şi interactive. Această componentă tinde să depăşească jumătate din codul program generat, în toate cele trei sisteme. Tendința este dată de maşinile interactive actuale şi de societatea informațizată care determină o utilizare în masă a calculatoarelor. Dialogul dat de interfață trebuie să fie cât mai "natural" pentru a elimina bariera psihologică dintre om şi maşină. Stilul de dialog poate fi întrebare-răspuns, limbaj de comandă, meniu, videoformat, ferestre etc., la care se adaugă facilitățile oferite de platformele multimedia (dacă acestea sunt disponibile).

e. Toate cele trei sisteme ajută decidentul în activitatea sa, îi fundamentează decizia. Contribuția fiecărui tip de sistem la sprijinul decidentului, în fundamentarea deciziilor este prezentată în tabelul 1.1.

Tabelul 1.1. *Contribuția fiecărui tip de sistem la procesul decizional*

Tip sistem	Ajutor pentru decident	Partea executată din activitatea decidentului
SI	Asistent	O mică parte
SSD	Colaborator	O parte însemnată
SE	Expert	O mare parte

f. Problemele rezolvate cu cele trei tipuri de sisteme sunt de natură diferită, deşi au şi elemente comune (provin din lumea reală etc.) Dacă într-o problemă criteriile sunt preponderent cantitative, iar caracteristicile problemei se formulează cantitativ, modelarea se face foarte bine algoritmic şi va rezulta un SI. Dacă însă există formulări mai puțin cantitative se tinde spre SSD sau SE, care însă .nu exclud folosirea algoritmilor. Pentru problemele complexe în condiții de incertitudine, se porneşte conceptual, dar şi practic, de la baze de date clasice spre baze de cunoştințe. Acestea au la bază cunoştințe incomplete, inconsistente, incerte, imprecise, ambigui. Pentru fiecare dintre aceste categorii de cunoştințe există o logică nestandard de care se ține cont în abordarea problemei. Acest lucru se tratează bine în SSD şi SE, şi foarte greu sau imposibil de tratat în SI.

Din analiza de mai sus rezultă evoluția în anumite condiții a şi spre SSD. La SE evoluția se constată în ceea ce priveşte conceptele (sistem, componente, modele, obiecte etc.), metodologia de realizare (principalele activități, metode, tehnici etc.), soluții software de implementare (limbaje, tehnici de programare, inginerie software etc.). Pe de altă parte, din punct de vedere al organizării datelor, se constată evoluția bazelor de date relaționale spre cele orientate obiect şi spre bazele de cunoştințe. Simplificarea modelului relațional şi îmbunătățirea lui a condus spre modelul orientat obiect. De asemenea, reprezentarea prin perechile A-V (atribut-valoare) din relațional o regăsim şi în bazele de cunoştințe (exemplul din limbajul Prolog).

4. Din punct de vedere al organizării datelor sistemele informatice se clasifică în:

   a. **SI care au colecțiile de date organizate în fişiere**. Fişierele pot fi cu organizare clasică (secvențiale, indexat-secvențiale, relative) sau cu organizare specială. Acest mod de organizare a datelor este tot mai rar utilizat astăzi, şi el este acceptate doar pentru sisteme mici. În orice caz, aceste sisteme trebuie să folosească şi fişiere care permit accesul direct pentru uşurința şi rapiditatea manipulării datelor. În cadrul acestor sisteme informatice datele se introduc de la tastatură în sistemul informatic ori de câte ori este nevoie, creându-se câte un fi□ier pentru fiecare tip de prelucrare necesară sistemului.

   b. **SI care au colecții de date organizate în bază de date**. Pentru acest lucru se foloseşte un model de date care poate fi arborescent, rețea, relațional sau orientat obiect şi un SGBD

adecvat. Cel mai utilizat model este cel relațional, dar în ultimii ani sunt utilizate din ce în ce mai mult bazele de date rela□ional obiectuale. Majoritatea şi sunt de acest tip datorită avantajelor oferite de bazele de date în crearea şi manipularea colecțiilor de date.

c. **SI mixte care au colecții de date organizate în bază de date, dar şi în fişiere**. Pot apărea şi astfel de situații în realizarea unui sistem informatic, în sensul că pe lângă baza de date sunt necesare şi o serie de fişiere relativ independente prelucrate din limbaje de programare, în afara SGBD-ului. Astfel de cazuri apar mai ales atunci când se colaborează cu alte sisteme sau aplicații informatice (procesoare de text, de imagini, de calcul tabelar, e-mail etc.).

Aceste criterii nu sunt singurele utilizate în clasificarea sistemelor informatice. Astfel, alte criterii au în vedere:

- Modul de introducere a datelor în sistemul informatic.
- Modul de prelucrare a datelor generate de sistemele economice.
- Obiectivele urmărite etc.

Curs 3
Stadiul actual şi tendinţele dezvoltării sistemelor informatice

În ultimii ani asistăm la una dintre cele mai importante transformări din istorie ale infrastructurii tehnologice a societăţii. Această schimbare constă de fapt în adăugarea unui nou substrat în infrastructura tehnologică, substrat care este uzual denumit tehnologia informaţiei. În acest nou substrat se evidenţiază în mod decisiv informatica. Extinderea într-o măsură din ce în ce mai mare a tehnologiei informaţiei a devenit posibilă datorită progreselor rapide şi importante ale microelectronicii. Această extindere este pe cale de a produce o schimbare majoră în societatea noastră, şi anume, trecerea de la orientarea industrială, în care accentul se pune pe maşină şi energie, la o nouă orientare, informaţională, în care accentul se pune pe robot şi informaţie. Este evident că şi în continuare maşina şi energia vor juca un rol important, fundamental, în societatea informaţională, dar pentru noile maşini, pentru noile industrii, ca şi pentru celelalte activităţi ale omului, devin esenţiale tehnologiile informatice care au la bază electronica, informatica şi comunicaţiile moderne. [Lungu&alt, 2003]

Informatizarea activităţilor economico-sociale a cunoscut profunde transformări. În cele ce urmează se enumeră câteva din schimbările şi tendinţele ce au loc în practica dezvoltării sistemelor informatice.

1. Se manifestă în mod clar o tendinţă spre divizarea costurilor software-ului sistemelor informatice. Reducerea costurilor sistemelor informatice se datorează, pe de o parte, reducerii costurilor hardware-ului, iar pe de altă parte, reducerii costurilor software-ului. În ceea ce priveşte componenta software, putem spune că cu ani în urmă nu erau aşa de multe produse software disponibile pe piaţă. Modul obişnuit de implementare a sistemelor informatice era de a programa de unul singur software-ul necesar. Fiecare implementare tindea să fie alcătuită din software-ul pentru un anumit scop. Acest mod de lucru era extrem de scump pentru că nu se obţineau reduceri de costuri provenite din generalizarea pe scară largă a sistemului. Costurile de proiectare, realizare, menţinere şi calitate pentru fiecare componentă ar trebui suportate doar de un singur utilizator al sistemului. În prezent se manifestă o tendinţă clară în dezvoltarea sistemelor informatice bazate tot mai mult pe platformele software de nivel înalt.
   O platformă software corespunde unei platforme de aplicaţii şi conţine funcţii software de bază şi funcţii specifice aplicaţiei companiei. Prin funcţiile software de bază se definesc şi se rezolvă problemele comune aplicaţiei în proporţie de circa 80-90%, iar prin software-ul specific aplicaţieie se definesc proprietăţile comportamentale suplimentare companieie.
   O astfel de abordare oferă posibilitatea generalizării şi implementării sistemelor în mai multe unităţi economice, cu efecte imediate de divizare şi reducere a costurilor pe unitate de implementare.
   Ideea de bază a unei platforme comune de aplicaţii este într-adevăr veche. Noua invenţie este că, în sfârşit, ideea a ajuns să fie implementată.
2. Se manifestă o intensă tendinţă spre tehnologia sistemelor informatice bazate pe reţele de calculatoare.
   Creşterea complexităţii, varietăţii aplicaţiilor şi apariţia de noi produse informatice cu un raport preţ/performanţă din ce în ce mai avantajos au făcut necesară şi rentabilă conectarea între ele a calculatoarelor în cadrul unor reţele care constituie la ora actuală suportul cel mai adecvat pentru teleinformatică.
   O importanţă remarcabilă în dezvoltarea reţelelor a avut-o Internet-ul (cu semnificaţia de reţea a reţelelor) care a oferit posibilitatea accesului nelimitat la diverse tipuri de informaţii, precum şi comunicarea între diverse persoane de pe întreaga planetă conectate la Internet.

Tendințele din domeniul rețelelor de calculatoare cuprind diverse aspecte, cum ar fi apariția şi dezvoltarea de noi protocoale şi medii de comunicație ce permit viteze de transport de ordinul gigabiților/sec, dezvoltarea fără precedent a comunicațiilor fără fir, dezvoltarea rețelelor de sateliți, a accesului la distanță în scopul unor operațiuni de comerț electronic sau pentru diverse tranziții electronice on-line.

3. În domeniul organizării datelor se manifestă tendința spre baze de date orientate obiect.
   Structurile clasice de date bazate pe text şi valori numerice fie se dovedesc insuficiente, fie complexitatea lor depăşeşte posibilitățile de stocare şi prelucrare oferite de tehnologiile clasice. Aplicațiile asociate cu disciplinele tehnologice, cum ar fi proiectarea asistată de calculator, sistemele informatice geografice şi sistemele bazate cunoştințe, presupun stocarea unor cantități mari de informații cu o structură complexă. Aceste aplicații necesită suport pentru tipurile de date care nu pot fi reprezentate în sistemele clasice. Unele aplicații informatice solicită monitorizarea unor desene formate din grupuri de elemente complexe ce trebuie să fie combinate, separate, suprapuse şi modificate astfel încât să permită elaborarea unor variante de proiect. Totodată, orientarea spre multimedia aduce elemente noi în lumea informaticii. Grafica, imaginea fotografică, imaginea video, sunetul, muzica nu pot fi tratate în aceeaşi manieră cu a structurilor tabelare de denumiri şi numere.
   Dacă eforturile de extindere a tehnologiilor actuale în domeniul colectării, stocării şi prelucrării acestor noi tipuri de informații ca elemente singulare sunt tot mai adesea finalizate cu succes, nu acelaşi lucru se poate spune despre administrarea corespunzătoare a unor colecții de astfel de date.
   Bazele de date clasice sau relaționale oferă prea puțin suport teoretic şi practi pentru tipurile neconvenționale de date.
   Bazele de date orientate obiect permit crearea de obiecte complexe din componente mai simple, fiecare având propriile atribute şi propriul comportament, în acest fel ele reuşesc să ofere soluții pentru problemele şi aplicațiile amintite anterior.
4. Sisteme informatice de tip nou.
   Pentru aplicațiile tradiționale pentru care au fost concepute, sistemele relaționale satisfac cerințele acestora. Descrierea datelor sub formă de tabele corespunde bine tipului de informații manipulate de aceste aplicații. O dată cu scăderea costului calculatoarelor şi creşterea puterii lor de calcul au apărut noi aplicații care manipulează cantități mari de date. Printre acestea enumerăm: sisteme pentru proiectarea asistată de calculator, sisteme multimedia, sisteme deschise.
   Aceste aplicații există deja şi reprezintă o piață foarte importantă pentru sistemele de gestiune a bazelor de date (SGBD). Majoritatea dintre aceste aplicații nu utilizează însă un SGBD, ci sunt construite cu sisteme dedicate. Acest lucru se datorează faptului că un sistem de gestiune a bazelor de date relațional SGBDR nu oferă funcționalitățile necesare. Noile generații de baze de date vor trebui să țină cont nu numai de aplicațiile tradiționale, dar şi de noile aplicații. Utilizarea unui SGBD standard în locul unui SGBD dedicat va permite reducerea considerabilă a costului de punere în funcțiune a acestor noi aplicații. Este foarte probabil că alte tipuri noi de aplicații vor apărea. Din această cauză noile generații de SGBD vor trebui să aibă implementat conceptul de extensibilitate. Adică ele trebuie să fie capabile să administreze nu numai tipurile de aplicații identificate la un moment dat, ci să se adapteze la alte tipuri de aplicații care nu au fost prevăzute inițial.
   a. Sisteme de proiectare asistată de calculator. Aplicațiile generează un ansamblu de faze (activități) pentru realizarea unui produs. Datele manipulate sunt adesea foarte complexe, descrierea unei componente fiind dependentă în mare măsură de celelalte componente ale aceluiaşi produs. Se regăseşte aici şi o parte importantă a informației de tip regăsire documentară.
   b. Sisteme multimedia. Aplicațiile de acest nou tip au drept caracteristică aceea că administrează datele într-un mod netradițional. Exemplele cele mai cunoscute sunt aplicațiile care administrează imagini şi sunet pe lângă text şi grafică. Există deja acum aplicații comerciale care folosesc astfel de date, cum ar fi, de exemplu, aplicațiile meteorologice. Acest

tip de aplicații se caracterizează printr-un volum foarte mare de date tratate. Imaginile sunt date foarte voluminoase care necesită un suport de stocare şi prelucrare performant. Tehnologia discurilor optice numerice este adaptată acestor aplicații. Un SGBD care suportă aplicații multimedia trebuie să folosească tratamentul clasic asupra imaginilor şi să administreze legăturile de tot felul dintre acestea De exemplu, într-o aplicație meteorologică trebuie căutate printre imaginile stocate pe toate cele care ajută la detectarea unui ciclon. Pentru o astfel de operație trebuie folosită tehnica de căutare şi acces classic într-o bază de date, precum şi tehnica specifică de tratare a imaginilor. Aceleaşi observații sunt valabile şi pentru tehnica specifică de tratare a sunetului. Pentru aplicațiile multimedia este deci nevoie de o integrare tehnologică nouă cu cea tradițională.

c. Sisteme deschise. Expresia „sisteme deschise" corespunde ea însăşi unui concept vag. Ideea este de a aduce un plus de flexibilitate organizațiilor prin aplicațiile care se dezvoltă pentru ele.
   Suplețea (flexibilitatea) în dezvoltarea şi exploatarea sistemelor aferente unei unități se realizează prin:
   - paleta largă de diferite tipuri de periferice şi platforme ce pot fi interconectate;
   - uşurința de utilizare a instrumentelor de proiectare a sistemelor deschise;
   - posibilitatea interconectarii aplicațiilor cu alte aplicații dezvoltate pentru platforme diferite.

   La sistemele deschise totul începe cu problema standardelor. Anumite standarde sunt stabilite de comitete naționale şi internaționale, altele sunt impuse de grupurile de proprietari sau vânzători, altele există pur şi simplu pentru anumite produse care sunt larg utilizate (exemplu: o versiune standard de C a fost acceptată de un comitet internațional, MOTIF este o interfață promovată de un grup de ofertanți încercând să normalizeze Unix, Windows este un produs impus de proprietar - Microsoft etc.). Standardele proprietarilor sau producătorilor sunt acceptate mai uşor dacă produsul oferă facilități de interconectare cu alte produse standard (exemplu: limbajul standard de regăsire din bazele de date - SQL).
   Pentru a se evalua nivelul de deschidere al unui produs informatic şi a decide dacă acesta poate fi considerat un instrument pentru elaborarea de aplicații acesta trebuie să aibă anumite caracteristici. Singurele instrumente cu adevărat dechise sunt limbajele de programare.

d. Sisteme pentru comerțul electronic. Ca urmare a extinderii Internet-ului, se manifestă un interes deosebit pentru o serie de noi aplicații dintre care comerțul electronic ocupă un loc însemnat.
   Comerțul electronic presupune o metodologie modernă care se adresează folosirii tehnologiei informației ca un potențial esențial al afacerii. În contextul în care Internet-ul poate fi privit ca un univers informațional, comerțul electronic apare ca un nou mare orizont şi trebuie doar să se □tie modalită□ile de valorificare deplină a acestui potențial.
   Deşi comerțul electronic este mai mult ca niciodată o problemă de afacere strategică, viitorul afacerii va depinde de abilitatea de a folosi interschimbul electronic de date.

Sintetizând se poate spune că *tendințele în dezvoltarea software* sunt:

1. Dezvoltarea de aplicații, pachete integrate ieftine şi care se pot utiliza în mai multe domenii.
2. Pachete software care utilizează resursele de rețea, permit conlucrarea la aceleaşi documente.
3. Integrarea software-ului cu Internet-ul.
4. Programe uşor de utilizat ce folosesc tehnologii obiectuale orientate grafic, inteligența artificială în scopul de a utiliza limbajul natural pentru a uşura programarea.
5. Dezvoltarea de experți-asistenți în cadrul sistemelor expert ce înglobează inteligența artificială.

3.1. Evoluția sistemelor informaționale în cadrul firmelor

În ultimele cinci decenii s-au înregistrat schimbări semnificative (tabelul nr. 1) în configurația structurilor informaționale ale firmelor care au devenit în ultima perioadă de timp mai informatizate.[Nistor&alt, 2003]

Până în anii 1960, rolul sistemelor informaționale era simplu: utilizarea lor pentru înregistrarea operațiunilor legate de funcțiuni critice şi de procesare a tranzacțiilor. La sfârşitul anilor 1960 apare conceptul de sistem informațional managerial, care se va transforma cu timpul în sistem informatic managerial. Rolul acestuia era de a furniza rapoarte predefinite managerilor, rapoarte care conțineau informațiile de care aveau nevoie în fundamentarea deciziilor.

Tabelul nr.1

Funcții	Atribuții tehnice	Control managerial	Inima acțiunilor instituționale
Perioada	1950	1960-1970	1980-1990

În perioada 1970-1980 s-a remarcat că rapoartele predefinite furnizate de sistemele informaționale manageriale (**M**anagement **I**nformation **S**ystem – **MIS**) erau insuficiente managerilor pentru luarea deciziilor, în condițiile în care au intervenit schimbări semnificative în condițiile de mediu, inclusiv la nivelul piețelor de referință. Astfel a apărut conceptul de sisteme suport a deciziilor (**D**ecision **S**uport **S**ystems – **DSS**). Aceste sisteme ofereau utilizatorilor finali, în special managerilor, un suport interactiv în procesul de luare a deciziilor. În plus, aceste sisteme erau mai bine adaptate diferitelor stiluri manageriale.

În perioada 1980-1990 s-au evidențiat noi roluri ale sistemelor informaționale. Această perioadă este caracterizată de dezvoltarea microcalculatoarelor, a pachetelor de aplicații software, şi a rețelelor de telecomunicații. Acum, utilizatorii finali pot folosi resursele informaționale ca suport necesar îndeplinirii sarcinilor, în loc să aştepte un suport indirect al serviciilor informatice din cadrul departamentelor.

S-a constatat că majoritatea managerilor nu utilizau în mod direct nici rapoartele furnizate de sistemele informaționale manageriale (MIS), nici modelele analitice de suport a deciziilor furnizate de sistemele suport a deciziilor (DSS). Astfel a apărut conceptul de sisteme informatice executive (**E**xecutiv **I**nformation **S**ystem – **EIS**). Aceste sisteme informatice furnizează managerilor informațiile critice exact cum le doresc aceştia şi în formatul pe care îl preferă.

Tot în această perioadă şi în special la sfârşitul anilor 1980 au apărut şi s-au dezvoltat aplicații ale inteligenței artificiale în procesele de afaceri şi astfel au apărut sistemele expert pentru afaceri (**B**usiness **E**xpert **S**ystem – **BES**). Astăzi, sistemele expert joacă rolul de consultanți în problemele de natură economică a oricărei firme.

Ultimul deceniu al mileniului este caracterizat de dezvoltarea sistemelor informatice strategice (**S**trategic **I**nformation **S**ystem – **SIS**). Acestea joacă un rol direct în obținerea avantajului competitiv al unei firme. Modelul lui Michael Porter a stat la baza creării a numeroase astfel de sisteme informatice strategice. Firmele de astăzi trebuie să se bazeze pe tehnologia informației pentru a fi competitive pe piețe puternic concurențiale.

În cadrul unei firme se pot întâlni şase tipuri de sisteme informatice structurate pe 4 nivele:

1. Nivelul strategic – acestui nivel îi corespunde sistemul informatic executiv, EIS.
2. Nivelul tactic – la acest nivel se regăsesc sistemul suport al deciziilor, DSS, şi sistemul informatic managerial, MIS.

3. Nivelul de cunoştinţe – sistemul bazat pe cunoştinţe, **K**nowledge **B**ased **S**ystem – **KBS**.
4. Nivelul operaţional – se regăsesc sistemul de automatizare a activităţilor din birouri (**O**ffice **A**utomation **S**ystem – **OAS**) şi sistemul de procesare a tranzacţiilor (**T**ransaction **P**rocessing **S**ystem – **TPS**).

3.2. Sisteme integrate pentru firme

Schimbările tot mai rapide în mediul de afaceri şi cresterea complexităţii activităţilor din cadrul unei firme necesită o adaptare permanentă, într-un ritm alert, care adeseori pune la încercare capacităţile de efort şi analiză ale factorului uman. Sistemele / aplicaţiile integrate au fost create ca soluţii la aceste provocări, fiind capabile să proceseze volume mari de date şi informaţii agregate în scopul optimizării şi eficientizării proceselor.

Un sistem integrat de aplicaţii pentru întreprinderi este o soluţie software complexă, de nivel înalt, multi-modulară, ale cărei elemente sunt integrate într-o platformă comună, care oferă suport pentru gestiunea resurselor şi coordonarea diferitelor procese dintr-o firmă în vederea realizării obiectivelor de afaceri. Soluţia caută să modernizeze, să integreze procesele economice, să sincronizeze funcţiile întreprinderii şi să coordoneze alocarea resurselor. De asemenea, virtual, permite extinderea firmei dincolo de limitele sale fizice: spre furnizori, clienţi şi parteneri.

În practică aceste sisteme de aplicaţii se regăsesc sub diferite denumiri: ERP, SCM, CRM, PLM, BI etc.

În continuare sunt descrise o serie de tipuri mai importante de aplicaţii:

- **ERP** (**E**nterprise **R**esource **P**lanninng) - în sens restrâns se referă la aplicaţiile pentru planificarea şi urmărirea proceselor de producţie, cu luarea în considerare a materialelor, proceselor tehnologice şi resurselor (maşini, utilaje) disponibile. Actualmente, noţiunea de ERP este adesea utilizată în sens larg, pentru a desemna sistemele integrate pentru întreprinderi. În acest sens, un sistem ERP poate include ca module celelalte tipuri de aplicaţii menţionate.
- **SCM** (**S**upply **C**hain **M**anagement) - se referă la gestiunea lanţului de aprovizionare din punct de vedere al planificării cantităţilor de produse transferate şi stocate între "verigile" unui astfel de lanţ - furnizori de materii prime, producători, distribuitori şi clienţi. Se urmăresc astfel optimizări pentru întregul lanţ, în locul optimizărilor locale ale fiecărei întreprinderi. Funcţionarea unui astfel de lanţ de aprovizionare presupune relaţii de parteneriat între întreprinderi, care să faciliteze integrarea soluţiilor software ale acestora, exemplul tipic fiind industria construcţiei de automobile.
- **CRM** (**C**ustomer **R**elations **M**anagement) – sunt aplicaţii care sprijină implementarea unei strategii de orientare spre client, prin gestiunea unitară a tuturor proceselor ce presupun interacţiunea cu clientul: vânzari, servicii de garanţie şi post-garanţie (callcenter / support) etc. Pot fi incluse şi procese analitice, precum analiza portofoliului de clienţi şi a comportamentului acestora, crearea campaniilor de marketing pe anumite grupuri ţintă etc.
- **BI** (**B**usiness **I**ntelligence) – sunt aplicaţii destinate proceselor de decizie de nivel înalt ale întreprinderii. Aplicaţiile de acest tip realizează colectarea şi agregarea datelor tranzacţionale (referitoare la derularea proceselor întreprinderii), pe baza unui sistem de indicatori de performanţă. Bazate pe tehnologii de tip Data Warehouse/OLAP, aceste aplicatii permit analize detaliate ale datelor pentru identificarea tendinţelor de evoluţie şi a cauzelor acestora. Prin instrumente vizuale panou de bord, se pot urmari performantele întreprinderii, cuantificate prin indicatorii de performanţă, fiind indicate şi abaterile acestora de la valorile planificate.
- **PLM** (**P**roduct **L**ifecycle **M**anagement) – se referă la procesele de gestiune a ciclului de viaţă al produselor şi serviciilor, trecând prin etapele de concepţie, proiectare, producţie, service şi retragere. Prin gestiunea unitară a datelor despre produse se urmăreşte accelerarea lansării produselor (time-to-market), îmbunătaţirea calităţii şi scăderea costurilor.

3.2.1. Enterprise Resource Planning

Scopul sistemelor ERP

Sistemul de gestiune integrată a proceselor de afaceri are drept scop integrarea şi ordonarea informaţiei, realizarea unei mai bune comunicări de date/informaţii în firmă, prin fluidizarea schimbului de date între departamente, îmbunătăţirea cooperării şi interac□iunii dintre diverse compartimente şi asistarea procesului de management de nivel superior.

Functionalităţile unui ERP

Modulele unui ERP acoperă mai multe domenii de interes ale unei afaceri:

- Financiar-contabil.
- Mijloace fixe.
- Planificarea producţiei.
- Gestiunea stocurilor.
- Gestiunea achiziţiilor.
- Gestiunea relaţiilor cu clienţii.
- Gestiunea resurselor umane.
- Managementul calităţii.
- Managementul proiectelor.
- Managementul lanţurilor de aprovizionare.
- Managementul ciclului de viaţă al produselor.
- Analiză şi suport decizional.

Caracteristicile unui ERP

Fiind o soluţie software integrată, un sistem ERP trebuie să aibă câteva caracteristici cheie:

- **Adaptabilitate** - Funcţionalitatea standard a aplicaţiilor poate fi modificată conform cerinţelor specifice ale întreprinderii utilizatoare, de regulă în procesul de implementare a unui ERP. Realizarea adaptărilor este posibilă prin configurare (parametrizare, setare, *customizing*) care presupune modificarea unor structuri de date sau parametri de sistem; modificări mai importante pot necesita modificarea / dezvoltarea codului aplicaţiei. În al doilea caz apare dezavantajul ca procesul de actualizare (*update*) devine mai complicat, deoarece modificările de cod trebuiesc refacute după aplicarea pachetelor de actualizare ale producatorului.
- **Generalitate** – să poată cuprinde şi să satisfacă cele mai multe tipuri de organizaţii, să suporte mai multe funcţii organizaţionale; generalitatea şi puterea de a sustine organizaţii mari şi complexe coexista cu flexibilitatea specifica organizaţiilor mici. Funcţionalitatea generală se referă la informaţizarea proceselor aplicabile oricărei organizaţii, de ex. contabilitate, resurse umane, aprovizionare etc. Oferta de aplicaţii de întreprindere cuprinde şi soluţii specifice anumitor domenii de activitate, cum ar fi sănătate, telecomunicaţii, comerţ cu amănuntul (retail), industrie aeronautică, administraţie publică etc.
- **Modularitate** - să aibă o structură modulară şi orice modul să poată fi inclus sau detaşat de câte ori este nevoie fără a afecta celelalte module sau întreaga structură; toate modulele sistemului sunt strâns legate între ele toţi utilizatorii lucrând simultan asupra aceloraşi date, prin reţeaua de calculatoare, în funcţie de atribuţiile pe care le au.
- **Sistem deschis** - să aibă o arhitectura de sistem deschis, să ofere facilitate care să permită integrarea cu alte aplicaţii deja existente şi/sau tranziţia cu eforturi şi costuri minime de la alte module ale aplicaţiei; să ofere un mediu de dezvoltare şi documentare pentru utilizatori avansaţi care preiau părţi din întreţinerea, adaptarea, extinderea sistemului, dar şi integrarea cu platforme şi tehnologii de ultimă oră cum sunt Web/Intranet/Internet/Data Warehouse.
- **Interfaţă cu utilizatorul standardizată** - Modul de operare este unitar, prin standardizarea design-ului formularelor, a meniurilor aplicaţiilor şi a regulilor de operare. În acest mod este facilitat procesul de învăţare pentru utilizatori.

- **Securitatea datelor** - să permită accesul la date în condiţii de securitate deosebite numai în conformitate cu drepturile acordate fiecărui utilizator; siguranţă şi securitatea datelor sunt asigurate la un nivel deosebit prin sistemul de drepturi de acces.
- **Conectivitate** - să nu fie limitat la graniţele organizaţionale ale companiei ci să suporte conectivităţi cu alte module de afaceri din alte companii.
- **Simularea realitatii** – să permită simularea proceselor de afaceri reale şi să poată atribui responsabilităţi utilizatorilor care controlează sistemul.

Pe lângă aceste caracteristici generale trebuie remarcat faptul că un ERP trebuie să aibă o colecţie a celor mai bune procese de afaceri şi practici de afaceri, pe care să le ofere utilizatorilor.

Avantaje şi limite ale utilizarii sistemelor ERP

Avantaje:

- asigură informaţii on-line şi în timp real pentru toate ariile funcţionale ale unei companii;
- informaţia este introdusă în sistem o singură dată într-o singură bază de date ceea ce asigură acurateţea şi standardizarea datelor şi elimină redundanţele;
- îmbunătăţeşte accesul la date în vederea luării deciziilor în timp util pentru a susţine deciziile de afaceri;
- diminuarea timpilor de răspuns către client dar şi la operaţiile de afaceri realizate - furnizează instrumente de raportare managerială divesificate ceea ce îmbunătăţeşte controlul proceselor de afaceri de către conducerea companiei;
- îmbunătăţeşte procesele de afaceri deoarece obligă la utilizarea “celor mai bune practici” ce sunt incluse în aplicaţii;
- facilitează companiilor mari să acopere toate domeniile funcţionale;
- asistă activităţile cele mai importante din companie şi constituie, din acest motiv, o soluţie din cele mai bune pentru management;
- conduce la integrarea completă a aplicaţiilor nu numai între departamentele unei companii ci şi între mai multe companii;
- elimină cele mai multe probleme ale unei afaceri: criza materiilor prime, sporirea producţivităţii, livrarea promptă, calitatea produselor etc.;
- reduce golurile de informaţii din organizaţie şi oferă un flux al informaţiilor sigur şi fără redundante facilitează introducerea celor mai noi tehnologii (Internet, Intranet, Videoconferinţe, E-Commerce, EFT - Electronic Fund Transfer, EDI - Electronic Data Interchange etc.);
- oferă o perspective globală asupra a ceea ce se întâmplă în cadrul structurii organizaţionale, asupra cerinţelor actuale ale companiei dar şi oportunităţi pentru îmbunătăţirea continuă şi rafinarea proceselor de afaceri;
- asigură companiei avantaje competiţionale şi îmbunătăţeşte imaginea companiei;
- îmbunătăţeşte serviciile către clienţi şi prin aceasta se îmbunătăţeşte imagine companiei.

Limite ale utilizarii ERP:

- durata relativ mare a implementării;
- costurile implementării: cheltuiala pentru achiziţionare, costuri suplimentare/ascunse (instruire, inegrare, testare, întreţinere, adaptare, conversia datelor din sisteme vechi, consultanţă);
- amplificarea problemelor de securitate.

3.2.2. Supply Chain Management

În anii 1990 câţiva autori au încercat să pună esenţa SCM într-o singură definiţie. Ea conţine:

- obiectivele teoriei de management;
- grupul ţintă;
- obiectiv-ul (ele);
- mijloacele pentru atingerea obiectivelor.

Obiectivul SCM este în mod evident "suplly chaine", care reprezintă o rețea de organizații ce sunt implicate prin legături amonte şi aval, în diferite procese şi activități care produc valori sub forma produselor şi serviciilor ce ajung la consumatorii finali.

Într-un sens mai larg un SCM constă din două sau mai multe organizații separate legal, dar unite prin fluxuri de materiale, financiare şi informații. Aceste organizații pot fi firme ce produc părți componente şi produse finite, firme ce asigură logistica şi chiar clientul final însuşi.

O rețea uzuală nu se concentrează numai pe fluxurile dintr-un singur lanț, ci există fluxuri de lucru divergente şi convergente într-o rețea complexă ce rezultă din mai multe ordine date de diferi clienți care trebuie realizate (deservite) în paralel. Pentru a uşura complexitatea o organizație dată se poate concentra numai pe o parte din SCM general. Ca un exemplu privind în aval imaginea unei organizații poate fi limitată de clienții clienților ei, iar în amonte de furnizorii furnizorilor ei.

Într-un sens restrâns termenul de SCM este de asemenea aplicat unei companii mari care are câteva sedii adesea localizate în țări diferite. Coordonarea fluxurilor de materiale, informații şi financiare într-o astfel de companie multinațională într-un mod eficient rămâne o sarcină formidabilă. Oricum luarea deciziilor trebuie să fie uşoară, deoarece aceste puncte fac parte dintr-o singură organizație mare cu un singur nivel de management. Un SCM în acest sens este numit şi SCM interorganizațional, în timp ce termenul intraorganizațional se referă la SCM. În sens restrâns, fără a ține seama de aceste diferențe între diferite unități funcționale ca marketing, producție, aprovizionare, logistică şi finanțe este necesară o strânsă colaborare, acesta fiind condiția esențială şi firească în firmele de astăzi.

Obiectivul care guvernează toate eforturile într-un SCM este creşterea competitivității. Aceasta deoarece în ochii consumatorului final nu este responsabilă o singură unitate organizatorică pentru competivitatea produselor şi serviciilor ei, ci SCM ca un întreg. Deci competitivitatea a fost luată şi ridicată de la o singură companie la un SCM. Evident convingerea unei companiei individuală să devină parte dintr-un SCM necesită o situație de succes pentru fiecare participant, la un drum lung de cursă lungă, în timp ce aceasta nu se poate implica pentru toate părțile într-o cursă scurtă. Un impediment general acceptat în îmbunătățirea complexității este asigurarea unui service superior clienților.

Ca alternativă o firmă poate să-şi crească competivitatea prin asigurarea unui service general, acceptat la un cost minim. Sunt două căii principale de îmbunătățire:

- una este o mai apropiată integrare a organizațiilor şi închiderea integrală a organizațiilor insolvabile,
- a doua este o mai bună coordonare a fluxurilor materiale, informaționale şi financiare.

Depăşind barierele organizaționale, alinierea strategiilor şi gestionarea lanțului de aprovizionare sunt subiecte comune în acest sens. Se va defini termenul de management al lanțului de aprovizionare ca fiind sarcina de a integra unitățile organizaționale de-a lungul unui lanț de aprovizionare, de a coordona materialele, informațiile şi fluxurile financiare pentru a acoperi cererile clientului, cu scopul de a îmbunătăți competitivitatea lanțului de aprovizionare ca un tot.

3.2.3. Customer Relationship Management

CRM-ul este un proces care ajută la o mai bună înțelegere a nevoilor şi comportamentului clienților. El ajută la dezvoltarea şi la trasarea strategiilor de afaceri pentru client şi de asemenea abordează problemele prin implementarea lor. În termeni simpli utilizarea eficientă a unui CRM ajută clientul şi organizația acestuia să îndeplinească acele țeluri mult dorite, acele ținte referitoare la potențialele performanțe, achiziții, creşterea şi păstrarea afacerii. O bază de date centralizată furnizează informațiile despre clienți şi asigură coordonarea echipelor de marketing, vânzări şi suport.

Un sistem eficient de CRM are drept scop gestionarea şi optimizarea ciclului de viaţă al clientului dar şi construirea unei înţelegeri corespunzătoare sau a unei legături între diverse departamente, forţe de vânzări şi clienţi în cadrul unor companii mari, care în final toate la un loc vor spori producţia companiei.

CRM (Managementul Relaţiei cu Clienţii) este o strategie comercială care urmăreşte stabilirea unei relaţii profitabile şi de lungă durată cu clienţii. Astăzi, CRM este introdus în majoritatea companiilor mari. Managementul informaţiei este susţinut şi cunoştinţele companiei sunt păstrate.

Prin folosirea CRM-ului, o organizaţie poate folosi înregistrările anterioare ale potenţialilor clienţi şi cumpărători precum şi tendinţele anterioare de achizitie, tendinţe care le-au preferat mai devreme dar şi nevoile acestora, pentru a crea un centru al clienţilor şi un sistem de marketing central referitor la nevoile clienţilor.

Companiile care utilizează sisteme CRM se pot aştepta imediat:

- la păstrarea şi obţinerea mai multor clienţi, în consecinţă la o creştere globală;
- la creşterea şi maximizarea absolută a ciclurilor de viaţă a clienţilor;
- la îmbunătăţirea serviciilor pentru clienţi prin personalizarea acestora;

Succesul unei aplicaţii CRM constă în continua dezvoltare a acestuia, pentru a satisface noile nevoi şi cerinţe ale utilizatorilor acestuia. Este foarte important ca utilizatorul de CRM să înteleagă că succesul unei astfel de aplicaţii constă în continua dezvoltare şi modelare a funcţiilor aplicaţiei după nevoile sale.

3.2.4. Business Intelligence

Luarea deciziilor bune în afacerile care sunt făcute este la fel de important ca şi în viaţa particulară. În fiecare zi trebuie să se ia decizii care să determine direcţia şi eficienţa activităţilor dintr-o organizaţie. Se iau decizii în legatură cu producţia, marketingul, personalul etc. Deciziile luate afectează costurile, vânzările, profitul etc. Ca şi în viaţa personală, cheia pentru succesul organizaţiei este în modul în care se ia deciziilor. Organizaţiile trebuie să ia decizii eficiente.

Cine ia deciziile?

La prima vedere, se poate spune ca doar persoanele din varful ierarhiei (CEO, director, preşedinte) trebuie să ia decizii eficiente care să aducă succesul în organizaţie. Planurile eficiente dezvoltate de conducerea organizaţiei pot eşua datorită deciziilor greşite luate de către persoane din părţile inferioare ale ierarhiei în procesul de implementare sau în cel de executie. În concluzie, decizii eficiente trebuie luate de toţi cei din organizaţie. Deciziile eficiente luate la fiecare nivel al organizaţiei duc către succes.

Ce sunt deciziile eficiente?

Deciziile eficiente sunt acele alegeri care duc organizaţia mai aproape de obiectivele stabilite în timp util. Dacă se ia în consideraţie definiţia de mai sus se poate observa trei componente importante care ajută personalul de decizie din cadrul organiza□iei să ia decizii eficiente:

- stabilirea obiectivelor;
- stabilirea unor mărimi de măsurat pentru stabilirea abaterii de la obiective;
- stabilirea termenelor în care trebuie atinse obiectivele.

Aceste informaţii reprezintă baza de plecare pentru luarea deciziilor, dar şi o metoda de evaluare a calitaţii deciziilor luate. Obiectivele trebuie să fie stabilite clar şi făcute cunoscute tuturor celor implicaţi în activitatea organizaţiei. Pentru ca un obiectiv să poată sta la baza unei decizii eficiente trebuie să definească şi o metodă de măsurare pentru stabilirea în orice moment a abaterii înregistrate în activitatea curentă.

Odată stabilite obiectivele clare, metodele de măsurare ale acestor obiective şi metodele de evaluare, se pune întrebarea: Cum poate o organizație să obțină şi să distribuie informațiile necesare pentru luarea deciziilor şi evaluarea eficienței acestora? Răspunsul la această întrebare este: Prin soluții de Business Intelligence.

Putem defini Business Intelligence *drept platformă de prezentare a informațiilor într-un mod corect, util şi specific către fiecare persoană de decizie în timp util pentru a putea servi în luare deciziilor eficiente*.

Business Intelligence nu reprezintă un set de rapoarte tipărite sau prezentate pe ecran. Rândurile unui raport de vânzări, spre exemplu, pot conține informații detaliate şi exacte, dar nu reprezintă o soluție de Business Intelligence până când nu sunt puse într-un format care poate fi uşor înțeles şi interpretat de către o persoană de decizie în vederea stabilirii unei soluții eficiente pentru o situație particular întâlnită în activitatea curentă.

3.3. Sistemul informatic de gestiune

Sistemul informatic de gestiune *este sistemul informatic folosit pentru administrarea □i controlul resurselor unui sistem economic*, deoarece termenul de gestiune are semnifica□ia, pe de o parte, de „administrare a bunurilor unei întreprinderi“ □i, pe de altă parte, de „răspundere a păstrării bunurilor □i amânuirii fondurilor unei întreprinderi“ ale cărei bunuri sunt reprezentate de resursele necesare pentru realizarea obiectivelor sale, denumită generic sistem economic. [Andronie, 2007]

Administrarea □i controlul resurselor unui sistem economic impun administrarea □i controlul activită□ilor pe care le desfă□oară sistemul economic respectiv pentru îndeplinirea obiectivelor sale.

Capitolul 4
Metodologii de realizare a sistemelor informatice

Procedeele pentru definirea, organizarea practică şi etapizarea activităților analizei de sistem constituie aşa numitele **metodologii pentru analiza şi proiectarea sistemelor** şi reprezintă obiectul preocupării unui număr mare de specialişti din întreaga lume.

Metodele şi conceptele fundamentale de realizare a sistemelor informatice pentru un organism oarecare au la bază termenii de *metodă* şi *metodologie* asociați direct domeniului respectiv. Aceşti termeni vor fi actualizați în raport cu specificul domeniului analizat, evoluția cadrului legislativ, dinamica activității de informatică şi tendințele fundamentale de evoluție a sistemului analizat. Pentru realizarea sistemelor informatice se utilizează următoarele concepte:

- **Metoda**: reprezintă modul unitar sau maniera comună în care analiştii de sistem, programatorii şi alte categorii de persoane implicate, realizează procesul de analiză a sistemului informațional – decizional, proiectarea şi introducerea sistemului informatic; *sau* reprezintă totalitatea conceptelor, modelelor, nivelelor, etapelor şi tehnicilor specifice de formalizare necesare pentru definițiile, listele, comunicațiile, datele şi prelucrările specifice unui sistem informatic.
- **Metodologia**: este reprezentată de setul finit particular definitoriu al unei metode prin intermediul unui sistem coerent de formulare şi/sau procese informatice necesare pentru modelarea şi formalizarea totală a unui sistem informatic.
- **Tehnici de lucru**: reprezintă felul în care se acționează eficient şi rapid, în cadrul unei metode, pentru soluționarea diferitelor probleme ce apar în procesul de proiectare.
- **Instrumentele**: utilizate în proiectarea sistemului informatic, sunt acele mijloace care se utilizează de către echipă pentru realizarea scopului propus, ele depind de metodele şi tehnicile utilizate, precum şi de procedurile analizate şi proiectate.
- **Procedee de lucru** (procedură): succesiunea operațiilor necesare parcurgerii unor etape ale acțiunii şi aplicării unor tehnici în cadrul metodelor în conformitate cu o rutină de lucru dată.

Este bine cunoscut faptul că preocupările pentru analiza şi proiectarea sistemelor informatice au apărut ca rezultat al aplicării unora dintre ideile analizei sistemice în tehnica folosirii echipamentelor moderne de culegere, prelucrare şi stocare a informației. De aici decurge faptul ca metodologiile de analiză şi proiectare a sistemelor informatice trebuie să fie axate în primul rând pe utilizarea eficientă a calculatoarelor electronice, pe valorificarea posibilităților şi facilităților pe care le oferă acestea proiectanților de sisteme şi în final utilizatorilor. Pe de altă parte, metodologiile de analiză şi proiectare a sistemelor informatice tind să integreze calculatorul în însuşi procesul de analiză. Sintetizând cele prezentate, se poate aprecia că:

- O primă caracteristică a metodologiilor de analiză şi proiectare a sistemelor informatice este orientarea spre soluții care prevăd organizarea centralizată a informației în unitățile care fac obiectul analizei.
- O a doua caracteristică a metodologiilor de analiză şi proiectare a sistemelor informatice derivă din posibilitățile remarcabile de prelucrare rapidă a informațiilor pe care le oferă calculatoarele.

Metodologiile de analiză şi proiectare a sistemelor informatice sunt numeroase iar alegerea uneia sau alteia în cadrul proiectării unui sistem informatic este o opțiune destul de grea şi care, de obicei, ține cont de pregătirea specialistului în utilizarea şi cunoaşterea cât mai multor metode.

În funcție de modul de abordare şi domeniul de aplicabilitate, metodologiile utilizate sunt: [Lungu1&alt, 1995]

1. Din domeniul gestiunii: AXIAL, al firmei IBM; MERISE, Ministerul Industriei din Franța; IE, James Martin; DA, Universitatea Michigam; SSADM, Marea Britanie.
2. Orientate obiect, se bazează în formalizarea sistemelor pe noțiunea de obiect (entitate pentru date şi acțiune pentru procese): OMT, General Electric SUA; OOD, SUA; ISD, Michael Jackson.
3. Pentru conducerea proiectelor de sisteme informatice, se utilizează pentru elaborarea sistemelor într-o concepție ştiințifică: SDM/S; METHOD/1; NAVIGATOR (Ernest /Zoung – James Martin).

O altă clasificare a metodologiilor de realizare a sistemelor informatice, care este cu precădere utilizată în prezent este:

1. Metodologii cadru, cu scop orientativ general.
2. Metodologii proprii unor firme producătoare de echipamente de calcul sau sisteme informatice.

Metodologiile de realizare a sistemelor informatice cuprind:

1. Modalitatea de abordare a sistemelor.
2. Regulile de formalizare a datelor şi a proceselor de prelucrare.
3. Instrumente pentru concepția, realizarea şi elaborarea documentației.
4. Modalitatea de derulare a proiectului şi acțiunile specifice fiecărei etape (ciclul de viață).
5. Definirea modului de lucru.
6. Modalități de administrare a proiectului.

Tendințele actuale în evoluția metodologiilor de proiectare sunt:

1. Înlăturarea monopolul deținut de-a lungul unei perioade de timp de anumite metodologii, precum MERISE în Franța şi SSADM în Marea Britanie.
2. Îmbogățirea reciprocă a metodologiilor existente.
3. Asimilarea tendințelor firești de evoluție a informaticii (arhitectura client/server, abordarea pe obiecte, modelul relațional pentru date).
4. Tendința spre standardizare:
   - modelul entitate-relație pentru date;
   - diagrama de flux pentru aspectele funcționale;
   - modele de prelucrare şi de comunicație pentru aspectele „dinamice”.
5. Integrarea metodologiilor existente, a limbajelor de modelare şi a instrumentelor în cadrul unor procese de dezvoltarea software care permit modelarea mai flexibilă a sistemului.

Pe parcursul timpului aceste metode, mijloace şi tehnici specifice analizei de sistem au evoluat în mod considerabil, şi ar fi imposibil ca să se poată surprinde toate aceste metode care au existat şi care există în momentul de față. Din acest motiv se vor enumera cele mai importante dintre metodologiile existente, şi se va insista pe una din metodologiile de dată recentă, metodologia UML.

4.1. Etape în proiectarea unui sistem informatic

Realizarea unui sistem informatic implică mai multe etape teoretice. Acestea pot fi sistematizate astfel:

- stabilirea cerințelor utilizatorului;
- analiza şi specificarea produsului;
- proiectarea generală şi de detaliu;
- implementarea codului;
- testarea produsului;
- instalarea şi mentenanța produsului.

În practică, majoritatea analiştilor de sistem au tendința de a parcurge doar anumite etape, dacă nu chiar una singură, şi anume aceea de scriere de cod. În general acest lucru este explicat prin lipsa de

timp şi în cele mai multe cazuri, de incapacitatea beneficiarului de a stabili şi formula încă de la început corect cerinţele produsului software.

Este însă adevărat că nu orice produs software trebuie să parcurgă toate etapele teoretice enumerate. Dar tot atât de adevărat este şi faptul că, parcurgerea acestor etape, poate duce la rezultate superioare şi la eliminarea unor erori de proiectare şi programare încă din fazele iniţiale ale proiectului. Ceea ce este mai dăunător este faptul că de obicei se renunţă la etapele de analiză şi proiectare ale produsului, care sunt etape cheie în dezvoltarea aplicaţiei, ele furnizând o schematizare şi o vedere de ansamblu, care duc la o înţelegere mai profundă a problemei abordate.

Proiectarea unei aplicaţii software complexe, de dimensiuni mari, dezvoltată în cadrul unei echipe de software, nu se face trecând direct la implementarea programului, ci trebuie să urmeze etapele enumerate mai sus. După cum se observă, proiectarea nu se termină odată cu implementarea, pentru că sistemul creat trebuie testat şi apoi instalat. După instalarea produsului software acesta trebuie întreţinut, operaţie care trebuie realizată continuu.

Ceea ce s-a descris mai sus constituie unul din cele trei cicluri ale metodologiilor utilizate în analiza şi proiectarea sistemului, şi anume **ciclul de viaţă** al unui sistem informatic – o metodologie comună de dezvoltare a sistemelor, caracterizată prin mai multe faze care marchează evoluţia eforturilor de analiză şi proiectare a sistemelor (Hoffer), şi care constituie obiectul de studiu al ingineriei programării (este un domeniu care încearcă o abordare sistematică a dezvoltării, operării, întreţinerii şi retragerii unui program de pe piaţă).

În general în cadrul metodologiilor se pot distinge trei cicluri majore:

- Ciclul de viaţă al unui sistem (recursivitate).
- Ciclul de abstractizare (sau ciclul de modelare, raţionamentul).
- Ciclul de decizie (sau de validare, conducere).

Din punct de vedere al unei metodologii utilizate, sistemele informatice au un ciclu de viaţă propriu, care începe cu decizia de realizare a sistemului informatic, cuprinde faza de elaborare, faza de utilizare şi faza de perfecţionare şi se încheie cu decizia de abandonare a sa în forma existentă şi înlocuirea lui cu un nou sistem.

Numărul acestor etape variază de la trei (analiză, proiectare, implementare) la peste douăzeci. Odată cu apariţia abordării orientate obiect etapele s-au diversificat. Totodată, după apariţia tehnicilor de dezvoltare rapidă a aplicaţiilor (mediile **RAD** – **R**apid **A**pplication **D**evelopment), numărul etapelor s-a redus din nou. Cu toate acestea, fiecare metodă realizează dezvoltarea unui sistem într-o succesiune de etape, sau faze, care diferă de la un model la altul.

Ceea ce este important de reţinut este faptul că indiferent de ce subsistem se doreşte a fi implementat, în realizarea acestuia se utilizează o anumită metodă.

Cu toate acestea majoritatea modelelor prezintă anumite etape tipice, care sunt:

1. Studiul şi analiza sistemului existent. Această activitate include stabilirea caracteristicilor generale, tehnico – economice, ale unităţii analizate, cu precădere a funcţiei de bază, precum şi studiul sistemului informaţional pentru conducere existent, ceea ce înseamnă identificarea caracteristicilor acestui sistem, a fluxului informaţional şi a metodelor şi mijloacelor tehnice de prelucrare a datelor.
2. Conceperea sistemului informatic, obiectul acestei etape îl constituie elaborarea proiectului de ansamblu al sistemului informatic, pe baza cerinţelor şi restricţiilor formulate în prima etapă.
3. Proiectarea de detaliu: obiectivul principal îl reprezintă elaborarea componentelor funcţionale detaliate ale sistemului şi a specificaţiilor de definire a produselor – program, în scopul acceptării lor de către beneficiar şi întocmirii suportului documentaţiei de prezentare, utilizare şi exploatare.

4. Elaborarea programelor. Această etapă este consacrată elaborării produselor – program şi procedurilor manuale aferente componentelor sistemului informatic, precum şi testării componentelor.
5. Implementare sistemului, obiectivul principal al acestei etape este lansarea în exploatare curentă a sistemului.
6. Exploatarea, întreținerea şi dezvoltarea sistemului informatic.

Fiecare din aceste etape au la rândul lor mai multe activități specifice. Astfel etapa de studiul şi analiza sistemului existent conține următoarele activități:

- studiul sistemului existent;
- evaluarea sistemului existent;
- evaluarea gradului de pregătire a unității economice;
- formularea cerințelor şi a restricțiilor pentru realizarea sistemului informatic.

Conform raportului Euromethod etapele care trebuie parcurse în dezvoltarea sistemelor informatice sunt:

- Planificarea strategică – cuprinde informații referitoare la organizația pentru care se elaborează sistemul informatic, cât şi la sistemul însuşi.
- Studiul de fezabilitate – are rolul de a asigura informațiile obiective necesare pentru a stabili dacă un proiect poate fi demarat sau nu. Dimensiunile şi durata studiilor de fezabilitate variază în funcție de mărimea şi de natura sistemului de implementat.
- Definirea cerințelor – variază în funcție de metoda utilizată. În general, funcțiile acestei etape sunt: înțelegerea scopului şi funcțiile sistemului, specificarea funcțiilor din perspectiva utilizatorului.
- Proiectarea – cuprinde la rândul său proiectarea logică şi proiectarea tehnică.
- Codificarea – reprezintă activitatea de transformare a algoritmilor definiți în etapa de proiectare într-un limbaj de programare.
- Implementarea – este etapa în care are loc procesul de instalare a echipamentelor şi a sistemului informatic (software-ului) în vederea finalizării sistemului şi dării lui în funcțiune.
- Întreținerea şi dezvoltarea – se referă la întreținerea sistemului elaborat.

4.2. Modele utilizate în proiectarea unui sistem informatic

În prezent există mai multe modele de realizare a sistemelor informatice (numite modele ale ciclului de viață ale dezvoltării sistemului), printre care pot fi menționate:

- **Modelul liniar** (în cascadă, sau Waterfall) este cel mai vechi şi cel mai cunoscut model. A fost dezvoltat în perioada anilor 1960 (a fost lansat la începutul anilor 1970 de către W.W. Royce), caracteristica principală constând în parcurgerea succesivă a unor etape, numite „faze", fiecare fază constituie o trecere de la un nivel de abstractizare ridicat la un nivel mai scăzut de abstractizare. Printre dezavantajele cele mai importante sunt următoarele: dacă s-a produs o eroare este foarte greu de revenit la faza la care s-a produs eroarea; metoda a fost dezvoltată într-o perioadă când sistemele dominante erau cele de dimensiuni mici şi cu o arhitectură compactă; nu se pretează la transpunerea să pe calculator. Fazele, la rândul lor, sunt structurate pe activități şi subactivități. Trecerea la faza următoare se realizează după parcurgerea în întregime a fazei precedente. Modelul este folosit de Booch în fazele de proiectare şi implementare din cadrul metodei OOADA (Object-Oriented Analysis and Design with Application) şi Ivar Jacobson pentru toate etapele metodei OOSE (Object-Oriented Software Engineering). Ideea de bază a modelului este regăsită şi în alte modele, cum ar fi: modelul V, modelul incremental, şi modelul fântânei arteziene.

- **Modelul incremental** (rapid-prototyping) a fost creat pentru a acoperi deficienţele modelului liniar; diferenţa principală între aceste modele constă în introducerea conceptului de dezvoltare incrementală. Conceptul de dezvoltare incrementală se referă la crearea unor mici salturi de la vechea variantă de sistem la noua variantă care conţine un plus de funcţii.
- **Modelul de concepţie în „V"**: pentru evidenţierea faptului că anumite faze ale ciclului de viaţă sunt condiţionate de faze anterioare s-a propus reprezentarea ciclului de viaţă al unui sistem informatic ca un ciclu de concepţie în „V". Acest tip de model a fost lansat în 1990 de Ould. Un mare dezavantaj al acestei metode este acela că nu se poate valida analiza făcută la începutul proiectului decât atunci când toate activităţile de programare, testare şi integrare sunt terminate.
- **Modelul operaţional** realizează specificaţii "executabile", măreşte rolul automatizării şi stabileşte o strânsă legătură între specificaţii şi implementare etc.

Atunci când se începe dezvoltarea unui sistem se poate alege unul din modelele prezentate, sau altele existente.

Problema cu cele mai multe metode este că ele au atât reguli implicite cât şi explicite. Regulile explicite sunt prezentate de obicei în documentaţia de realizare, iar cele implicite nu sunt prezentate nicăieri.

Alegerea modelului depinde de mărimea sistemelor de analizat şi dezvoltat, precum şi de domeniile cărora le aparţin. O altă latură care trebuie avută în vedere atunci când se alege o metodă sau alta este modalitatea de abordare a sistemului: în întregime sau pe părţi componente. Un alt aspect, dar nu neapărat ultimul, care trebuie abordat este "informaţizarea" metodei de proiectare. Informaţizarea reprezintă un suport pentru o metodă şi este o parte integrată în procesul de dezvoltare a sistemului informatic.

4.3. Metodologii de analiză şi proiectare a sistemelor informatice

Metodologiile de analiză şi proiectare a sistemelor informatice au apărut ca o necesitate de definire a unor paşi şi reguli generale, ce trebuie îndeplinite pentru analiza, proiectarea şi implementarea diferitelor tipuri de probleme. Pentru a nu "descoperi" de fiecare dată procedeul care a dus la succesul realizării unui produs anterior, s-a căutat definirea unor metode pragmatice de structurate şi ulterior de analiză şi proiectare a acestora. "Reutilizarea" este un cuvânt cheie în dezvoltarea aplicaţiilor din ultimii ani, iar acesta nu se referă numai la reutilizarea părţilor de cod, ci la toată experienţa acumulată pe parcursul dezvoltării produsului software.

Un factor important care a dus la apariţia conceptului de reutilizare, în special în ingineria programării, este *timpul*, termenele realizării proiectelor devenind tot mai scurte. Astfel produsul este dezvoltat "din mers", în sensul că programatorul încearcă să ghicească şi să implementeze cerinţele schematice ale clientului. Acesta urmează să formuleze cerinţele efective pornind de la prototipul furnizat de programator.

Întrebarea firească este de ce nu există o singură metodologie care să fie general valabilă şi aplicabilă în toate cazurile ? Un răspuns posibil este acela că metodologiile diferă între ele destul de mult, fiind mai eficiente în anumite domenii sau cazuri. Ele au luat naştere din generalizarea soluţiilor oferite pe cazuri particulare.

4.3.1. Metodologii de analiză sistemică

Ţinându-se cont de complexitatea celor mai multe sisteme existente în natură studierea sistemelor se face într-o manieră aparte numită abordare sistemică. Spre deosebire de abordarea carteziană, care

constă în a separa şi a izola fiecare subproblemă pentru o prelucrare ulterioară, abordarea sistemică propune o viziune unică şi globală a problemei de rezolvat.

O idee importantă care caracterizează analiza sistemică, şi implicit a sistemelor informatice, este posibilitatea perfecționării sistemelor printr-o activitate de analiză şi proiectare ştiințifică a acestora. O altă idee fundamentală, specifică analizei de sistem este caracterul său multidisciplinar. Organizarea acestor activități multidisciplinare constituie o altă idee de bază a analizei de sistem şi poate cea mai importantă dintre toate.

O problemă decisivă este alegerea metodei ce urmează a fi aplicată pentru fiecare caz în parte. Se poate afirma că, acesta este pasul cel mai important în desfăşurarea proiectului. Dacă alegerea făcută este corectă, acest fapt va duce la rezultate pozitive din toate punctele de vedere. În cazul în care s-a optat însă pentru o metodă nepotrivită problemei de rezolvat, rezultatele proiectului vor fi pe măsură (din această cauză sunt situații când la sfârşitul proiectului se constată că aplicația trebuie reproiectată şi implementată într-un mod diferit).

Alegerea aplicării metodei pentru rezolvarea unei anumite probleme depinde de mai mulți factori, printre care amintim:

- tipul problemei de rezolvat;
- domeniul în care se încadrează problema;
- pregătirea şi calificarea echipei de dezvoltare a produsului software;
- resursele hardware şi software disponibile;
- bugetul şi timpul alocat proiectului.

În legătură cu evoluția şi progresele din ultimii ani în analiza de sistem, se poate descifra două principale tendințe de aplicare ale acestei metodologii:

- *un sens restrâns*, în acest sens analiza se orientează în special spre aspecte legate de culegerea, transmiterea, prelucrarea şi stocarea informațiilor într-un sistem fără a se aprofunda procedeele ştiințifice de rezolvare a problemelor decizionale aferente muncii de conducere din sistemul supus analizei. În general această variantă se numeşte *analiza şi proiectarea sistemelor informaționale*, iar metodologiile sunt de tip informa□ional.
- *un sens mai larg*, în care se include, pe lângă preocupările enumerate mai sus, şi pe cele care vizează metodele de optimizare a problemelor de decizie. Această variantă este denumită *analiza şi proiectarea sistemelor informațional-decizionale*., iar metodologiile sunt de tip informa□ional-decizional.

Metodologii de tip informațional

Printre tehnicile de analiză sistemică din aşa numită generația a II-a, o mare utilizare au cunoscut-o grilele de analiză informațională neautomatizată, a căror sarcină era de a stabili informațiile de intrare în sistem necesare pentru obținerea anumitor documente de ieşire.

Printre primele metode de analiză informațională cu ajutorul grilei temporale automatizate, un instrument metodologic auxiliar al analistului, care au stabilit relația între intrări şi ieşiri a fost **metoda T.A.G.** (**T**ime **A**utomated **G**rid) - grila automatizată de analiză) [Bodur&alt, 1982], propusă mai întâi în 1962 ca o tehnică manuală şi apoi pusă la punct în anul 1966 de firma I.B.M., care realiza automatizarea unei părți a procesului de analiză, permițând definirea unei baze de date minimale pentru un anumit sistem, precum şi definirea unor importante operații auxiliare pentru utilizarea eficientă a respectivei baze de date. T.A.G. este definită ca o tehnică general aplicabilă pentru proiectarea sistemelor informatice în domeniul comercial; ea îşi propune să asiste pe analist chiar din faza releveului informațional, cu o finalitate multiplă: pentru colectarea şi analiza datelor, pentru sistematizarea acestora în colecții sau fişiere şi apoi pentru definirea fluxurilor informaționale prin activități.

Tehnica T.A.G. apelează la principiul regresiv: se porneşte de la ultimele 'ieşiri' ale sistemului supus cercetării (ieşiri pentru care analistul determină necesarul de informaţii şi-l înscrie pe un formular special); din acestea, T.A.G. determină necesarul de intrări pentru faza în amonte a fluxului informaţional, grupându-le după momentul lor de apariţie şi apoi, iterativ, solicită analistului date ş.a.m.d. până la primele "intrări", din afara sistemului. În acest mod T.A.G. poate să definească o bază de date minimală pentru sistemul respectiv, căutând totodată să reducă şi durata totală a fluxului informaţional.

Aceste procedee de analiză cu ajutorul grilei automatizate fac parte din tendinţa de abordare a problemelor nedecizionale dintr-un sistem, progresul faţă de metodele din generaţiile I şi a II-a constând doar în transferarea către calculator a unora dintre operaţiile care erau efectuate manual.

Preocupările de a integra calculatorul în procesul de analiză sistemică s-au soldat cu elaborarea unor procedee care reuşesc să automatizeze parţial sau total unele din activităţile de analiză. Ceea ce au în comun toate aceste procedee este faptul că ale se axează în exclusivitate asupra analizei şi proiectării sistemului informatic, asupra elaborării automate a programelor şi a corelării lor într-un sistem. Cerinţele sistemului se presupun a fi gata definite, ceea ce înseamnă că analiza şi proiectarea informaţional – decizională rămâne să fie executate manual.

Una dintre cele mai evoluate tehnici din această categorie (proiectarea sistemelor integrate) o reprezintă sistemul **I.S.D.O.S.** (**I**nformation **S**ystem **D**esign and **O**ptimization **S**ystem), elaborat de un grup de cercetători de la Universitatea din Michigan, conduşi de profesorul Daniel Teichroew.

Sistemul I.S.D.O.S. cuprinde mai multe metode care se înlănţuie pentru a prelua specificaţiile cerinţelor, a le corela, a organiza etapele proiectării sistemului informatic aferent, în scopul obţinerii cerinţelor cu ajutorul unor grafuri A.D.C.[1] şi în final a elabora fişierele sistemului.

Caracterizarea generală a acestor procedee este faptul că ele sunt necomputerizate, ceea ce nu înseamnă că nu se orientează spre promovarea unor metode de analiză logică riguroasă şi spre algoritmizarea operaţiilor componente. Ceea ce le diferenţiază radical este faptul că nu utilizează calculatorul în munca de analiză, ci momentul când este necesară folosirea lor în analiza de sistem.

Tot în jurul anilor 1960 o serie de mari firme constructoare de calculatoare elaborează metodologii care încearcă să etapizeze şi să sistematizeze operaţiile necesare proiectării unui sistem de conducere bazat pe utilizarea calculatoarelor. Cea mai reprezentativă metodă este **S.O.P.** (Study Organizaţion Plan), elaborată de firma I.B.M.

Aceste metode sunt singurele care cuprind descrierea întregii munci de analiză şi proiectare a sistemelor, cu parcurgerea tuturor etapelor şi fazelor, de la studii preliminare, până la implementarea, bineînţeles la nivelul hardware - ului şi software – ului de atunci şi fără a integra calculatorul în însăşi munca de analiză.

Aceste metodologii elaborate, în general, de diviziunile de cercetare ale marilor firme de calculatoare reprezintă o poziţie intermediară, de trecere de la metodologiile cu specific de analiză-proiectare a sistemelor pur informatice spre metodologiile A.S.I.D. – metodologii de analiză şi proiectare informaţional–decizională.

Metodologii de tip informaţional-decizională

Metodologiile de analiză şi proiectare informaţional – decizională pun la dispoziţia analistului de sisteme procedee cuprinzătoare şi eficiente în vederea perfecţionării activităţii social – economice. Printre caracteristicile metodologiilor A.S.I.D. sunt *diversitatea* şi *mobilitatea* lor; aceste metodologii

[1] A.D.C. = analiza drumului critic, metodă de conducere a acţiunilor complexe

nu formează încă un ansamblu unitar, standardizat, de reguli şi prescripţii, ci constituie elaborate – unicat, destul de deosebite unul de altul în ceea ce priveşte terminologia, cât şi în ceea ce priveşte împărţirea în etape, subetape şi faze şi chiar în privinţa modului în care aplică ideea principală care stă la baza lor – regula priorităţii aspectelor decizionale.

O altă caracteristică importantă a metodologiilor A.S.I.D. o constituie abordarea problemelor şi sectoarelor vitale ale societăţii, acelea în care problematicile decizionale sunt mai complexe şi în care, prin optimizarea fluxurilor informaţional – decizionale, se pot obţine avantaje considerabile.

Tot atât de însemnată este şi o altă trăsătură fundamentală a metodologiilor A.S.I.D., şi anume caracterul lor multidisciplinar (utilizarea unor metode şi tehnici aparţinând metodologiilor de analiză de tip informatic, la care se adaugă procedee preluate din cercetarea operaţională, dinamica industrială, tehnici de simulare etc.).

Care ar fi diferenţa dintre metodologiile de analiză – proiectare a sistemelor informatice şi cele de analiză şi proiectare a sistemelor informaţional – decizionale ? În realitate, un sistem informatic va aparţine întotdeauna unui sistem informaţional – decizional, analiza pur informatică şi de programare reprezentând o subetapă a analizei sistemului informaţional – decizional.

Mai târziu s-au elaborat metodologii de analiză şi proiectare informaţional – decizională de tip ameliorativ, care se orientează în special spre rezolvarea problemelor decizionale din sistem şi care propun diverse procedee de integrare a metodelor cercetării operaţionale pentru optimizarea deciziilor din sistemul proiectat, precum şi utilizarea psihosociologiei în însăşi munca de analiză. Metodologiile ameliorative se caracterizează prin faptul că pleacă de la sistemul informaţional – decizional real, în funcţiune, al unei societăţi sau al unei părţi din aceasta şi îşi propun îmbunătăţirea acestui sistem, preluând tot ce este bun în el, dar în acelaşi timp transformând ceea ce poate fi perfecţionat.

Deci sintetizând se poate spune că metodologiile A.S.I.D. se pot grupa în două mari categorii:

- metodologii care pornesc de la o exploatare prealabilă cât mai amănunţită a situaţiei prezente a sistemului, procedând apoi, prin îmbunătăţiri succesive, la elaborarea unui nou proiect de sistem adică metodologii ameliorative;
- metodologii ce îşi propun să construiască un proiect de sistem, pornind de la descrierea (releveul) situaţiei de fapt, cât şi de la o analiză fundamentală a obiectivelor pe care este chemat să le satisfacă sectorul supus (re)proiectării; acestea se cunosc sub numele de metodologii constructive sau aval – amonte.

Dintre cea mai cunoscută **metodă de analiză aval – amonte** este cea **a lui André Delville**, care a fost elaborată în Franţa în anul 1966. La baza metodei aval – amonte stă ideea conform căreia întreg sistemul de elaborare, circulaţie şi prelucrare a informaţiei trebuie construit de la obiectivele economice concrete ale societăţii. Cunoscând obiectivele, printr-un procedeu deductiv se stabilesc informaţiile "în aval" adică cele necesare realizării obiectivelor (informaţii care reprezintă cerinţele informaţionale ale sistemului). Apoi, tot deductiv, se determină informaţiile "în amonte", adică necesare a fi prelucrate pentru obţinerea informaţiilor în aval şi totodată procesele de prelucrare, inclusiv cele decizionale, prin care se transformă informaţiile din amonte în informaţii – aval. Procedeul se repetă din aproape în aproape, urmărind din spre aval spre amonte întreaga cascadă informaţional – decizională până se ajunge la informaţiile care provin din afara sistemului şi care reprezintă informaţiile de intrare.

În etapa următoare se aleg mijloacele tehnice şi metodele decizionale necesare şi avantajoase pentru realizarea întregii succesiuni de operaţii informaţional – decizionale stabilite prin parcurgerea cascadei aval – amonte.

Aşadar, metoda Delville se caracterizează prin analiza informațional – decizională combinată, cu ajutorul căreia se construieşte deductiv întregul sistem plecând de la necesități informaționale legate de realizarea obiectivelor economice concrete, fără a ignora sistemul informațional existent din care se poate prelua ceea ce este eficient. Metoda are în vedere utilizarea tehnicilor moderne de prelucrare automată dar nu intră în detaliile proiectării sistemului informatic.

4.3.2. Metodologii de analiză structurate

Aceste metodologii au apărut în principal din nevoia de a rezolva criza software-ului (de a sistematiza şi organiza o aplicație) apărută în anii 1970-1980. Analiza structurată a unei aplicații se bazează pe construirea unor diagrame de flux de date, aceasta fiind una dintre cele mai răspândite metodologii de inginerie a programării. Dificultățile au apărut, însă, pentru etapa de analiză structurată a unei probleme, etapă care precede programarea; respectiv, a fost dificil de precizat cum trebuiau să arate şi ce trebuiau să conțină diagramele de flux de date.

În acest sens, au existat mai multe abordări, dintre care se amintesc:

- Metodologia **SADT** (Structured Analysis and Design Technique), reprezintă o metodă grafică utilizată pentru analiza cerințelor şi proiectare sistemelor. Forma să finală a fost dezvoltată de Douglas T. Ross în anul 1974, scopul principal fiind înțelegerea şi specificarea cerințelor înaintea începerii realizării efective a produsului software.
- Metodologia **JSD** (Jackson System Development). JSD este o metodologie definită de Cameron în 1989, reprezentând o extensie a metodei JSP (Jackson Structured Programming). Metoda constă în realizarea unei corespondențe între structura problemei de rezolvat şi structura unui program. Această metodă este recomandată a fi utilizată în cadrul aplicațiilor care se bazează pe implementări secvențiale şi pe prelucrarea exclusivă a datelor de intrare (aplicații financiare, bancare şi de asigurări).
- Metodologia **DSSD** (Data Structured System Development), a fost introdusă de Warnier în 1976. Ideea de bază pe care este construită metodologia este aceea că identificarea corectă, precisă a structurilor de date ale problemei de rezolvat va conduce la realizarea unui program bine structurat şi deci bine proiectat. Utilizarea acestei metode este recomandată a fi utilizată pentru aplicații care au drept obiectiv principal procesarea datelor.
- Metodologia **SA/SD** (Structured Analysis/Structured Design) presupune folosirea unor multitudini de diagrame pentru a descrie logic un sistem.
- Metodologia **MERISE**. Dezvoltarea metodei MERISE se datorează inițiativei ministerului francez al industriei, care a venit cu propunerea realizării unei metodologii de analiză şi proiectare a sistemelor informatice în timp real ce implică manipularea informațiilor stocate în baze de date. Scopul principal al acestei metode este acela de concepție a unui sistem informațional organizațional şi informatizat.
- Metodologia **Yourdon/Constantine**, care mai este cunoscută şi sub denumirea de proiectare structurată, foloseşte diagrama fluxului datelor pentru analiză, urmată de descompunerea funcțională specifică urmărind obținerea unor module stabile cu interconectare minimă cu alte module. Ca faze principale enumerăm: studiul şi analiza sistemului existent, conceperea sistemului, proiectarea de detaliu, şi elaborarea programelor.
- Metodologia **Meyer**, cunoscută şi sub denumirea de proiectare compusă, are drept obiectiv minimizarea relațiilor intermediare şi maximizarea relațiilor intramodulare. Această metodă permite o structurare a sistemului/programelor în funcție de fluxul datelor, specificându-se ordinea de execuție a unei funcții de prelucrare elementare sau agregate.

Metodologia MERISE

Dezvoltarea metodei MERISE se datorează inițiativei ministerului francez al industriei, care a venit cu propunerea realizării unei metodologii de analiză şi proiectare a sistemelor informatice în timp real ce implică manipularea informațiilor stocate în baze de date.

Scopul principal al acestei metode este acela de concepție a unui sistem informațional organizațional şi informatizat.

În prezent există un produs informatic AMC*Designer, care implementează metoda MERISE prin formalizarea automată a unii sistem informatic şi elaborarea unei documentații adecvate.

În cadrul metodei MERISE se face o distincție între aspectul static şi dinamic al sistemului informatic, pe de o parte, şi *viziunile de referință* asupra acestui sistem (comunicație; date, prelucrări sau tratamente; definiții, descrieri, instrumente auxiliare). Astfel, **aspectul static** al unui sistem informatic este reprezentat de comunicații, definiții şi descrieri, deoarece, în principiu, aceste elemente sunt invariante în timp. Rezultă că aspectul static are în vedere latura semantică a unui sistem informatic. **Aspectul dinamic** este redat prin prelucrări şi date, elemente ce au un anumit grad de variabilitate în raport strict cu evoluția traiectoriei întregului sistem asociat unui organism.

Din punct de vedere al metodologiei MERISE, se disting trei cicluri care modelează şi dezvoltă noul sistem informatic:

- ciclul de viață (recursivitatea);
- ciclul de decizie (conducerea sau ciclul de validare);
- ciclul de abstractizare (raționamentul sau ciclul de modelare).

Dacă se utilizează reprezentarea tridimensională a acestor cicluri se obține următoarea abordare:

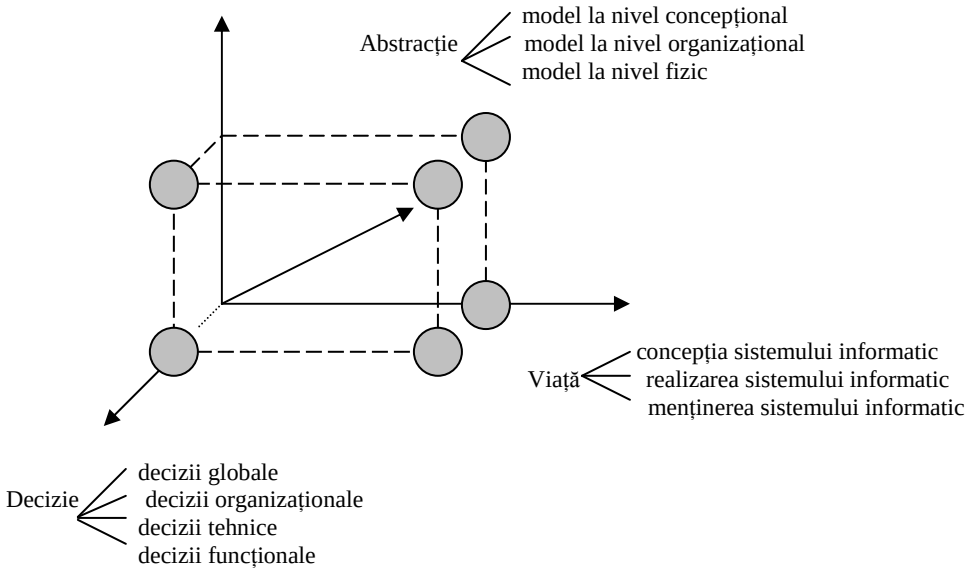


Fig. 4.1. – Definirea etapelor de realizare a unui sistem informatic prin intermediul metodei MERISE

Aceste cicluri asigură folosirea următoarelor elemente în realizarea unui sistem informatic:

- **Perioada de realizare**, care este intervalul de timp în limitele căruia sunt realizate aspecte ale unui astfel de sistem, adică sunt construite şi / sau utilizate componentele globale ale unui astfel de sistem. În realizarea efectivă a unui sistem informatic există trei perioade fundamentale:
  - Perioada de concepție asigură definirea soluțiilor generale în termeni generali, evaluarea soluțiilor de organizare şi a celor tehnice, definirea completă a viitorului sistem informațional organizațional. Perioada de concepție are sarcina de a realiza punctul de vedere al utilizatorului.
  - Perioada de realizare va asigura specificațiile tehnice complete ale viitorului sistem informațional informatizat, elaborarea procedurilor automate necesare, inclusiv punerea în

funcțiune a noului sistem. Rezultă că această perioadă realizează punctul de vedere al realizatorului noului sistem.

- Perioada de menținere în funcțiune trebuie să asigure funcționarea efectivă, eliminarea anomaliilor apărute şi, eventual evoluția sistemului realizat.

♦ **Etapele** (subetapele) sunt părți componente ale perioadei de concepție, realizarea şi menținerea în funcțiune, prin intermediul cărora se realizează anumite aspecte generale sau particulare ale proiectării noului sistem informatic. Aceste etape vor asigura:

- vederea globală asupra sistemului informațional organizațional;
- vederea externă, a utilizatorului sistemului informațional organizațional;
- vederea internă, a realizatorului sistemului informațional informatizat;
- vederea funcțională, a organismului respectiv.

Din cele prezentate rezultă că pentru fiecare perioadă de realizare există etape specifice, prin intermediul cărora se realizează vederile aferente noului sistem, astfel:

Perioada	Etapa	Vederea asigurată
De concepere	▪ schema directoare ▪ studiu de evaluare ▪ studiu de detaliu ▪ studiu organizațional	▪ vedere directoare/globală ▪ vedere externă ▪ vedere externă ▪ vedere externă
De realizare	▪ studiu tehnic ▪ proiectarea procedurilor ▪ implementarea	▪ vedere internă ▪ vedere internă ▪ vedere internă
De mențiune în funcțiune	▪ exploatarea curentă ▪ întreținerea ▪ dezvoltarea se noi versiuni	▪ vedere funcțională ▪ vedere funcțională ▪ vedere funcțională

♦ **Fazele utilizate în realizarea unui sistem informatic** sunt elementele componente ale unei etape, prin intermediul cărora sunt realizate aspecte pur particulare ale unui segment unitar din structura sistemului informatic abordat.

În mod concret, proiectantul sistemului informatic derulează aceste trei stadii pe tot parcursul existenței noului sistem. Aceste cicluri se desfăşoară simultan prin intermediul etapelor de proiectare.

De asemenea trebuie amintit că această metodă utilizează şase nivele de abstractizare:

♦ Nivelul global (NG) – are rolul de a coordona toate celelalte nivele: conceptual, organizațional, logic, fizic şi exploatare.

♦ Nivelul conceptual (NC) – acest nivel cuprinde componentele fundamentale pe care se va baza noul sistem informatic, independent de restricțiile de organizare a organismului supus analizei sau de dotarea prezentă sau viitoare cu tehnică de calcul şi de comunicație.

♦ Nivelul organizațional (NO) – rezultă din conversia nivelului conceptual, asigurând organizarea comunicațiilor, datelor şi prelucrărilor în raport cu cerințele compartimentale şi umane.

♦ Nivelul logic (NL) – rezultă din conversia nivelului organizațional, realizând detalierea acestuia prin partajarea activităților între utilizatorii din compartimentele organismului şi resursele practice de tehnică de calcul utilizate concret. Practic acest nivel poate conduce la mai multe variante de organizare a noului sistem informatic.

♦ Nivelul fizic sau operațional (NF) – cuprinde stabilirea soluțiilor tehnice pentru asigurarea comunicației, a datelor şi a prelucrărilor pentru noul sistem informatic proiectat.

♦ Nivelul de utilizare sau exploatare (NE) – acest nivel are ca sursă nivelul fizic şi se concretizează practic prin modelul de funcționare efectivă a noului sistem informatic proiectat.

Nivelele conceptual şi organizațional sunt adaptate concepției sistemului informațional, în timp ce următoarele trei nivele sunt asociate proiectării sistemului informațional informatizat.

Metoda MERISE foloseşte *concepte specifice* pentru fiecare nivel şi proces de modelare deoarece aceasta trebuie să asigure următoarele deziderate fundamentale:

1. *În plan vertical*, metoda MERISE permite următoarele operaţii: abstractizarea, asigurarea soluţiei de organizare, alegerea soluţiei tehnice, realizarea stării de exploatare. Aceste operaţii sunt realizate succesiv, pornind de la nivelul conceptual, continuând cu cel organizaţional şi terminând cu cel de exploatare.
2. *În plan orizontal*, MERISE asigură trei nivele: nivelul de transmitere asigură trecerea de la prelucrările locale către cele distribuite, nivelul de detaliu cuprinde transformarea de la general la particular, nivelul de descriere asigură conversia între nivelele conceptual , organizaţional, logic, fizic şi exploatare

Conversia dintre planurile vertical şi orizontal se asigură prin *validări* şi *interacţiuni* între diferite metode, aflate însă la acelaşi nivel de abordare.

Concluzionând se poate spune că proiectarea sistemelor informatice conform metodei MERISE trebuie să se bazeze pe cele trei cicluri. Ea presupune parcurgerea unor etape: elaborarea schemei directoare; studiul prealabil; studiul detaliat; studiul tehnic; elaborarea programelor; implementarea; menţinerea în funcţiune a sistemului informatic.

Etapele de elaborare a schemei directoare, studiul prealabil şi studiul detaliat reprezintă de fapt conceperea sistemului informatic. Studiul tehnic poate să facă parte din conceperea sistemului informatic sau din realizarea şi lansarea lui în execuţie. Realizarea şi lansarea în execuţie a sistemului informatic este alcătuită din etapele: elaborarea programelor şi implementarea sistemului.

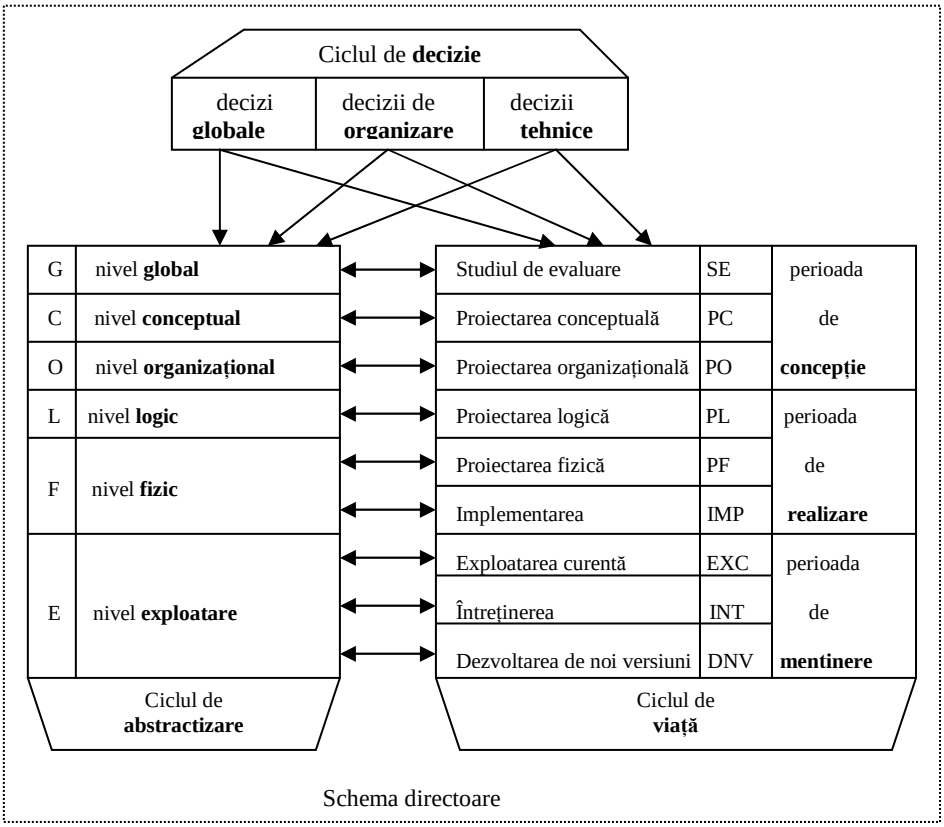


Fig. 4.2. – Corelaţia ciclurilor metodei MERISE

Metodologia MERISE utilizează modelul entitate – relaţie pentru analiza şi proiectarea modelelor statice, şi reţele Petri pentru modelele dinamice.

MERISE nu utilizează diagramele de flux de date care fac legătura între modelul static şi cel dinamic. De aceea, pentru a analiză şi o proiectare completă şi eficientă a sistemelor informatice, se recomandă utilizarea acestei metodologii în conjuncție cu alte tehnologii. Elementele care stau la baza operației de descriere a unui proces dinamic sunt următoarele elemente: evenimente, operații şi sincronizări. O schemă generală de construire a unui model MERISE cuprinde următorii paşi: construirea diagramei de flux de date utilizând o altă tehnologie; ordonarea informațiilor identificate şi selectate; construirea unui model liniar MERISE; verificarea sincronizării diferitelor operații.

Rezultatele utilizării MERISE constau în crearea unei vederi de ansamblu asupra fluxului informațiilor, a ordinii temporale şi a interdependențelor dintre acestea. Aceste informații vor fi utile atât la analiza, cât şi la proiectarea sistemului.

Capitolul 5 – Analiza şi proiectarea orientată obiecte

5.1. Concepte utilizate în tehnologia orientată obiect

Proiectarea şi analiza orientate obiect se bazează pe aşa numitul **model obiectual**. Acesta a introdus principii precum abstractizarea, încapsularea datelor, modularizarea, ierarhia şi persistența. Nici unul din aceste principii nu este nou, dar ceea ce este important este faptul că toate aceste concepte sunt tratate împreună, încercând armonizarea lor prin diferite tehnici.

Tehnologia orientată obiect reprezintă esența combinației a patru idei: [Meyer, 1997]

- *O metodă structurală* care se aplică pentru descompunerea şi reutilizarea software–ului. Sistemele software permit anumite acțiuni pe anumite tipuri ale obiectelor; pentru a obține un sistem flexibil şi reutilizabil, este indicat ca structura acestuia să se bazeze pe tipuri de obiecte şi nu pe acțiuni. Conceptul care rezultă este un mecanism mult mai puternic şi multilateral numit *clasă*, care în construirea software–ului orientat obiect serveşte ca bază atât pentru structura modulară cât şi pentru tipul sistemului.
- *O disciplină riguroasă* este o abordare corectă a problemei construirii software–ului pe care aceasta trebuie să o suporte. Ideea este că orice sistem trebuie să fie tratat ca o colecție de componente care colaborează la îndeplinirea cu succes a sarcinilor.
- *Principiul epistemologic* se referă la întrebarea cum pot fi descrise clasele. În tehnologia orientată obiect, obiectele descrise printr-o clasă sunt definite numai de modul în care ele pot fi utilizate: operațiile (cunoscute şi sub denumirea de *caracteristici*) şi proprietățile formale ale acestor operații. Această idee este exprimată formal de teoria **tipurilor de date abstracte**.
- *Tehnica de clasificare* se ocupă de observațiile în urma cărora munca este împărțită pe domenii. Software–ul nu face excepție, şi metoda orientată obiect se bazează pe o clasificare cunoscută sub numele de **moştenire**.

Nu există îndoieli asupra faptului că proiectarea orientată obiect este fundamental diferită față de metodele apărute anterior. Ea necesită un mod diferit de percepție a problemelor, de înțelegere a modului cum urmează să fie analizate, proiectate şi implementate acestea. În centrul atenției se află noțiunea de **obiect** (respectiv de clasă), aceasta punându-şi amprenta asupra întregului proces de proiectare şi implementare.

O **abordare orientată obiect** se axează mai întâi pe identificarea obiectelor din domeniul supus analizei, şi apoi se alege procedura potrivită. Putem spune, la o primă vedere, că termenul de orientat obiect înseamnă organizarea software-ului ca o colecție de obiecte discrete, fiecare obiect încorporând atât structuri de date, cât şi comportament.

Construcția fundamentală în abordarea orientată obiect o constituie **obiectul** care combină structura datelor şi comportamentul într-o singură entitate. Deci, un model orientat obiect va fi format dintr-un număr de obiecte care constituie părți bine delimitate ale sistemului modelat. [Vătui, 2000]

Obiectul este o entitate care se poate distinge dintre alte entități şi care are o semnificație în contextul aplicației modelate. Tot prin obiect se înțelege ceva asupra căruia se poate întreprinde o acțiune sau care poate întreprinde o acțiune asupra altui obiect.

Un **obiect** poate fi considerat o entitate care încorporează atât structuri de date (denumite atribute), cât şi comportament (numite operaţii).

Pentru a fi un obiect, entitatea respectivă trebuie să aibă următoarele caracteristici:

- **Identitatea**: obiectul este o entitate discretă, care se distinge dintre alte entităţi.
- **Clasificarea**: obiectele cu aceleaşi atribute şi operaţii se grupează în **clase**. Fiecare obiect poate fi considerat drept o instanţă a unei clase.
- **Polimorfismul**: aceeaşi operaţie (cu acelaşi nume) poate să aibă comportament diferit în clase diferite. Implementarea concretă a unei operaţii într-o anumită clasă se numeşte **metodă**.
- **Moştenirea**: atributele şi operaţiile se transmit de-a lungul claselor bazate pe o relaţie ierarhică.

Fiecare obiect este unic şi se caracterizează prin:

- Identitate: înseamnă că obiectele se disting între ele prin existenţă şi nu prin atributele lor. G. Booch defineşte identitatea ca fiind "acea proprietate a unui obiect care-l distinge de alte obiecte ". Jim Rumbaugh defineşte identitatea astfel:"o caracteristică distinctivă a unui obiect care indică existenţa separată a obiectului chiar dacă obiectul respectiv are aceeaşi valoare ca un alt obiect".
- Stare: cuprinde toate atributele (de regulă, statice) ale unui obiect împreună cu valorile curente (de regulă, dinamice) ale acestor atribute. Atributul poate fi o valoare simplă sau un alt obiect, iar starea unui obiect influenţează comportamentul său la primirea unui nou mesaj. G. Booch defineşte starea unui obiect "… un set de valori păstrate în mod curent de atributele sale„.
- Comportament: reprezintă activitatea să testabilă şi vizibilă din exterior, el arată modul cum "un obiect acţionează şi reacţionează, în termenii schimbării stării şi comunicării prin mesaje„ – G. Booch.

Construirea modelului obiectelor este etapa cea mai importantă a metodologiei utilizate şi reprezintă punerea în evidenţă atât a obiectelor, cu atributele şi operaţiile proprii, cât şi a relaţiilor dintre ele. Modelul obiectelor reprezintă modelul static al unei aplicaţii.

Un **model** este o noţiune abstractă şi este construit pentru a înţelege problema înainte de a implementa soluţia. Cadrul conceptual, pentru abordarea orientată obiect, îl constituie modelul obiectului care cuprinde următoarele principii:

- **Abstractizarea** constă în surprinderea aspectelor esenţiale ale unei entităţi, între anumite entităţi, situaţii sau procese din lumea reală şi concentrarea numai asupra acestor similarităţi ignorând diferenţele dintre ele. De asemenea, abstractizarea presupune omiterea părţilor sistemului care nu sunt necesare pentru înţelegerea sistemului la un anumit nivel de detaliu. Accentul, pentru un obiect, se pune pe ce este acesta şi pe ce trebuie să facă, înainte de a stabili concret detaliile de implementare. Acesta este motivul pentru care, în crearea unei aplicaţii orientate obiect, etapa esenţială este cea de analiză şi nu de implementare.
- **Încapsularea** (ascunderea informaţiilor) este un principiu al abordării orientate obiect care prevede ca obiectele să constituie entităţi de sine stătătoare; atributele unui obiect nu sunt accesibile utilizatorului în mod direct, ci doar prin intermediul metodelor definite, care astfel trebuie să se constituie într-un set cât mai complet de operaţii relative la obiect. Încapsularea este un principiu derivat al abstractizării şi constă în separarea aspectelor externe ale unui obiect, care sunt accesibile altor obiecte, de aspectele implementării interne ale obiectului, care sunt ascunse celorlalte obiecte, ceea ce face posibilă modificarea implementării obiectului fără a afecta aplicaţiile care-l utilizează. Ea este legată de securitatea programării, furnizând un mecanism care asigură accesul controlat la starea şi funcţionalitatea obiectelor. Potrivit acestui mecanism, o clasă trebuie să aibă membrii împărţiţi în două secţiuni: partea publică, care este constituită din atributele şi metodele pe care obiectele le oferă spre utilizare altor obiecte (reprezintă partea de interfaţă a obiectului respectiv), şi partea privată, care cuprinde atributele şi/sau metodele care servesc exclusiv obiectelor clasei respective (rămân inaccesibile utilizatorului).

- **Modularizarea** reprezintă împărțirea sistemului în obiecte individuale cu legături minime între ele. Acest principiu ajută în munca de programare, deoarece duce la reducerea complexității unui program, prin împărțirea acestuia în module mai mici, uşor de modificat şi de implementat.
- **Ierarhizarea** reprezintă ordonarea pe diferite niveluri a abstracțiilor. După Grady Booch, cele mai importante tipuri de ierarhii ale unui sistem complex sunt:
  - ierarhia **is-a**, adică apartenența unei clase la o familie de clase de nivel mai înalt, ceea ce se traduce prin conceptul de moştenire;
  - ierarhia **is-part-of**, adică apartenența unei clase la o altă clasă, în calitate de componentă, ceea ce duce la apariția relației de tip parte-întreg (whole-part).

Împachetarea datelor şi a comportamentului în acelaşi obiect, se referă la faptul că un obiect conține atât structuri de date, cât şi de operații, şi este utilă în cazul în care există mai multe operații cu acelaşi nume în clase diferite (în acest fel se va şti întotdeauna cărei clase îi aparține o anumită metodă). Existența unor operații (metode) diferite cu acelaşi nume, dar cu comportament diferit, se numeşte **polimorfism**.

Partajarea. Acest concept este utilizat la diverse nivele. Astfel ea poate însemna transmiterea aceloraşi structuri de date şi operații de-a lungul unor clase dintr-o ierarhie de clase. Dacă o clasă moştenește o altă clasă, atunci ea moştenește toate atributele şi operațiile clasei de bază, iar din punct de vedere al codului, acestea sunt identice cu cele ale clasei de bază, şi nu copii ale lor.

O **clasă de obiecte** descrie o mulțime de obiecte cu atribute similare, comportament similar (operații) şi relații similare cu alte obiecte din sistem. Denumirea de clasă de obiecte se poate utiliza similar cu denumirea de clasă, iar în loc de obiect care aparține unei clase se poate utiliza noțiunea de instanță a unei clase, fiecare obiect fiind o instanță a unei clase. Obiectele din aceeaşi clasă au o serie de atribute comune şi un comportament comun.

După cum se observă din figura 5.1., o clasă poate fi:

- **Clasă abstractă**, este acea clasă care nu are instanțe directe, dar ai cărei descendenți (subclase) au instanțe directe.
- **Clasă concretă** este o clasă cu instanțe directe. O clasă concretă poate avea subclase (descendenți) abstracte, dar care, la rândul lor, trebuie să aibă descendenți concreți.

Pentru uşurarea descrierii se utilizează diferite diagrame de reprezentare a claselor şi obiectelor. Fiecare autor al unei metode de analiză şi proiectare îşi creează propriile notații pentru diagramele folosite. Deosebirile între diferite abordări se referă mai mult la notații decât la noțiunile utilizate, conceptele de bază fiind în mare parte aceleaşi.

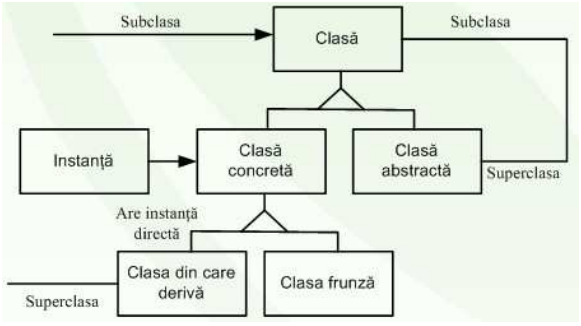


Fig. 5.1. – Model de definire a claselor într-o metodă orientată obiect

La fel ca şi obiectele, clasele au o viziune internă şi una externă. Viziunea externă a unei clase este dată de **interfața** sa, care este formată din declarații privind toate operațiile aplicabile clasei, dar poate include şi declarații ale altor clase constante etc. Interfața poate fi:

- Publică, ceea ce înseamnă că ea este accesibilă tuturor clienților săi.
- Protejată, în acest caz clasa este accesibilă numai clasei însăşi, subclaselor sale şi claselor prietene.
- Privată, clasa este accesibilă numai clasei însăşi şi claselor prietene.

Starea unui obiect trebuie să aibă o anumită reprezentare în clasa din care face parte. Acest lucru se realizează sub formă de declarații de constante şi variabile plasate în zona protejată sau privată a interfeței clasei. În acest mod, reprezentarea comună a tuturor instanțelor unei clase este încapsulată şi modificarea reprezentării nu va afecta funcționalitatea nici unui obiect.

Implementarea clasei reprezintă viziunea să internă care este formată din toate operațiile definite în interfața clasei.

Un **atribut** reprezintă o caracteristică a unei clase care are asociat un identificator şi un tip. Valoarea pe care o poate lua un atribut trebuie să fie compatibilă cu tipul său. Un atribut care poate fi calculat din alte atribute se numeşte atribut derivat.

Constrângerile sunt legături funcționale între entitățile (obiect, clasă, atribut, legătură sau asociere) din cadrul unui model al obiectelor. O constrângere restricționează valorile pe care le poate lua entitatea.

O constrângere asupra unei legături poate fi ilustrată prin exemplu din figura 5.2.:

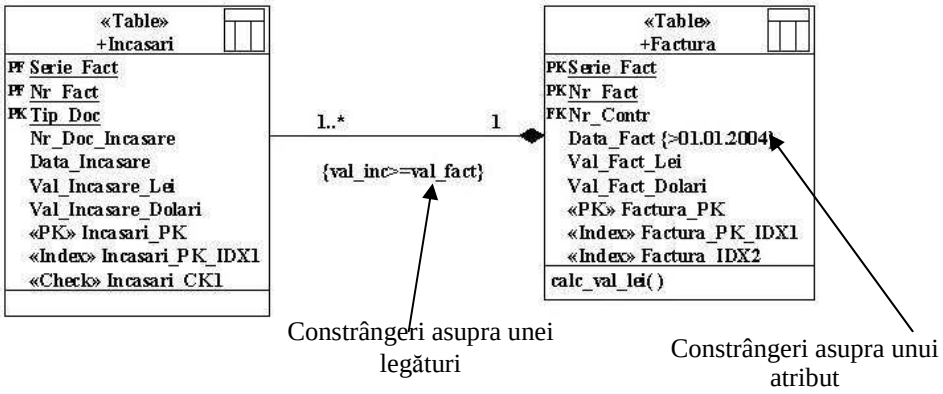


Fig. 5.2. – Constrângeri asupra unei legături şi a unui atribut

O clasă poate să conțină şi constrângeri asupra atributelor (figura 5.2., data_fact>01.01.2004). Acestea se numesc restricții. Ele se pot aplica şi asupra unei subclase. De exemplu, un pătrat este un paralelipiped căruia i se pune condiția ca toate laturile să fie egale; un cerc este o elipsă cu condiția ca ambele semiaxe să fie egale.

O **operație** care face parte dintr-o clasă este o funcție care se aplică obiectelor aparținând clasei respective. Fiecare operație are ca argument implicit clasa din care face parte. Operația poate fi polimorfică, adică aceeaşi operație poate apare în mai multe clase diferite. O clasă abstractă poate să-şi definească nişte operații, dar fără să le implementeze. Aceste operații se vor numi **operații abstracte**. O subclasă poate să redefinescă şi o operație a unei superclase, aceasta numindu-se redefinire (overriding).

Operația unui obiect este o acțiune pe care un obiect o execută asupra altui obiect pentru a primi o reacție din partea acestuia. Un mesaj este o operație pe care un obiect o execută asupra altui obiect. O operație reprezintă un serviciu pe care o clasă îl oferă clienților săi. Cele cinci tipuri de operații pe care un client le poate executa asupra unui obiect sunt:

- Modificare – în urma operației starea unui obiect se modifică.
- Selectare – operația accesează starea unui obiect fără a o modifica.
- Iterare – operația accesează toate părțile unui obiect într-o ordine bine definită.
- Constructor – operația prin care se creează un obiect şi/sau se inițializează starea sa.
- Destructor – operația prin care se distruge obiectul şi/sau se eliberează starea sa.

Metoda reprezintă implementarea concretă a unei operații pentru o anumită clasă. Numărul şi tipul argumentelor, împreună cu tipul valorii returnate, se numeşte semnătura unei operații.

Obiectele contribuie la comportamentul sistemului colaborând cu alte obiecte. Pentru a stabili relațiile şi asocierile între obiecte se utilizează legături şi asocieri. O **legătură** este o conexiune fizică sau conceptuală între obiecte. Din punct de vedere matematic, o legătură este un tuplu, adică o listă ordonată de instanțe de obiecte. Ca participant la o legătură, un obiect poate juca unul din următoarele roluri:

- **Actor** este un obiect care acționează asupra altor obiecte, dar asupra căruia nu poate acționa alt obiect.
- **Server** este un obiect pe care pot opera alte obiecte, dar el nu poate opera niciodată asupra altor obiecte.
- **Agent** este un obiect care operează asupra altor obiecte şi asupra căruia pot opera alte obiecte. Un agent lucrează în numele unui actor sau unui alt agent.

Pe de altă parte, o legătură este o instanță a unei asocieri. O **asociere** descrie un grup de legături cu aceeaşi structură şi aceeaşi semantică şi sunt cele mai frecvente relații existente între clase.

Legăturile sunt de mai multe tipuri:

- **Unu la unu** (one-to-one), caz în care unei clase îi corespunde numai o singură clasă.

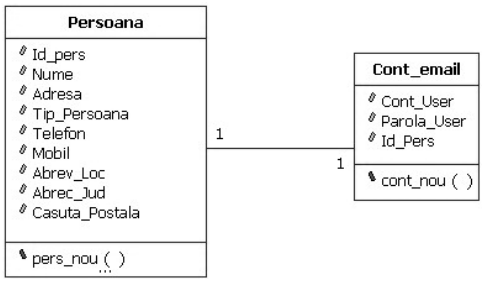


Fig. 5.3. – Diagrama claselor (legătură unu la unu)

- **Unu la mulți** (one-to-many), în această situație unei clase îi pot corespunde mai multe clase.

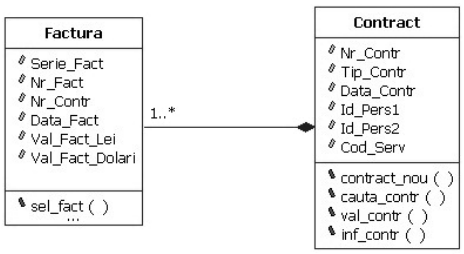


Fig. 5.4. – Diagrama claselor într-o legătură unu la mulți

- **Mulți la mulți** (many-to-many), este cazul în care fiecărei clase care participă într-o astfel de legătură îi vor corespunde mai multe clase.

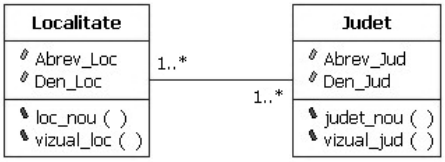


Fig. 5.5. – Diagrama claselor într-o legătură mulți la mulți

Din punct de vedere al numărului de clase care participă la o asociere aceasta poate fi: **binară**, **ternară** şi **n-ară**.

Asocierile sunt caracterizate prin **multiplicitate** şi prin **ordonarea** obiectelor (opțional) unei clase care participă la asociere. Termenul de **multiplicitate** arată câte instanțe ale unei clase pot fi legate de o singură instanță a unei alte clase. Multiplicitatea este descrisă prin numărul de instanțe care intervin în legătură, trecută în dreptul liniei la unul sau la ambele capete ale ei (0, 1, 0..*, 1..*, 0..1, *).

De multe ori este necesar, ca în cadrul unei asocieri *mulți* să se specifice dacă **ordonarea** este importantă. În acest caz se va utiliza simbolul *{ordered}* în dreptul asocierii. Ordonarea este un caz particular de constrângere asupra asocierii (capitolele unei cărți sunt ordonate în cuprinsul acesteia).

Asocierile, la fel ca obiectele, pot avea un nume şi atribute.

Uneori este necesar ca relația de *asociere să fie modelată ca o clasă*, fiecărei legături corespunzându-i o instanță a clasei respective. Ceea ce este important de reținut este faptul că această nouă clasă nu poate să existe în absența claselor de care este dependentă (figura 5.6.).

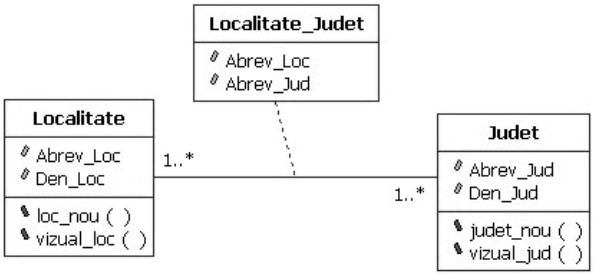


Fig. 5.6. – Asociere clasificabilă

Un alt termen utilizat în conjunctură cu cel de asociere este cel de **rol**, care reprezintă capătul unei asocieri. Astfel, într-o asociere binară fiecărui capăt i se poate atribui un rol (nume), care are un înțeles în contextul aplicației. Numele unui rol identifică în mod unic capătul respectiv al asocierii. Se recomandă utilizarea numelor de roluri în cazul în care asocierile sunt între obiectele aceleiaşi clase.

O asociere poate fi **calificată**. O asociere calificată leagă două clase de obiecte şi un **calificator**, reprezentând un atribut special care reduce efectul multiplicității unei asocieri. Dacă există o asociere de tip *unu la mulți* sau *mulți la mulți*, atunci folosirea unui calificator în partea *mulți* va identifica în mod unic obiectul respectiv. De exemplu: între clasa *Departament* şi *Angajat* există o legătură de unul la mai mulți. Deci un departament poate avea unul sau mai mulți angajați, dar un angajat lucrează numai într-un singur departament; utilizarea calificatorului *nume_angajat* reduce asocierea unu la mai mulți la o asociere unul la unul, întrucât un departament şi un nume de angajat identifică unic un angajat.

Agregarea este o relație specială, de tip **parte-întreg** (part–whole) sau **parte-din** (a-part-of), în care anumite obiecte reprezentând componente ale unui întreg sunt asociate cu acel obiect, care reprezintă întregul, sau altfel spus agregarea este o relație care leagă o clasă ansamblu cu o clasă component,

fiind o formă specială de asociere, şi nu un concept în sine, cu următoarele proprietăți: este tranzitivă şi asimetrică. Deci o clasă agregat este definită ca o clasă a cărei structură constă din unul sau mai multe obiecte simple.

Agregările pot fi clasificate în trei categorii:

- ***Agregare fixă***: clasa agregat constă dintr-un număr fix de componente, deci agregarea are o structură fixă (numărul şi tipurile părților sunt predefinite). De exemplu un Pătrat are patru Unghiuri şi patru Laturi.
- ***Agregare variabilă***: clasa agregat constă dintr-un număr finit de elemente, dar numărul de elemente poate varia. De exemplu, un Act Adițional poate avea un număr variabil de Servicii.
- ***Agregare recursivă***: conține, direct sau indirect, o instanță a aceluiaşi tip de agregare. Numărul de nivele posibil este nelimitat. De exemplu, un *Termen* este o generalizare a clasei *Expresie*, dar clasa *Expresie* este formată din *Termeni*.

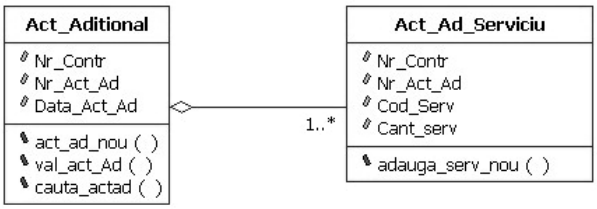


Fig. 5.7. – Relații de agregare

O posibilă rafinare a unei asocieri o reprezintă relația de utilizare care permite accesul clientului numai la interfața publică a furnizorului.

Generalizarea este relația dintre o clasă şi una sau mai multe versiuni mai rafinate ale ei. Clasa care se rafinează se numeşte clasă de bază, sau superclasă, iar fiecare versiune mai rafinată se numeşte clasă derivată, sau subclasă. Atributele şi operațiile comune unui grup de subclase sunt grupate în superclasă şi se spune că sunt moştenite de fiecare subclasă. Fiecare subclasă are şi propriile atribute şi operații. Generalizarea poate avea mai multe niveluri (maximum 2-3), iar generalizarea şi moştenirea sunt tranzitive prin aceste niveluri. Generalizarea este o construcție utilă atât pentru modelarea conceptuală cât şi pentru implementare şi facilitează modelarea structurând clasele şi evidențiind similitudinile şi diferențele între clase. Moştenirea operațiilor este utilă în implementare pentru că permite reutilizarea codului.

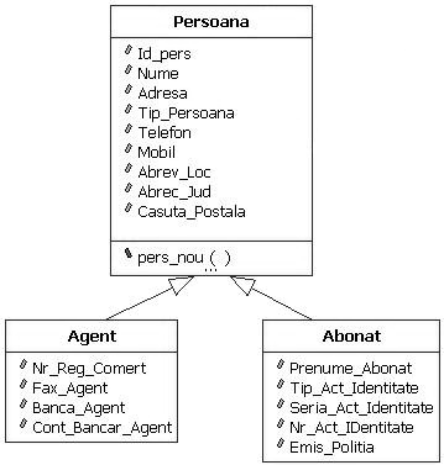


Fig. 5.8. – Generalizarea

O caracteristică importantă a unui sistem este dată de caracterul său dinamic, iar pentru descrierea acestuia se vor utiliza o serie de concepte, care la rândul lor vor fi folosite pentru realizarea **modelului dinamic al sistemului**.
La un moment dat un obiect se află într-o anumită **stare**, reprezentată prin mulțimea valorilor tuturor atributelor şi legăturilor obiectului. Ca urmare a unor stimuli externi, numiți **evenimente**, starea unui obiect poate să se schimbe în timp. Pentru fiecare obiect, această schimbare este reflectată de diagrama dinamică a obiectului respectiv. O stare este deci o abstractizare a valorilor atributelor şi legăturilor unui obiect, care specifică răspunsul unui obiect la un eveniment. Astfel, un obiect rămâne într-o stare atâta timp cât nu apar evenimente externe. Trecerea dintr-o stare în alta, cauzată de un eveniment extern, se numeşte **tranziție**.

Un **eveniment** este o modificare a contextului, care are loc la un anumit moment de timp şi nu are durată. Două evenimente pot fi legate unul de celălalt astfel: un eveniment poate să preceadă sau să succeadă logic altui eveniment (deci să fie condiționat de acesta), sau cele două evenimente să nu fie condiționate reciproc (ele nu se pot influența unul pe altul). Despre două evenimente necondiționate se spune că sunt **concurente** şi în acest caz ele pot să apară simultan. Un eveniment este o transmisie unidirecțională de informație de la un obiect la altul.

O **diagramă de stare** descrie comportamentul unei singure clase (**modelul dinamic** este o colecție de diagrame de stare, care interacționează unele cu altele prin evenimentele partajate). Astfel când un obiect recepționează un eveniment, starea să următoare depinde atât de starea curentă, cât şi de evenimentul recepționat. Diagrama de stare reflectă toate stările posibile ale unui obiect al unei clase, obiect care poate avea un ciclu de viață finit sau infinit. O diagramă de stare este un graf ale cărui noduri sunt stările şi ale cărui arce sunt tranzițiile între stări. Toate tranzițiile din aceeaşi stare trebuie să corespundă unor evenimente diferite. O astfel de diagramă este reprezentată în figura 5.9. O diagramă pentru un obiect cu viață finită are o stare de pornire (inițială) şi una de sosire (finală).

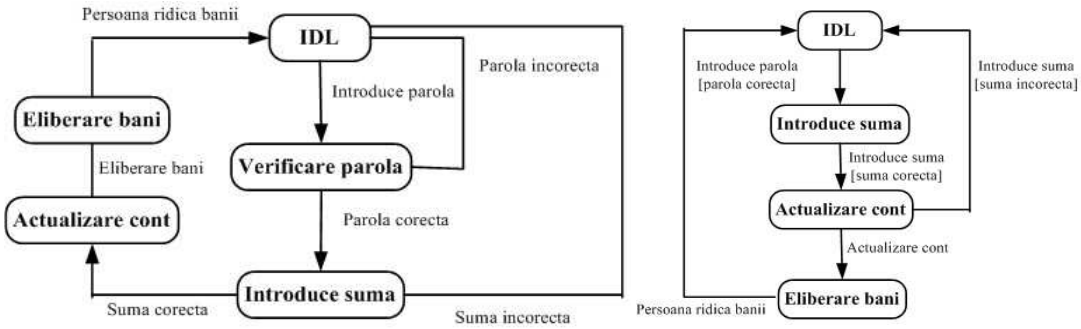


Fig. 5.9. – Diagrama de stări pentru un automat de scos bani

O **condiție** este o funcție booleană având drept parametri obiecte. Condițiile pot fi folosite pentru a stabili momentul când va avea loc o anumită tranziție. O tranziție condițională se execută numai atunci când apare evenimentul şi condiția corespunzătoare este îndeplinită.

Diagramele de stare nu specifică numai evenimentele, ci ele descriu şi ce face un obiect, ca răspuns la un eveniment. Răspunsul obiectelor este descris de operații, care pot fi de două tipuri:

- **Activitatea** este o operație care are o anumită durată (necesită un interval de timp pentru a fi executată). Ea este asociată unei **stări** şi se execută în starea respectivă. O activitate implică o operație continuă, ca de exemplu menținerea unei ferestre pe ecran. Reprezentarea unei activități se face, în diagrama de stări, prin cuvântul ***do:***, urmat de ce se realizează.
- **Acțiunea** este o operație instantanee. Ea este asociată cu un **eveniment**. Notația pentru o acțiune este slash (/) urmat de numele acțiunii care succede evenimentului ce a produs acțiunea.

Trebuie reamintit faptul că modelul dinamic specifică secvențe permise de transformări ale obiectelor din modelul obiectelor. O diagramă de stare descrie toată sau o parte din comportarea unui obiect al unei clase date. Stările sunt clase de echivalență ale valorilor atributelor şi legăturilor unui obiect. Evenimentele pot fi reprezentate ca operații în modelul obiectelor. Modelul dinamic al unei clase este moştenit de subclasele sale, acestea moştenind atât stările, cât şi tranzițiile superclasei.

5.2. Limbajul unificat de modelare - UML

UML (**U**nified **M**odeling **L**anguage) a fost conceput cu scopul de reuni elementele cele mai semnificative ale tuturor metodelor de modelare şi analiză cunoscute până în acel moment (anul 1994) şi de a furniza un mecanism eficient de dezvoltare a programelor, dar mai ales de a impune o metodă unică de analiză. Această operație de unificare prezintă anumite avantaje şi dezavantaje.[Chiorean, 1998]

Avantaje:

- eliminarea unor diferențe de concepție şi de notație;
- eliminarea conceptelor ce nu s-au dovedit viabile şi păstrarea celor care s-au dovedit utile.

Dezavantaje:

- introducerea inevitabilă a unui număr de noi concepte, sporind astfel complexitatea metodei;
- fiecare metodă prezentată până în acest moment ținea cont de anumite particularități ale problemelor, ceea ce nu mai este posibil în noul context.

UML (Unified Modeling Language) furnizează o arhitectură pentru analiza şi proiectarea obiectuală a unui sistem cu unul din limbajele corespunzătoare pentru specificarea, vizualizarea, construirea şi documentarea sistemului software executat de proiectant şi pentru modelarea afacerilor şi a altor sisteme non-software.

Această specificare reprezintă convergența celor mai bune metode practice utilizate în industria tehnologiei obiectuale. UML este succesorul, am putea spune cel mai bun pentru limbajele de modelare obiect, al celor trei metode precedente orientate obiect, şi anume, Booch, OMT şi OOSE, incluzând în plus expresivitate în manipularea problemelor de modelare pe care aceste trei modele nu le utilizează pe deplin. Unul dintre scopurile principale ale UML a fost acela de a îmbunătăți situația industriei software prin asigurarea interoperabilității instrumentelor de modelare vizuală a obiectului.

Totuşi, pentru a permite schimbul corect al modelului informațional între instrumente, este necesară o convenție în ceea ce priveşte semantica şi notația utilizată. În acest scop, UML îndeplineşte următoarele condiții:

- Definirea formală a metamodelului comun de analiză şi proiectare a obiectului (OA&D) reprezintă semantica modelului OA&D, care include modele statice, modele comportamentale, modele de utilizare şi modele structurale.
- Specificații **IDL** (**I**nterface **D**efinition **L**anguage) pentru mecanisme care să asigure interoperabilitatea modelului între instrumente OA&D. Acest document include un set de interfețe IDL care ajută la crearea unei construcții dinamice şi a unei linii transversale de intersecție a modelului utilizat.
- Notație convențională (human-readable) pentru reprezentarea modelelor OA&D. UML prezintă o grafică elegantă pentru a prezenta pe larg semantica sa bogată. Notația este o componentă esențială a modelării OA&D şi UML.

Totul a pornit de la **OMG** (**O**bject **M**anagement **G**roup) al cărui obiectiv este de a asigura dezvoltarea tehnologiei obiectuale şi influențarea ei în direcția stabilită de **OMA** (**O**bject **M**anagement **A**rchitecture), care furnizează infrastructura conceptuală pe care toate specificațiile OMG sunt

bazate. Adoptarea de către OMG a specificației UML reduce gradul de confuzie în cadrul industriei limbajelor de modelare şi permite schimbul reciproc între instrumentele de dezvoltare vizuale utilizate.

5.2.1. Evoluția limbajului UML

Precursori UML: limbajele de modelare orientate obiect au început să apară la începutul anilor 1970, şi în 1980 existau deja o varietate de metodologii experimentale cu diferențe apropiate față de analiza şi proiectarea orientată obiect. Unele dintre aceste tehnologii au influențat aceste limbaje, printre care enumerăm modelarea entitate-relație (Entity-Relationship modeling), **SDL** (Specification & Description Language) şi alte tehnologii. Numărul acestor limbaje de modelare a crescut în perioada 1989–1994 de la 10 la mai mult de 50. La mijlocul anilor 1990, au început să apară noi variante ale acestor metode, cel mai reprezentativ fiind modelul Booch'93, dezvoltarea modelului OMT, şi Fusion. Aceste metode încep să încorporeze tehnici unele de la altele, şi astfel apar metodele OOSE, OMT-2, şi Booch'93 îmbunătățită.

UML a fost dezvoltat de Rational Software şi partenerii săi. Efortul de unificare a început oficial în octombrie 1994 când Grady Booch şi Jim Rumbaugh de la Rational Software Corporation încep unificarea metodelor Booch şi OMT. Prima versiune UML, numită Unified Method, (UML 0.8) a apărut în octombrie 1995. În toamna aceluiaşi an, Ivar Jacobson, şi compania lui Objectory, se alătură companiei Rational; şi se încorporează şi metoda OOSE. Ca autori ai celor trei metode Booch'93, OMT, şi OOSE, Grady Booch, Jim Rumbaugh, şi Ivar Jacobson au motivat crearea unui limbaj unificat de modelare din trei puncte de vedere:

1. Aceste metode au evoluat cu scopul de a fi independente unele față de altele.
2. Prin unificarea semanticii şi notației, aceste metode ar putea aduce o anumită stabilitate pe piața de produse orientate-obiect.
3. Ei sperau ca această colaborare să înlăture incovenientele celor trei metode.

Efortul lor s-a concretizat în apariția documentației pentru variantele UML 0.9 şi UML 0.91, în perioada iunie–octombrie 1996. Din acest moment în dezvoltarea acestui limbaj şi-au depus eforturile şi firmele: Digital Equipment Corp., HP, I-Logix, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, Texas Instruments, Intellicorp şi Unisys. Această colaborare a condus la realizarea unui limbaj de modelare puternic, expresiv, foarte bine definit, şi cu aplicabilitate generală, numit UML.

În ianuarie 1997 IBM & ObjectTime, Platinum Technology, Ptech, Taskon & Reich Technologies, şi Softeam propun separarea **RFP** (Request For Proposal – care din 1996 a fost obiectul OMG) de OMG.

Prima versiune, care a fost publicată şi propusă aprobării OMG în ianuarie 1997, a fost **UML 1.0** care cuprindea următoarele documente principale: UML Semantics; UML Summary; UML Notation Guide; UML Process-Specific Extensions, şi drept surse secundare următoarele documente: OMA; CORBA 2.0; Object Analysis & Design RFP-1; Rational Process. OMA, CORBA 2.0 şi Object Analysis & Design RFP-1 au fost utilizate pentru a susține acordul OMG şi a furniza termenul de obiect distribuit care completează UML-ul, iar Rational Process pentru a furniza termenii de proces şi arhitectură care completează, de asemenea UML-ul.

Diagrame inițiale care au fost propuse sunt:

- Diagrama cazurilor de utilizare;
- Diagrama claselor;
- Diagrame de comportament:
  - Diagrama de stare;
  - Diagrama de activitate;
  - Diagrama secvențială;

- Diagrama de colaborare;

♦ Diagrame de implementare:
- Diagrama de componente;
- Diagrama de aplicație.

Îmbunătățirile aduse acestei prime variante au condus la apariția în iulie 1997 a versiunii **UML 1.1**, care a fost acceptată spre publicare în septembrie 1997, şi adoptată în 14 noiembrie 1997, devenind astfel standard OMG.

Drept diagrame au fost propuse cele nouă diagrame de la versiunea 1.0 cu singura deosebire că diagrama secvențială şi de colaborare au fost grupate în diagramele de interacțiuni.

Documentele necesare a fi parcurse pentru înțelegerea primei variante a limbajului, UML 1.1, sunt:

♦ UML Semantics, acest document defineşte semantica utilizată de UML (metamodelul UML).
♦ UML Notation Guide, specifică sintaxa grafică (diagramele) utilizate pentru exprimarea semanticii folosite de metamodelul UML (prezentarea metodei).
♦ OCL (Object Constraint Language) Specification, defineşte sintaxa OCL, semantica, şi gramatica utilizată (instrumentul necesar înțelegerii limbajului).

Dar pentru înțelegerea deplină mai trebuiesc parcurse şi următoarele documente:

♦ OA&D CORBAfacility Interface Definition – specifică o interfață de interoperabilitate instrument utilizând CORBA IDL. Acest document specifică interfețele pentru CORBAfacility pentru Object Analysis & Design, consecvent cu UML v.1.1. OA&D Facility reprezintă un depozit pentru modele exprimate în UML. Uşurința (îndemnarea – facility) permite crearea, memorarea, şi manipularea modelelor UML, deoarece permite o varietate largă a capacităților de dezvoltare bazate pe model, incluzând:
- Desenarea modelelor UML în alte notații şi UML;
- Aplicarea unor procese şi metode în stilul liniilor directoare;
- Matrici, interogări, şi rapoarte;
- Automatizarea activităților dezvoltării ciclului de viață.

O primă directivă a OA&D Task Force este de a realiza interoperabilitatea semantică între instrumentele OA&D. Figura următoare descrie mai multe alternative pentru schimbarea informației de model între instrumente.

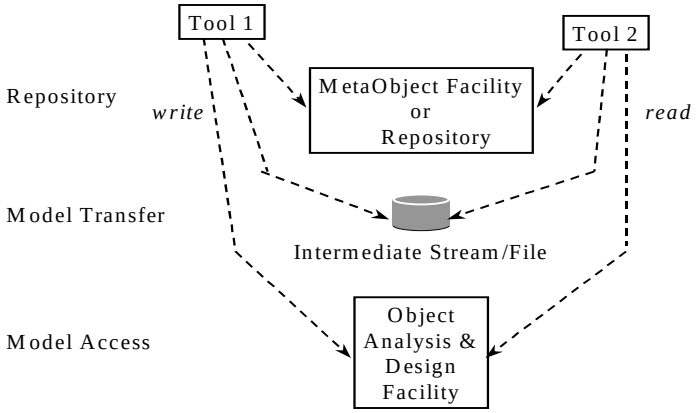


Fig. 5.10. – Alternative de model partajat între instrumente OA&D

Repository Două instrumente pot interfața către acelaşi depozit şi accesa un model de aici. MOF (MetaObject Facility) poate fi acest depozit.

Model Transfer Două instrumente pot înțelege același format de structură și schimba modele prin această structură, care poate fi un fișier. RFP face referire la acest format drept "import facility". Având această interfață va fi necesară furnizarea unei căi pentru instrumente care nu sunt implementate într-un API (CORBA sau non CORBA) sau mediu de lucru depozit.

Model Access Două instrumente pot schimba modele pe o bază detaliu-prin-detaliu. RFP-ul se referă la această metodă ca o facilitate de conectivitate "connection facility".

- **UML Proposal Summary** – face un sumar a propunerii OMG și discută relația UML-ului cu alte tehnologii, incluzând meta-metamodelul MOF.

Alte documente cuprinse în această versiune sunt:

- UML Extension for the Objectory Process for Software Engineering;
- UML Extension for Business Modeling.

Principalele deosebiri dintre versiunile UML 1.0 și UML 1.1 erau:

- Creșterea formalismului;
- Unificarea semantici interacțiuni și colaborării;
- Simplificarea modelului interfeței / tipului / clasei;
- Unificarea semanticilor relațiilor;
- Extinderea semantici managementului modelului, incluzând modele și subsisteme;
- Extinderea semanticii cazului de utilizare;
- Îmbunătățirea structurii împachetării;
- Îmbunătățirea transpunerii notației către semantică.

A urmat în 1998 UML 1.2; în 1999 UML 1.3; în mai 2001 UML 1.4; tot în 2001 UML 1.5 **RTF** (Revision Task Force); și în 2002 UML 2.0 RFP. Mai trebuie amintit că din 1994 și până la 17 ianuarie 1997 s-au înregistrat 5 propuneri distincte, dintre care cele mai importante au fost UML (Unified Modeling Language), OML (Open Modeling Language) și Object Time (IBM). În martie 2003 a fost standardizată versiunea UML 1.5, numit și Unified Modeling Language Specification.

Specificarea UML 1.3 cuprinde:

- UML Summary
- UML Semantics
- UML Notation Guide
- UML Standard Profiles:
  - Software Development Processes
  - Business Modeling
- UML CORBAfacility Interface Definition
- UML XML Metadata Interchange DTD
- Object Constraint Language

Principalele deosebiri dintre UML 1.2 (și UML 1.1) și UML 1.3 sunt:

- Modificarea relațiilor folosite între cazurile de utilizare. UML 1.1 avea două relații între cazurile de utilizare <<uses>> și <<extends>>, amândouă fiind de fapt stereotipuri ale relației de generalizare. UML 1.3 are trei relații: <<include>>, care este un stereotip al relației de dependență, și ea înlocuiește utilizarea lui <<uses>>; relația de generalizare, și <<extends>>, un stereotip al relației de dependență.
- Diagrama de activitate. Modificările se referă la utilizarea barelor de sincronizare, care sunt de bifurcație și de reuniune, la concurența dinamică etc.

Din anul 2002 versiunea **UML 2.0** este propusă pentru evaluare (proposals under evaluation). În această versiune există 13 diagrame și cuprinde 4 componente:

- UML Infrastructure
- UML Superstructure
- OCL (Object Constraint Language)

- UML Diagram Interchange

Dintre firmele care utilizează UML drept standard în procesele de dezvoltare şi producţie, care acoperă o gamă largă de discipline, precum modelarea afacerilor, managementul resurselor, analiză şi proiectare, programare, enumerăm: Data Access Corporation, Electronic Data Systems Corporation, Enea Data, Hewlett-Packard Company, IBM Corporation, I-Logix, ICON Computing, IntelliCorp and James Martin & Co (James Odell), OAO Technology Solution,ObjectTime Limited, Oracle Corporation, PLATINUM Technology Inc., Rational Software (Grady Booch, Ivar Jacobson,Jim Rumbaugh), SAP, SOFTEAM, Sterling Softwrae, Taskon, Unisys Corporation.

UML este îmbunătăţit în continuare prin intermediul unei echipe, conduse de Cris Kobryn, formate din personalităţi precum: James Odell, Dilhar DeSilva, Martin Griss, Eran Gery, Karin Palmkvist, Guus Ramackers, Bran Selic, Sridhar Iyengar, Gunnar Overgaard, şi Jos Warmer. De asemenea, Grady Booch, Ivar Jacobson, şi Jim Rumbaugh lucrează în echipă pentru îmbunătăţirea acestuia, datorită experienţei lor în acest domeniu. Există de asemenea contribuţii individuale cum ar fi: Peter Coad, Tim Harrison, Bertrand Meyer, Bill Premerlani, Ed Yourdon, Oliver Wiegert, şi mulţi alţii.

O succintă evoluţie a UML-ului este prezentată în figura 5.11.

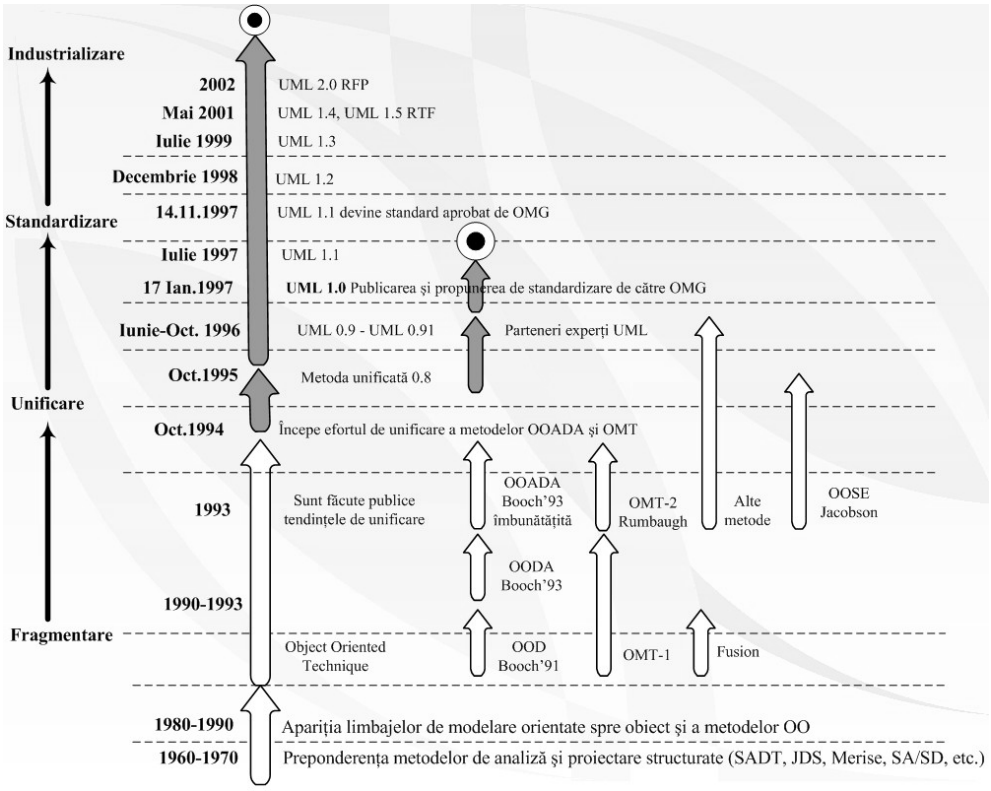


Fig. 5.11. – Evoluţia limbajului de modelare UML

UML a fost selectat drept standard al limbajelor de modelare orientate-obiect (***OOML*** – *Object Oriented Modeling Language*) de către OMG în noiembrie 1997, şi este utilizat de dezvoltatorii de produse software.

Această specificare reprezintă convergenţa celor mai bune metode practice utilizate în industria tehnologiei obiectuale şi care au avut succes în modelarea sistemelor mari şi complexe.

UML este un limbaj pentru specificarea, construirea, vizualizarea şi documentarea unui sistem software complex.

Un programator anonim a spus "1 bitmap = 1 megaword". Din această afirmație putem să tragem concluzia cât de semnificativă este utilizarea unor metode, limbaje pentru care au fost create diverse instrumente de lucru, care redau descrierea sistemului prin intermediul diferitelor notații grafice, mult mai sugestive, şi care uşurează enorm munca în echipă, reducând în acelaşi timp perioada pentru analiza şi proiectarea sistemului. [Kobryn, 2001]

Sfera de întindere a limbajului unificat de modelare este:

- În primul rând şi cel mai important, UML este un limbaj care reuneşte conceptele utilizate de Booch, OMT şi OOSE. Rezultatul este un limbaj de modelare unic, comun şi în mare măsură uşor de folosit pentru utilizatorii acestor metode, precum şi altele.
- În al doilea rând, UML extinde sfera de cuprindere a ceea ce se doreşte a fi făcut cu metodele existente.
- În al treilea rând, UML se bazează pe limbaje standard de modelare şi nu pe un proces standard. Aşa cum sugerează chiar numele limbajului, sprijinul oferit de limbaj vizează în primul rând construirea modelelor.

Acestea ar fi noțiunile care țin de UML, dar în afara acestora ar mai fi de adăugat următoarele: UML este un *limbaj vizual de modelare*, dar care nu are pretenția de a fi un limbaj de programare vizual, el este un limbaj universal pe care dezvoltatorii îl pot utiliza pentru descrierea sistemelor lor. De ce este un limbaj ? Deoarece UML utilizează o sintaxă (reprezintă elementele limbajului care sunt asamblate în expresii) şi o semantică (înțelesurile expresiilor sintactice) pentru a descrie sistemul. Pentru descrierea sistemului dezvoltatorii folosesc anumite notații grafice standard. Astfel, UML *surprinde şi descrie în limbaj natural sistemul*. Ceea ce nu rezolvă UML este problema schimbului de date, care poate fi rezolvat cu Microsoft Repository. Pentru programarea codului se poate utiliza unul din limbajele orientate obiect cele mai cunoscute.

UML dispune la ora actuală de mai multe instrumente prin intermediul cărora toată munca este executată prin intermediul calculatorului. Instrumentele şi interoperabilitatea dintre ele este dependentă de definiția semanticii şi a notației: UML defineşte un metamodel semantic, şi nu un instrument de interfață, memorare, sau execuție a modelului.

5.2.2. Noțiuni generale despre UML

UML reprezintă o arhitectură bazată pe 4 niveluri de abstractizare (figura 5.12.). Cele patru niveluri de abstractizare sunt: **meta-metamodelul** (reprezintă infrastructura pentru o arhitectură de metamodele), **metamodelul** (reprezintă o instanță a unui meta-metamodel şi defineşte semantica necesară pentru reprezentarea modelelor aplicației cu ajutorul UML), **modelul** (reprezintă o instanță a metamodelului) şi **obiecte utilizator** (reprezintă o instanță a unui model, utilizată pentru descrierea unui domeniu de informație specific).

Metamodelul UML este un model logic şi nu unul fizic (de implementare) şi are în componența sa trei pachete logice: Elemente de comportament (Behavioral Elements), Elemente de bază (Foundation) şi Mecanisme generale (Model Management), după cum se poate observa din figura 5.13.

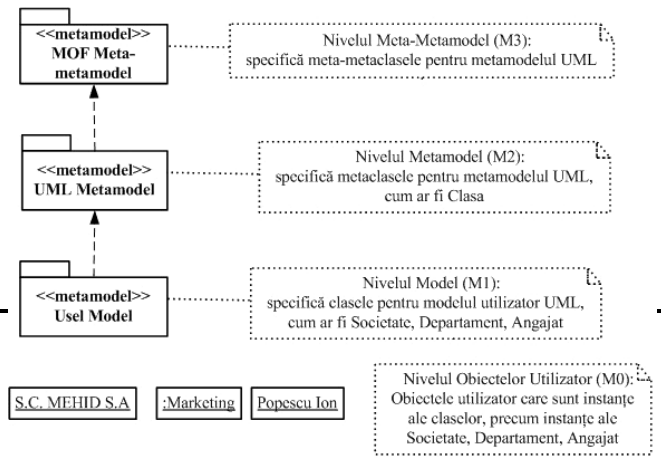


Fig. 5.12. – Arhitectura metamodelului UML

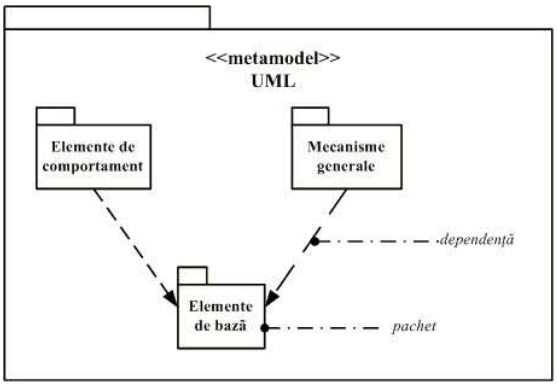


Fig. 5.13. – Structura pachetului UML

În cadrul fiecărui pachet, elementele unui model sunt definite în următorii termeni:

- **Sintaxa abstractă**: este prezentă în diagrama claselor UML şi prezintă metamodelul UML, conceptele sale (metaclase), relaţiile şi constrângerile. Diagrama prezintă reguli foarte bine formulate, în special cerinţele legate de multiplicitatea legăturilor, şi când, sau nu, instanţa unui subconstructor particular trebuie să fie ordonată. Pentru fiecare metaclasă, atributele sale sunt enumerate împreună cu o scurtă explicaţie. În plus, numele rolurilor asocierilor conectate la metaclasă sunt scrise la capetele fiecărei asocieri.
- **Reguli foarte bine formate** (reguli de corectitudine - well-formedness rules): sunt definite regulile şi constrângerile impuse modelelor corecte. Semantica statică a metaclaselor UML, exceptând multiplicitatea şi constrângerile, sunt definite ca un set de invariante ale unei instanţe a metaclasei.
- **Semantica**: specifică modelele utilizate pentru structura şi comportamentul obiectului, prezentat în mod normal în limba engleză.

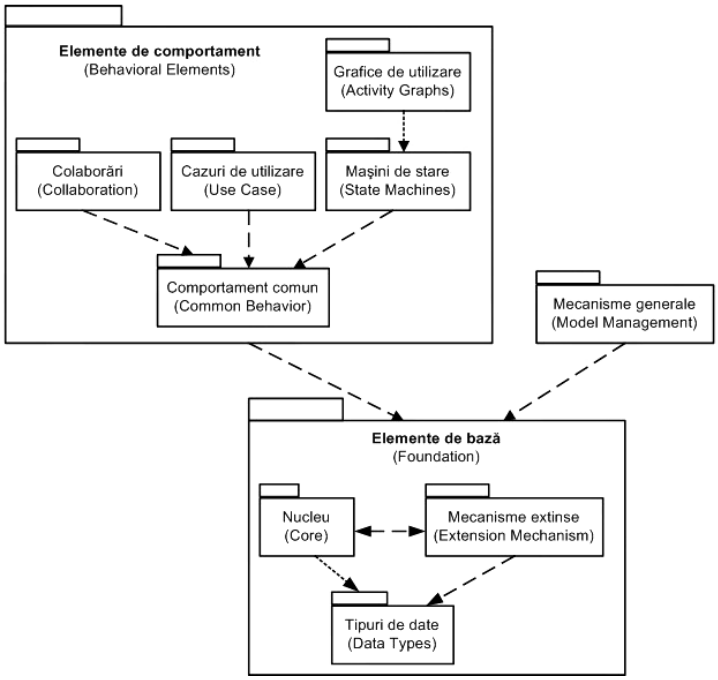


Fig. 5.14. – Diagrama claselor UML,
structurarea în pachete şi subpachete

Figura 5.14. arată dependențele între elementele modelului UML din diferite pachete. **Modelele structurale** (modele statice) accentuează structura obiectelor într-un sistem, incluzând clasele lor, interfețele, atributele şi relațiile. **Modelele de comportament** (modele dinamice) accentuează comportamentul obiectelor într-un sistem, incluzând metodele, interacțiunile, colaborările şi starea lor istorică. Semantica este definită într-o modalitate independentă de limbajele de implementare. O implementare a semanticii, fără o interfață consistentă şi alegerile implementării, nu garantează interoperabilitatea instrumentului.

Alegerea modelelor şi diagramelor are o profundă influență asupra problemei supuse analizei şi modului în care soluția corespunde problemei. Din acest motiv:

- Fiecare sistem complex va trebui supus unei analize din mai multe puncte de vedere.
- Fiecare model poate fi exprimat la diferite nivele de fidelitate.
- Cel mai bun model este cel legat de realitate.

Un model reprezintă o colecție de obiecte (Classes, Packages, Actors, Use Cases, Components şi Nodes), relații (Association, Dependencies şi Generalization) şi diagrame. Din punctul de vedere al unui model, UML defineşte diferite tipuri de diagrame grafice, al căror număr a crescut de la 9 tipuri standard, varianta UML 1.1, până la 13 tipuri, varianta UML 2.0.

- Diagrama cazurilor de utilizare (use case diagram).
- Diagrama claselor (class diagram).
- Diagrama obiectelor (object diagram).
- Diagrame de comportament (behaviour diagrams):
  - diagrama de stare, numită şi diagrama graficelor de stare (statechart diagram);
  - diagrama de activitate (activity diagram);
  - diagrame de interacțiuni (interaction diagrams):
    - diagrama secvențială (succesiune) (sequence diagram);
    - diagrama de colaborare (collaboration diagram);
- Diagrame de implementare (implementation diagrams):
  - diagrama de componente (component diagram);
  - diagrama de aplicație (de dezvoltare) (deployment diagram).

Aceste diagrame furnizează perspective multiple asupra sistemului analizat, privind analiza şi/sau dezvoltarea sa. Legătura dintre modele şi diagrame este ilustrată în figura 5.15.

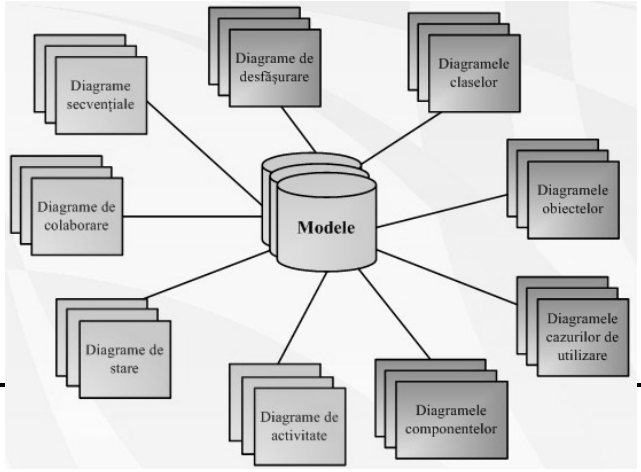


Fig. 5.15. – Modele, vederi şi diagrame

De ce este UML un limbaj de modelare şi nu o metodologie ? Modelarea este proiectarea aplicațiilor software înainte de implementarea codului în diferite limbaje de programare. O metodologie este alcătuită din două componente: un limbaj de modelare şi un proces de modelare care arată succesiunea de paşi care trebuie parcurşi pentru realizarea modelului.

Un limbaj de modelare trebuie să includă:

- Elementele modelului – conceptele de modelare fundamentale şi semantică.
- Notația – interpretarea vizuală a elementelor modelului. Notația UML este un element esențial pentru UML care permite comunicarea între membrii echipei de lucru. Acceptarea notației este opțională, dar semantica nu va fi foarte semnificativă dacă nu este bine exprimată. Acceptarea notației este definită în cadrul fiecărui tip de diagramă UML: cazuri de utilizare, clase, grafice de stare, de activitate, secvențială, de componente şi de dezvoltare.
- Linii directoare (guidelines) – utilizarea sintagmelor (expresiilor) în interiorul produsului.

Scopurile principale ale UML sunt următoarele:

- Să furnizeze utilizatorilor un limbaj de modelare vizual expresiv pentru dezvoltarea şi specificarea semnificației modelului.
- Mecanisme extensibile şi specializate pentru a extinde conceptele de bază.
- Suport pentru specificații care sunt independente de un limbaj de programare particular şi dezvoltarea proceselor.
- Furnizează o bază formală pentru înțelegerea limbajului de modelare.
- Încurajează creşterea vânzărilor de instrumente obiect.
- Suportă concepte de dezvoltare de nivel superior precum componente, colaborări, frameworks şi modele.
- Integrează cele mai bune practici existente la ora actuală.

5.2.3. Concepte UML

Unul din scopurile modelării vizuale este de a înțelege arhitectura aplicației. Membrii echipei de lucru pot înțelege această arhitectură a aplicației prin intermediul diagramelor de modelare UML. Pentru înțelegerea conceptelor UML sunt utilizate trei tipuri principale de blocuri constructive:

- Elementele.
- Relațiile.
- Diagramele.

precum şi:

- Mecanisme extensibile: stereotipuri, constrângeri, şi valori etichetate (tagged value).

Conceptele cele mai utilizate sunt cele preluate de la cei trei care au contribuit la apariția limbajului de modelare UML, Booch, Rumbaugh şi Jacobson. Cu toate acestea, anumite elemente constructive sunt preluate de la alte metode aparținând altor specialişti din domeniul analizei şi proiectării sistemelor: Shlaer & Mellor, Martin & Odell, Wirfs & Brock, Fusion, Meyer, Harel, Gamma etc.

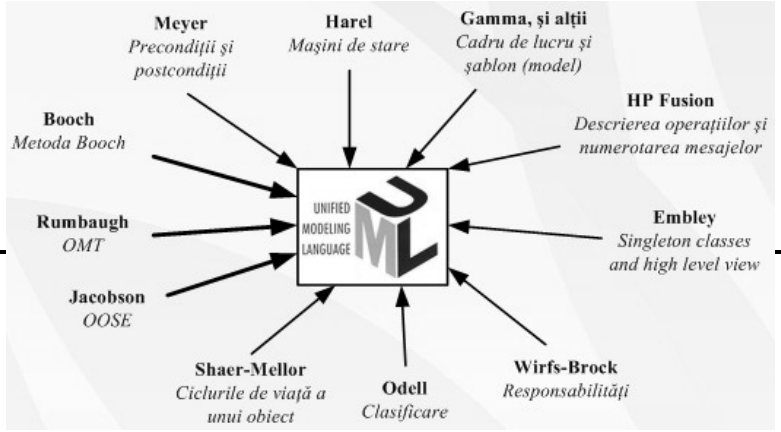


Fig. 5.16. – Contribuții la dezvoltarea UML

Elementele UML

Elementul este constituentul atomic al unui model. Într-un metamodel elementul reprezintă cea mai înaltă metaclasă în ierarhia metaclaselor, fiind o clasă abstractă. Există două metasubclase: Element de Model (ModelElement) şi Element de Prezentare (PresentationElement), care la rândul lor sunt construcții abstracte. Elementele UML pot fi grupate astfel: structurale (statice), comportamentale (dinamice), de grupare şi suplimentare. Un tip special de ModelElement este clasificatorul, care este un element ce descrie caracteristicile structurale şi de comportament, dintre care amintim: clasa, interfața, tipuri de date, nodul, componenta, semnalul, actorul, caz de utilizare şi subsistem.

Elementele structurale

Elementele structurale reprezintă părțile statice ale unui model, reprezentând elemente conceptuale sau fizice şi care sunt componente ale pachetelor Elemente de bază (clasa, interfața, nodul, componenta, tipuri de date) şi o parte din Elemente de comportament (colaborarea şi caz de utilizare).

În continuare se vor prezenta acele aspecte ale elementelor structurale care nu au fost descrise în prezentarea conceptelor de bază ale tehnologiei orientate obiect.

O **clasă** reprezintă o descriere a unui set de obiecte care au în comun aceleaşi atribute, operații, metode, relații şi semantici. Scopul unei clase este de a declara o colecție de metode, operații şi atribute care descriu complet structura şi comportamentul obiectelor. Clasele pot fi considerate drept obiecte, dar ele sunt meta-obiecte, nu obiecte ale lumii reale. Obiectele care descriu clasele au propriile lor caracteristici, deci au clase proprii numite metaclase. Fiecare obiect instanțiat dintr-o clasă conține propriul set de valori corespunzător atributelor şi suportă operațiile declarate în descrierea completă a clasei. Fiecare clasă trebuie să aibă un nume care să o distingă de celelalte clase.

O clasă activă este o clasă ale cărei obiecte au propriile fire de control (posedă unul sau mai multe procese sau cursuri ale prelucrării şi care poate iniția o activitate de control) şi al căror comportament este concurent cu alte elemente. În caz contrar, avem de-a face cu o clasă pasivă.

UML suportă următoarele tipuri de clase:

- Clasă abstractă – este acea clasă care nu poate avea instanțe directe.
- Clasă rădăcină – este acea clasă care nu are părinți (strămoşi), deci nu poate fi o subclasă a altor clase.
- Clasă frunză – este acea clasă care nu poate avea are copii (descendenți), deci nici o altă clasă nu poate fi o subclasă a clasei.

O clasă poate implementa un set de interfețe pentru a specifica setul de operații pe care le furnizează mediului său. O subclasă poate adăuga noi atribute şi/sau operații față de caracteristicile claselor ascendente, caz în care spunem că avem de-a face cu o extensie a clasei. De asemenea, o subclasă poate restrânge atributele claselor ascendente, caz în care spunem că avem de-a face cu o restricție a clase, limitându-le tipul la un singur subtip al tipului respectiv. Fiecare generalizare trebuie să se facă

după o singură proprietate. Dacă o clasă poate fi rafinată după mai multe dimensiuni distincte şi independente se realizează generalizări multiple.

O **interfață** reprezintă un set de operații, care împreună definesc un serviciu oferit de un clasificator (clasă sau componentă) şi care descrie comportamentul vizibil extern al acestuia, integral sau parțial. Un clasificator poate oferi mai multe servicii, ceea ce înseamnă că el poate realiza mai multe interfețe şi mai mulți clasificatori pot realiza aceeaşi interfață.

Un **nod** este un obiect fizic run-time (există în timpul execuției) şi care reprezintă o resursă de calcul având în general memorie şi mai multe procese capabile de prelucrare.

O **componentă** reprezintă o parte fizică a unui sistem, care implementează pachete şi permite realizarea unui set de interfețe; ea reprezintă o implementare a unui sistem, incluzând cod software (surse, binare, sau executabile) sau echivalente precum script-uri sau fişiere de comandă.

Un **tip de date** reprezintă un tip ale cărui valori nu au identitate. Tipurile de date includ tipuri primitive (precum integer sau string), tipuri enumerate etc.

O **colaborare** defineşte o interacțiune şi descrie modul în care diferiți clasificatori şi alte elemente (asocierile clasificatorilor) se utilizează la realizarea unui anumit comportament particular (un task particular); o clasă poate participa la mai multe colaborări, iar scopul unei colaborări este de a specifica modalitatea în care o operație sau un clasificator, cum ar fi un caz de utilizare, este realizată de un set de clasificatori şi asocieri.

Un **caz de utilizare** este folosit pentru a defini comportamentul unui sistem fără a dezvălui structura internă a entității. Fiecare caz de utilizare specifică descrierea unei succesiuni de acțiuni pe care sistemul le poate realiza, interacționând cu actorii sistemului, pentru a obține un rezultat observabil de un anumit actor.

În afara celor 7 elemente structurale de bază, UML mai utilizează şi alte elemente structurale, printre care pot fi menționate:

- **Actor**, este un element definit în subpachetul Use Cases al pachetul Behavioral Elements. Un actor defineşte un set coerent al rolurilor pe care utilizatorii unei entități (sistem, subsistem, clasă) îl poate executa atunci când interacționează cu entitatea respectivă. Instanța unui actor este un utilizator specific care interacționează cu entitatea, iar în cazul în care entitatea este un sistem actorii reprezintă atât utilizatori umani cât şi alte sisteme. Actorii pot face parte din interiorul sau din afara entității cu care interacționează; ei pot comunica cu un set al cazurilor de utilizare, pot avea un set de interfețe şi pot avea relații de generalizare cu alți actori. Un actor este un obiect care operează asupra altor obiecte dar asupra căruia nu poate opera alt obiect (de regulă, obiect activ).
- **Semnal**, este un element definit în subpachetul Common Behavior al pachetului Behavioral Elements. Printre caracteristicile de bază ale unui semnal se impun: un semnal este o specificare a unui stimul asincron comunicat între instanțe; un semnal poate fi ataşat unui clasificator, ceea ce înseamnă că instanța clasificatorului va fi capabilă să primească acest semnal; semnalul este un element generalizabil şi este definit independent de clasele care utilizează semnalul. Atunci când un semnal este cauzat de o caracteristică de comportament specifică apariției unei erori, se spune că s-a produs o excepție.

Elemente comportamentale reprezintă părțile dinamice ale modelelor UML şi ele sunt definite în pachetul Behavioral Elements (acest pachet specifică comportamentul dinamic al modelelor). Principalele elemente comportamentale sunt: interacțiunea şi maşina de stări.

O **interacțiune** este un comportament care conține un set de mesaje schimbat între un set de obiecte într-un anumit context, definit de ClassifierRoles, pentru a îndeplini un anumit obiectiv. Caracteristicile unei interacțiuni sunt definite în subpachetul Collaborations, iar în realizarea unei interacțiuni sunt implicate şi alte elemente:

- **Mesajul**, este definit în subpachetul Collaborations şi defineşte o comunicare particulară între instanțele specificate de o interacțiune. El specifică rolurile instanțelor expeditorului şi receptorului, în aşa fel încât rolul asocierii va specifica legătura de comunicare. Mesajul este ataşat unei acțiuni, specificând instrucțiunea care, atunci când este executată, determină comunicarea specificată.
- **Acțiunea**, este o metaclasă abstractă definită în subpachetul Common Behavior, fiind o specificare a unei instrucțiuni executabile care formează o abstractizare a unei proceduri de calcul, având drept rezultat o modificare a stării modelului, care poate fi realizat prin trimiterea unui mesaj către un obiect sau prin modificarea unei legături.
- **Legătura**, definită în subpachetul Common Behavior, reprezintă o instanță a unei relații de asociere şi defineşte o conexiune între instanțe.

O **maşină de stare** reprezintă o specificație care descrie succesiunile de stări ale unui obiect sau interacțiuni în timpul ciclului de viață, ca răspuns la evenimente, împreună cu răspunsurile sale la acele evenimente. Comportamentul este modelat ca o parcurgere a unui grafic al nodurilor de stare interconectate la una sau mai multe arcuri de tranziții asociate unor stări ale instanțelor de evenimente. În timpul acestei parcurgeri, maşina de stare execută o serie de acțiuni asociate diferitelor elemente ale maşinii de stare. Caracteristicile şi modul de comportare a unei maşini de stări este definit în subpachetul State Machines. La rândul ei, o maşină de stări implică un număr de alte elemente:

- **Starea** este o metaclasă abstractă care modelează o situație în timp ce altă condiție invariantă o susține. Totodată poate reprezenta o condiție a unui model dinamic, precum procesul realizării unei activități. În timpul execuției unui eveniment o stare poate fi activă (este o intrare ca rezultat a unei tranziții) sau inactivă (există ca rezultat al unei tranziții).
- **Tranziția** este o relație directă între un nod de stare sursă şi un nod de stare destinație. Ea poate fi parte a unei tranziții compuse, care duce maşina de stare dintr-o stare de configurare în alta, ca rezultat al unui eveniment particular.
- **Evenimentul** este generat ca rezultat al unei acțiuni fie din interiorul sistemului sau din mediul înconjurător sistemului.
- **Acțiunea** (ca răspuns la o tranziție).

Cele două elemente (interacțiunea şi maşina de stare) sunt conectate la diferite elemente structurale: clase, colaborări, obiecte.

Elementele de grupare reprezintă părțile organizaționale ale modelelor UML, definite în pachetul Model Management, care cuprinde următoarele elemente:

- **Pachetul** (package) este un mecanism cu scop general pentru organizarea elementelor în grupuri de elemente structurale, elemente comportamentale sau chiar alte elemente de grupare. Un pachet reprezintă de fapt o grupare a elementelor unui model.
- **Modelul** reprezintă o abstractizare a unui sistem fizic, cu un anumit scop, specificând sistemul fizic din diferite puncte de vedere ale abstractizării, el reprezentând de fapt o descriere completă a unui sistem dintr-o anumită perspectivă (particulară). Scopul avut în vedere va determina elementele care vor fi considerate (incluse în model) semnificative şi nesemnificative pentru model. Un model va conține o ierarhie de elemente care împreună descriu sistemul fizic. De asemenea, va conține şi un set de elemente care reprezintă mediul înconjurător al sistemului, cum ar fi actorii împreună cu interacțiunile lor, precum relațiile de dependență, generalizări şi constrângeri.
- **Subsistemul** este o grupare a elementelor de model care reprezintă un element comportamental al unui sistem fizic. Un subsistem oferă interfețe şi are operații proprii.

Elemente suplimentare, printre care enumerăm:

- Note – pentru formularea constrângerilor şi altor comentarii.
- Comentariu – este o notație ataşată unui element sau unui set de elemente ale modelului.
- Cerințe – pentru specificarea comportamentului dorit din perspectiva exterioară modelului.

Relaii UML

O **relație** reprezintă o conexiune între elementele modelului, fiind un termen convențional (abstract), fără specificații semantice. Relațiile suportate de UML sunt de: asociere, dependență, realizare (sau de flux), şi generalizare.

O **relație de asociere** este o relație structurală care descrie un set de legături, o legătură fiind o conexiune între obiecte. Asocierea declară o conexiune între instanțele clasificatorilor asociați, constând din cel puțin două terminații de asociere, fiecare specificând un clasificator conectat şi un set de proprietăți care trebuie să fie realizat pentru ca fiecare relație să fie validă (corectă). De asemenea, se poate specifica rolul pe care fiecare clasificator îl îndeplineşte în relația de asociere; multiplicitatea pentru fiecare capăt al asocierii, care va specifica câte (numărul) instanțe ale clasificatorului aflat la sfârşitul asocierii pot fi asociate cu o singură instanță a clasificatorului sursă (aflat la începutul asocierii). Unei relații de asociere i se poate ataşa şi un calificator, care reprezintă un atribut special care reduce multiplicitatea efectivă a relației de asociere, rezultând o asociere cu calificator.

În UML asocierile pot fi de trei feluri:

- Asociere ordinară – reprezintă relația de asociere cea mai utilizată, prin intermediul căreia doi calificatori sunt conectați.
- Agregare de compoziție (composite aggregate) – este o formă puternică a agregării. Ea presupune că o instanță (un obiect) poate face parte la un moment dat numai dintr-un singur obiect compus şi că obiectul compus este singurul responsabil pentru caracterul părților sale. Dacă obiectul compus este distrus, el trebuie să distrugă toate părțile sale componente.
- Agregare partajată (shared aggregate) – partea poate fi inclusă în mai multe agregări şi proprietățile sale se pot modifica în timp.

Agregarea este un tip special al relației de asociere, reprezentând o relație structurală între un întreg şi părți ale acestuia (relație parte-întreg), deci ea leagă o clasă ansamblu cu o clasă component. Numai asocierile binare pot fi agregări.

O **relație de dependență** este o relație semantică între două elemente prin care o modificare în unul din elemente, numit element independent, poate afecta semanticile celuilalt element, numit element dependent. O dependență este un termen convențional pentru o relație alta decât asocierea, generalizarea, realizarea (flux), sau alte metarelații (precum relația între un clasificator şi una din instanțele sale).

Într-un metamodel, o dependență este o relație directă între un client (sau clienți) şi un furnizor (sau furnizori), stabilind dependența clientului față de furnizor. Tipurile relațiilor de dependență suportate de UML sunt:

- Abstractizarea – este o relație de dependență care leagă două elemente, sau seturi de elemente, reprezentând acelaşi concept la nivele diferite de abstractizare, sau din puncte de vedere diferite.

- Legătura *(*binding*)* – o conexiune între elementele din cadrul aceluiaşi model. O legătură trebuie să aibă un furnizor (desemnează elementul care nu este afectat de o modificare) şi un client. Un element furnizor poate participa în mai multe relaţii de legătură către diferiţi clienţi. Un element client poate participa numai într-o singură relaţie de legătură cu un furnizor. O legătură este o relaţie de dependenţă unde furnizorul este un model şi clientul reprezintă instanţierea modelului care îndeplineşte substituirea parametrilor modelului.
- Utilizarea – este o relaţie în care un element are nevoie de un alt element (sau set de elemente) pentru implementarea sa completă. O utilizare este o relaţie de dependenţă în care clientul necesită prezenţa furnizorului, de exemplu: o clasă apelează o operaţie a unei alte clase (<<call>>); o metodă având un argument al altei clase (<<create>>); o metodă dintr-o clasă instanţiază altă clasă (<<instantiate>>).
- Permisiunea – este o relaţie care acordă unui element (client) dreptul să acceseze alte elemente (furnizori). Clientul primeşte permisiunea să facă referire la conţinutul furnizorului. Furnizorul trebuie să fie o asociere sau un clasificator.

O **relaţie de realizare** este o relaţie semantică între clasificatori, în care un clasificator specifică un contract a cărui realizare e garantată de un alt clasificator. Este o relaţie între două versiuni ale unui obiect sau între un obiect şi o copie a sa. Exemple de relaţii de realizare: între interfeţele şi clasele sau componentele care le realizează, între cazurile de utilizare şi colaborările prin care acestea se realizează.

O **relaţie de generalizare** este o relaţie de clasificare între mai multe elemente generale (un element generalizabil este un element care poate participa într-o relaţie de generalizare şi / sau de specializare) şi mai multe elemente specifice. O generalizare este o relaţie de generalizare-specializare în care obiectele elementului specializat (fiu sau copil) partajează structura şi comportamentul obiectelor elementului generalizat (părinte). Generalizarea între clase implică substituirea. Astfel, dacă o clasă este specificată ca root, ea nu poate fi o subclasă a altei clase (nu poate avea strămoşi). În mod similar, dacă ea este specificată drept clasă frunză (leaf), nici o altă clasă nu poate fi o subclasă a clasei (nu poate avea descendenţi).

În afara acestor relaţii, în UML mai sunt definite două tipuri de relaţii care sunt utilizate în conjunctură cu cazurile de utilizare (use cases):

- **Relaţia de extindere** (extend), definită în subpachetul Use Cases din pachetul Behavioral Elements, este o relaţie prin care comportamentul şi structura unui caz de utilizare poate fi îmbunătăţit cu comportamentul şi structura unui alt caz de utilizare. Relaţia are loc numai dacă condiţia care o însoţeşte este îndeplinită.
- **Relaţia de includere** (include), definită în subpachetul Use Cases din pachetul Behavioral Elements, semnifică faptul că un caz de utilizare conţine comportament definit în alt caz de utilizare. Relaţia de includere este o relaţie între două cazuri de utilizare.

5.2.4. Diagramele UML

Înainte de a trece la prezentarea diagramelor utilizate de UML, trebuie specificat că există trei perspective care trebuie utilizate în desemnarea diagramelor (sau a oricărui model) [Cook&Daniels, 1994]:

- Conceptuală: în acest caz se utilizează o diagramă care reprezintă conceptele din domeniul sistemului supus analizei. Aceste concepte vor avea legătură cu clasele pe care le implementează, dar nu este o reprezentare directă. Modelul este proiectat fără a ţine seama de software-ul care îl va implementa şi este în general independent de limbaj.
- Specificaţia: în acest moment atenţia este îndreptată către software, dar de fapt către interfeţele software-ului şi nu către implementarea propriu zisă. Dezvoltarea orientată obiect pune un mare accent pe diferenţa între interfaţă (tip) şi implementare (clasă). Tipurile reprezintă o interfaţă care

poate avea mai multe implementări datorită mediului de implementare, caracteristicilor de performanță etc.

- **Implementarea**: în momentul implementării se utilizează clasele.

După cum s-a arătat în versiunii UML 2.0 există 13 diagrame (în loc de 9), împărțite în 3 categorii:

1. **Diagrame de structură**: descriu structura statică a aplicației.
2. **Diagrame de comportament**: descriu aspecte diferite ale comportamentului dinamic.
3. **Diagrame de interacțiune**: reprezintă un subset al diagramelor de comportament care accentuează interacțiunile dintre obiecte.

Diagrame	Descriere
1. Diagrama de activitate	Ilustrează procesele de afaceri, incluzând fluxul de date.
2. Diagrama claselor	Arată o colecție de elemente ale modelului static, precum clasele şi tipul lor.
3. Diagrama de comunicație	Arată instanțe ale claselor, relațiile de interacțiune dintre ele, şi fluxul mesajelor dintre ele. Diagrama se concentrează pe organizarea structurală a obiectelor care trimit şi primesc mesaje. La vechile versiuni era denumită diagrama de colaborare.
4. Diagrama de componente	Ilustrează componentele care compun întreprinderea, sistemul, sau aplicația.
5. Diagrama de structură compusă	Ilustrează structura internă a unui clasificator (clasă, componentă, caz de utilizare), inclusiv punctele de interacțiune ale clasificatorului cu alte părți ale sistemului.
6. Diagrama de aplicație	Arată execuția arhitecturii de sistem. Aceasta include noduri, componente hardware sau medii de execuție software, precum şi produse middleware.
7. Diagrama de concluzionare a interacțiunilor (interaction overview)	O variantă a diagramei de activitate care face o privire de ansamblu a fluxului de control în interiorul sistemului sau procesului de afaceri. Fiecare nod / activitate din interiorul diagramei poate reprezenta altă diagramă de interacțiune.
8. Diagrama de obiecte	Ilustrează obiectele şi relațiile lor la un anumit moment de timp, în mod normal un caz special fie a diagramei claselor sau a diagramei de comunicație.
9. Diagrama de pachete	Arată cum elementele de model sunt organizate în pachete, precum şi dependențele dintre pachete.
10. Diagrama secvențială	Modelează logica secvențială, în fapt ordonarea în timp a mesajelor dintre clasificatori.
11. Diagrama maşinilor de stare	Descrie stările unui obiect, precum şi tranzițiile dintre stări.
12. Diagrama de sincronizare (timing)	Ilustrează modificarea în stare sau condiție a instanței unui clasificator sau rol în timp. Este utilizată pentru a arăta modificarea stării unui obiect în timp ca răspuns la evenimente externe.
13. Diagrama cazurilor de utilizare	Arată cazurile de utilizare, actorii, şi relațiile dintre ele.

O diagramă poate conține colecții ale următoarelor tipuri de obiecte: actori (diagrama cazurilor de utilizare), clase (diagrama claselor), componente (diagrama componentelor sau de aplicație), noduri (diagrama de aplicație), pachete (diagrama claselor sau pachetelor), cazuri de utilizare (diagrama cazurilor de utilizare), stări (diagrama de activitate sau de stare), obiecte (diagrama obiectelor, de colaborare, secvențială, componentelor, de aplicație, sau de activitate), semnale (diagrama de activitate), asocieri (diagrama claselor, cazurilor de utilizare, sau obiectelor), generalizări (diagrama claselor, pachetelor, sau cazurilor de utilizare), dependențe (diagrama claselor, cazurilor de utilizare, componentelor, sau de aplicație), tranziții (diagrama de activitate sau de stare), interacțiuni (diagrama de colaborări), mesaje (diagrama secvențială) etc.

Diagrama Claselor (Class Diagram)

Diagrama claselor reprezintă tehnica centrală de modelare pe care se bazează toate modelele orientate obiect. Diagrama claselor este un grafic al clasificatorilor conectați prin relații statice care există între clasificatori precum asocierea, compoziția (agregarea de compoziție), delegația (permisiunea) și generalizarea. O astfel de diagramă mai poate conține interfețe, pachete, relații, și chiar instanțe, precum obiecte și legături. Un nume mai corect pentru această diagramă ar fi "diagrama de structură statică".

Obiectivul principal al acestei diagrame este să definească clasele de bază împreună cu caracteristicile utilizate în alte diagrame dinamice.

Diagrama claselor este utilizată pentru:

- Modelarea clasele de bază (caracteristicile arătate în alte diagrame dinamice).
- Modelarea colaborărilor (specificarea claselor și a relațiilor acestora).
- Modelarea schemei logice a bazei de date.
- Modelarea vocabularului sistemului (specificarea abstracțiilor și a responsabilităților acestora).

Diagrama claselor poate fi implementată ca o clasa actuală, care este utilizată în limbajele suportate de clasă, și care cuprinde: *clase*, *interfețe*, *note și legarea notelor*, *text*, *relații de dependență*, *de generalizare*, *de asociere*, *de realizare* și *de agregare* (agregarea de compoziție), *colaborări*.

Noțiunile de clasă, interfață, colaborare și relațiile utilizate au fost explicate în subcapitolele precedente. Pentru înțelegerea corectă a acestei diagrame mai este necesară explicarea altor concepte utilizate în proiectarea ei. Ceea ce trebuie subliniat este faptul că un **clasificator** declară o colecție de caracteristici, precum: atribute, metode, și operații. Numele fiecărui clasificator este unic și reprezintă o metaclasă abstractă.

Un **atribut** reprezintă un nume din interiorul unui clasificator și care descrie un domeniu de valori pe care le poate lua instanțele clasificatorului, putând avea o valoare inițială. De asemenea, un atribut poate avea una din următoarele proprietăți, care se referă la posibilitatea de modificare a valorii, după ce obiectul a fost creat:

- changeable (modificabil) – în acest caz nu este impusă nici o restricție asupra valorii atributului;
- addOnly - valabil numai pentru atributele cu multiplicitate mai mare decât unu; o valoarea odată adăugată nu mai poate fi ștearsă sau modificată;
- frozen - valoarea atributului nu poate fi modificată după inițializarea obiectului.

O **operație** este un serviciu care poate fi cerut de către un obiect pentru a efectua un comportament; ea este o semnătură, care descrie parametrii actuali care sunt posibili (inclusiv posibile valori returnate). O operație poate fi polimorfică, ceea ce înseamnă că, într-o ierarhie de clase, pot fi specificate operații cu aceeași semnătură la nivele diferite în ierarhie dar cu implementări diferite.

O operație poate avea una din următoarele proprietăți (se referă la simultaneitatea apelurilor concurente către aceeași clasă pasivă – este o instanță a unui clasificator cu atributul isActive = False):

- isQuery – se referă şi la o metodă (operaţia şi metoda reprezintă caracteristici dinamice ale unui element al modelului) şi arată dacă o execuţie a unei operaţii, sau a unei metode, schimbă sau nu starea sistemului. Dacă are valoarea True starea sistemului rămâne neschimbată;
- sequential (secvenţială) – apelurile trebuie coordonate în aşa fel încât numai unul poate apela o instanţă (pe orice operaţie secvenţială), la un moment dat; în caz contrar semantica şi integritatea sistemului nu pot fi garantate;
- guarded (prudentă) – apeluri multiple se pot produce simultan către o instanţă, dar numai un apel este permis să înceapă; celelalte operaţii sunt blocate până când realizarea primei operaţii este terminată complet. O operaţie de acest tip este utilizată pentru secvenţierea apelurilor în cazul fluxurilor multiple ale controlului;
- concurrent (concurentă) – apeluri multiple se pot produce simultan către o instanţă, fiecare dintre ele putând fi realizate în paralel.

O operaţie poate avea şi parametri care pot fi utilizaţi în specificarea operaţiilor, semnalelor, mesajelor, evenimentelor etc., şi fiecare parametru include un nume, tip şi direcţia comunicării. Pentru fiecare parametru al unei operaţii se poate specifica direcţia astfel:

- in – reprezintă un parametru de intrare care nu poate fi modificat;
- out – reprezintă un parametru de ieşire, care poate fi modificat pentru a comunica informaţii către alte elemente;
- inout – reprezintă un parametru de intrare care poate fi modificat;
- return – reprezintă o valoare returnată unui apel.

O **metodă** reprezintă o implementare a unei operaţii, şi specifică algoritmul sau procedura care produce efectele unei operaţii.

Unul dintre cele mai importante detalii care poate fi specificat pentru atributele şi operaţiile unui clasificator este vizibilitatea acestora. Vizibilitatea unei caracteristici (atribut sau operaţie) specifică când aceasta poate fi, sau nu, utilizată de un alt clasificator.

În UML se pot specifica următoarele nivele de vizibilitate:

- public (public) – orice alt clasificator, care are vizibilitate, poate utiliza atributul sau operaţia respectivă;
- protected (protejat) – orice descendent al clasificatorului poate utiliza atributul sau operaţia;
- private (privat) – numai clasificatorul însuşi poate utiliza atributul sau operaţia.

Un alt detaliu care poate fi specificat drept o caracteristică (atribut sau operaţie) a unui clasificator este **scopul** (scope), care poate fi:

- instance (instanţă) – fiecare instanţă a clasificatorului conţine propria valoare a caracteristicii;
- classifier (clasificator) – există o singură valoare pentru toate instanţele unui clasificator.

Multiplicitatea clasei specifică numărul de instanţe ale clasei. Multiplicitatea se aplică şi la nivel de atribut.

Pentru relaţiile de dependenţă (abstractizarea, legătura, utilizarea şi permisiunea) între clasele şi obiectele din diagrama claselor se pot defini următoarele stereotipuri:

- Relaţia de abstractizare:
  - «derive» - specifică o relaţie de derivare de-a lungul elementelor unui model care sunt în mod normal de acelaşi tip; se poate utiliza când se doreşte modelarea relaţiei între două atribute sau două asocieri, dintre care una este concretă iar cealaltă este conceptuală;
  - «realize» - specifică o relaţie de realizare între un element (sau elemente) numit furnizor (respectiv furnizori) şi un alt element (sau elemente) numit client (respectiv clienţi) pe care le implementează;

- «refine» - specifică o relație de rafinare între elementele modelului la nivele semantice diferite, precum analiza şi proiectarea (sursa este la un grad mai fin de abstractizare decât ținta);
- «trace» - reprezintă o relație de urmărire între elementele modelului, sau seturi de elemente, care reprezintă acelaşi concept în modele diferite;

♦ Relația de permisiune*:*
- «access» - este un tip de relație de dependență între două relații de asociere sau doi clasificatori, indicând că părțile publice ale destinației sunt accesibile sursei pachetului;
- «friend» - este un tip de relație de dependență a cărei sursă este un element, precum o operație, clasă, sau pachet şi a cărui destinație este un element dintr-un pachet diferit, precum o operație, o clasă, sau pachet; acordă accesul sursei la destinație indiferent de modul de declarare a vizibilității;
- «import» - este un tip de relație de dependență între două asocieri, sau doi clasificatori, indicând că părțile publice ale destinației sunt adăugate sursei.

♦ Relația de utilizare:
- «call» - sursa este o operație şi destinația este tot o operație;
- «create» - clasificatorul client creează instanțe ale clasificatorului furnizor;
- «instantiate» - operațiile pentru client creează instanțe ale furnizorului;
- «send» - sursa este o operație şi destinația este un semnal (sursa trimite un semnal).

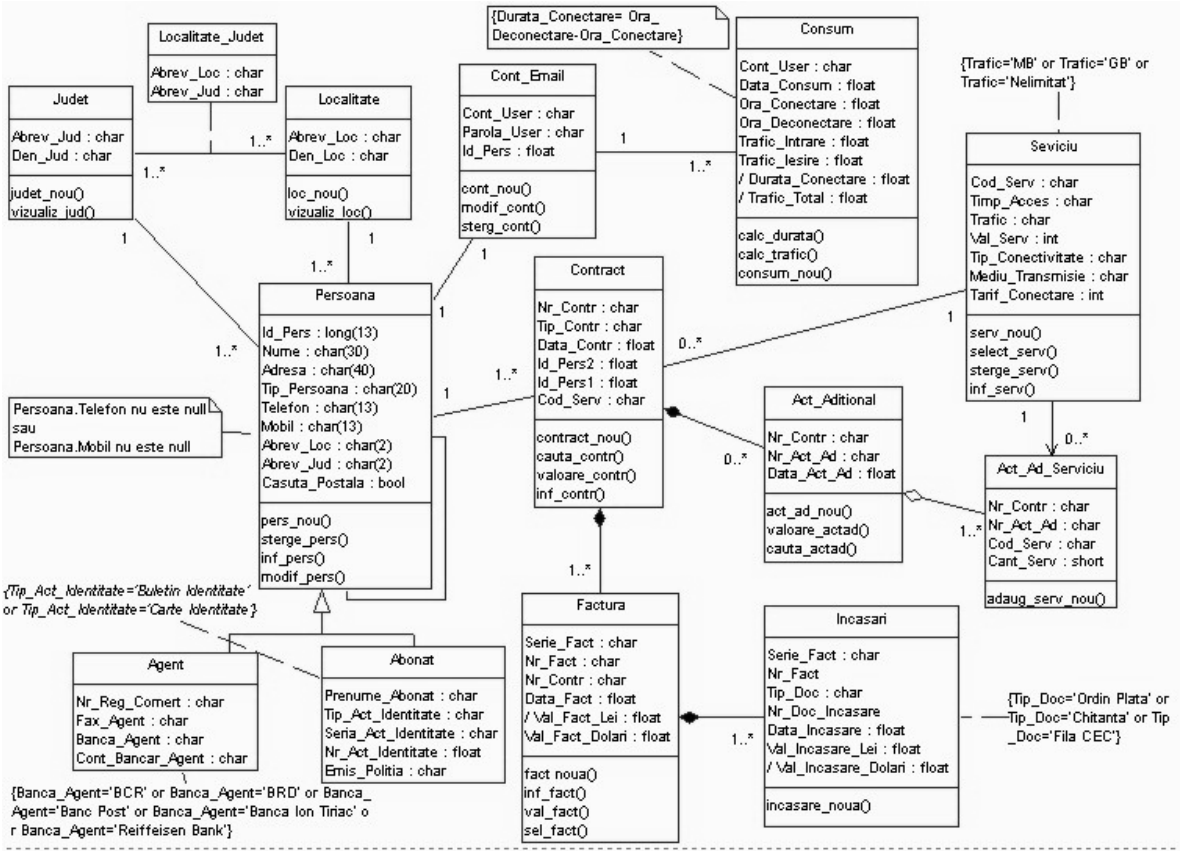


Fig. 5.17. – Diagrama claselor

Pentru relația de asociere mai pot fi specificate:

♦ Direcția de navigare de la obiectele de un tip la obiecte de alt tip.

♦ Vizibilitatea asocierii – specifică vizibilitatea unui sfârşit al asocierii din punctul de vedere al clasificatorului de la celălalt capăt al asocierii; valorile posibile sunt: public, protected şi private.

- Calificator – reprezintă un atribut al asocierii ale cărui valori partiționează setul de obiecte legate la un obiect printr-o asociere; dacă nu are nici o valoare relația de asociere nu este calificată.
- Specificator de interfață - pentru limitarea interfeței la contextul asocierii.

De asemenea, UML permite definirea de:

- Clase de asocieri – reprezintă o asociere care este de asemenea o clasă, ea nu poate exista în afara asocierilor care o determină.
- Constrângeri care este o condiție semantică sau o restricție exprimată în text prin intermediul unui limbaj natural. Unele constrângeri sunt definite de UML, altele pot fi definite de utilizator. Constrângerea este o afirmare şi nu un mecanism executabil. Ea indică o restricție care trebuie să fie impusă pentru descrierea corectă a unui sistem. Constrângerile pot fi aplicate atât relației de asociere cât şi atributelor unei clase.
- Interfețe. O interfață este o colecție de operații utilizate pentru a specifica un serviciu al unei clase. Interfața poate participa la relații de dependență, de asociere, de generalizare şi de realizare.

Diagrama Obiectelor (Object Diagram)

Diagrama obiectelor evidențiază un set de obiecte şi relațiile cu alte obiecte la un moment dat şi poate fi considerată un caz special al diagramelor claselor sau al diagramei de colaborări, fiind de fapt o instanță a unei diagrame de clase, ea fiind un instantaneu al unei stări al unui sistem la un moment dat. Ea conține: obiecte, legături, eventual note şi constrângeri. Această diagramă este utilizată pentru vizualizarea, specificarea, construirea şi documentarea structurii obiectelor.

Modelarea structurii obiectelor presupune:

- Identificarea mecanismului de modelat (o anumită funcție sau comportamentul unei părți a sistemului).
- Pentru fiecare mecanism, se identifică clasele, interfețele şi alte elemente care participă la această colaborare.
- Se consideră un scenariu prin acest mecanism; se determină fiecare obiect care participă la mecanism.
- Selectarea obiectelor care au responsabilități de nivel înalt pentru fluxul de lucrări (workflow).
- Identificarea precondițiilor stărilor inițiale şi postcondițiilor stărilor finale.
- Specificarea activităților şi acțiunilor începând cu starea inițială.
- Evidențierea tranzacțiilor care conectează aceste activități şi acțiuni.
- Eventual, evidențierea obiectelor importante implicate în fluxul de lucrări, cu evidențierea schimbării valorilor.

Modelarea operațiilor obiectelor presupune:

- Colectarea abstracțiilor implicate în operații (parametri, atribute ale claselor).
- Identificarea precondițiilor stării inițiale şi postcondițiilor stării finale.
- Specificarea activităților şi acțiunilor începând cu starea inițială.
- Folosirea ramificării dacă este necesar.
- Folosirea bifurcării şi reunirii pentru specificarea fluxurilor de control paralele.

Diagrama cazurilor de utilizare (Use Case Diagram)

Această diagramă este utilizată în faza de analiză a sistemului şi a cerințelor utilizatorului şi arată actorii (utilizator sistem sau sistem extern) şi cazurile de utilizare ale sistemului împreună cu relațiile lor. Cazurile de utilizare reprezintă funcționalitatea unui sistem sau a unui clasificator, precum un subsistem sau o clasă. Prin utilizarea acestei diagrame, utilizatorii şi dezvoltatorii pot uşor adăuga şi modifica cerințele utilizatorului.

Diagrama cazurilor de utilizare este un graf al actorilor, un set al cazurilor de utilizare, posibil unele interfețe, şi relațiile dintre aceste elemente. Relațiile sunt de **asociere** între actori şi cazurile de utilizare, **generalizări** între actori şi **generalizări**, **extensii**, şi **includeri** între cazuri de utilizare. Cazurile de utilizare pot fi opțional incluse într-un dreptunghi care va reprezenta granița (interfața) sistemului conținut.

Diagrama cazurilor de utilizare se foloseşte pentru modelarea aspectelor dinamice ale sistemului şi anume pentru modelarea comportamentului unui sistem, unui subsistem sau unei clase. O astfel de diagramă conține: **cazuri de utilizare**, **actori**, **relații de dependență**, **de generalizare** şi **de asociere** şi se utilizează pentru **modelarea contextului sistemului** (specificarea actorilor şi înțelegerea rolului lor) şi pentru **modelarea cerințelor sistemului** (CE trebuie să facă sistemul, şi nu CUM).

Modelarea contextului sistemului presupune:

- Identificarea actorilor.
- Organizarea actorilor într-o ierarhie de generalizare / specializare.
- Populareа diagramei cu actorii identificați şi specificarea căilor de comunicare între fiecare actor şi cazurile de utilizare ale sistemului.

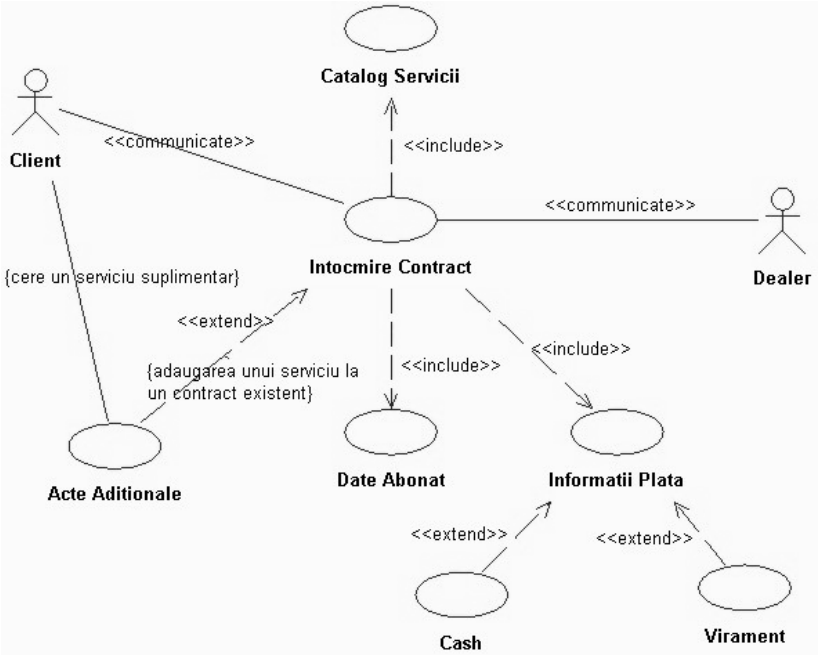


Fig. 5.18. – Diagrama cazurilor de utilizare

Modelarea cerințelor sistemului presupune:

- Stabilirea contextului sistemului prin identificarea actorilor.
- Stabilirea pentru fiecare actor a comportamentului cerut sau aşteptat din partea sistemului.
- Specificarea cazurilor de utilizare pe baza comportamentelor comune.
- Identificarea de noi cazuri de utilizare care sunt utilizate de altele.
- Modelarea cazurilor de utilizare, a actorilor şi a relațiilor între ei într-o diagramă a cazurilor de utilizare.

O diagramă a cazurilor de utilizare bine structurată este acea diagramă care:

- Este centrată pe comunicarea unui aspect al viziunii statice a cazurilor de utilizare ale sistemului.
- Conține numai acele cazuri de utilizare şi actori care sunt esențiali pentru înțelegerea aspectului analizat.
- Prevede detalii consistente cu nivelul de abstractizare (numai ceea ce este esențial pentru înțelegere).

Diagramele de Interacțiuni (Interaction Diagrams)

Diagrama de interacțiuni reprezintă un termen generic care este aplicat mai multor tipuri de diagrame care subliniază interacțiunea dintre obiecte. Diagramele de interacțiuni sunt utilizate în cazul în care se doreşte reflectarea comportamentului mai multor obiecte dintr-un singur caz de utilizare. În funcție de criteriul care se are în vedere, aceste diagrame sunt de două tipuri:

- Diagrama secvențială care evidențiază ordonarea în timp (secvențialitatea) a mesajelor.
- Diagrama de colaborare evidențiază organizarea structurală a obiectelor care trimit şi primesc mesaje (arată relațiile dintre instanțe).

Diagramele de interacțiune conțin: obiecte, legături, mesaje şi eventual constrângeri.

Diagrama secvențială (Sequence Diagram)

Arată fluxul dinamic al mesajelor diferitelor obiecte din sistem. Obiectivul principal al acestei diagrame este acela de a exprima fluxul de mesaje al obiectelor în timp secvențial. Diagrama secvențială arată interacțiunea obiectelor aranjate în timp secvențial. În particular, ea arată obiectele care participă la o interacțiune şi succesiunea mesajelor care sunt schimbate. O diagramă secvențială poate exista într-o formă generică (descrie toate scenariile posibile) şi într-o formă de exemplu (descrie un scenariu actual).

Modelarea fluxului controlului prin ordonarea în timp a mesajelor presupune:

- Stabilirea contextului interacțiunii (sistem, subsistem, operație, sau clasă, sau un scenariu al unui caz de utilizare sau colaborare).
- Identificarea obiectelor care au un rol în acțiune.
- Stabilirea pentru fiecare obiect a faptului dacă persistă prin întreaga interacțiune; pentru obiectele create şi distruse în timpul interacțiunii trebuie să se indice explicit, prin mesaj, acest lucru.
- Pentru fiecare mesaj începând cu primul (care inițiază acțiunea) şi continuând cu celelalte în ordinea succesiunii, se prezintă proprietățile (parametrii).
- Eventual, timpul şi spațiul cerut, pentru fiecare mesaj.
- Eventual, precondiții sau postcondiții pentru fiecare mesaj.

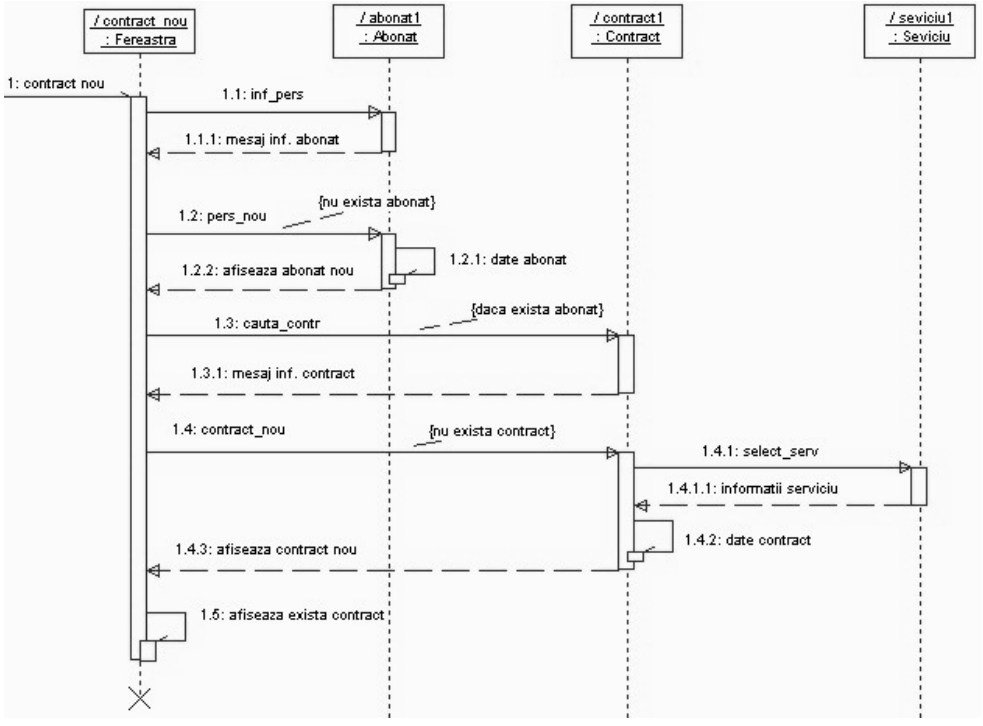


Fig. 5.19. – Diagrama secvențială

Pentru a evidenția fluxul complet al controlului se pot utiliza mai multe diagrame secvențiale.

Diagrama secvențială are două dimensiuni: una verticală, care reprezintă timpul şi una orizontală, care reprezintă diferite obiecte.

Diagrama de colaborare (Collaboration Diagram)

La fel ca şi diagrama secvențială, diagrama de colaborare arată caracteristica dinamică a sistemului. Amândouă diagramele sunt identice, în ceea ce priveşte informația internă a diagramei, dar diagrama de colaborare pune un accent mai mare pe colaborarea dintre obiectele sistemului decât pe succesiunea în timp, specifică diagramei secvențiale. Deci în funcție de obiectivul urmărit, se va alege una din următoarele diagrame: secvențială sau de colaborare.

Diagrama de colaborare arată interacțiunile din interiorul structurii unui model, utilizând fie clasificatori şi asocieri sau instanțe şi legături. O diagramă de colaborare prezintă o colaborare, care conține un set de roluri ale obiectelor, sau poate prezenta o interacțiune, care defineşte un set de mesaje specifice ei.

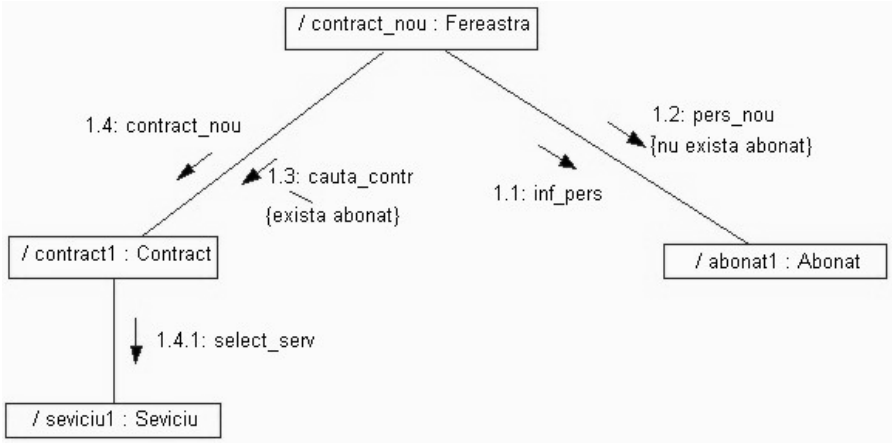


Fig. 5.20. – Diagrama de colaborare

Modelarea controlului prin organizare presupune:

- Stabilirea contextului interacțiunii (sistem, subsistem, operație, sau clasă, sau un scenariu al unui caz de utilizare sau colaborare).
- Identificarea obiectelor care joacă rol în acțiune.
- Stabilirea proprietăților inițiale pentru fiecare obiect; dacă starea sau rolul unui obiect se schimbă semnificativ în timpul interacțiunii, se plasează un obiect duplicat în diagramă care se actualizează cu noile valori şi se conectează printr-un anumit mesaj.
- Specificarea legăturilor între obiecte prin care pot trece mesajele: legăturile asocierii sau alte legături (global, local).
- Se începe cu mesajul care inițiază interacțiunea şi se ataşează fiecare mesaj subsecvent conform legăturilor între obiecte.
- Eventual, timpul şi spațiul cerut, pentru fiecare mesaj.
- Eventual, precondiții sau postcondiții pentru fiecare mesaj.

Diagrama de Stare

Diagrama de stare este una din diagramele prin intermediul cărora se evidențiază realizarea cazurilor de utilizare. Este folosită în situația în care se doreşte modelarea unui obiect care este utilizat în mai multe cazuri de utilizare. Se foloseşte pentru modelarea comportamentului unei interfețe, unei clase sau unei colaborări şi evidențiază comportamentul orientat-eveniment al unui obiect. Arată starea internă şi modificarea stărilor obiectelor clasei specificate, fiecare stare posibilă a obiectului putând fi exprimată cu ajutorul acestei diagrame.

O diagramă de stare este un graf care reprezintă o maşină de stări arătând fluxul controlului de la o stare la alta şi va conține: stări, maşini de stare, tranziții. Diagrama de stare se utilizează pentru modelarea unui anumit aspect dinamic al sistemului în contextul sistemului ca întreg, unui subsistem sau unei clase. Se poate folosi pentru modelarea unui scenariu din diagrama cazurilor de utilizare.

O maşină de stări este un comportament care arată succesiunea de stări prin care trece un obiect pe durata sa de viață ca răspuns la evenimente, precum şi răspunsurile la aceste evenimente. În mod normal, această diagramă este utilizată pentru descrierea comportamentului claselor, dar poate de asemenea să descrie şi comportamentul altor entități precum: cazuri de utilizare, actori, subsisteme, operații sau metode.

Diagrama de stare se foloseşte pentru modelarea obiectelor reactive şi presupune:

- Selectarea contextului (clasă, caz de utilizare, sistem).
- Selectarea stărilor inițiale ale obiectelor.
- Ordonarea parțială a stărilor stabile în durata de viață a obiectelor.
- Decide evenimentele care pot declanşa o tranziție de la o stare la alta.
- Ataşează acțiuni la aceste tranziții şi/sau la aceste stări.
- Verifică dacă toate stările pot fi atinse printr-o combinație de evenimente.
- Verifică secvențele de evenimente aşteptate şi răspunsurile lor.

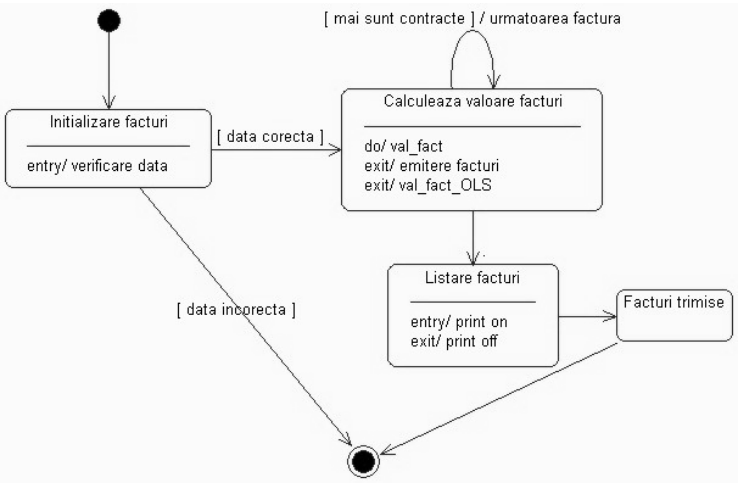


Fig. 5.21. – Diagrama de stare

Diagrama de Activitate (Activity Diagram)

Cu ajutorul acestei diagrame este exprimată tranziția activităților. Este utilizată în special atunci când este necesară evidențierea legăturii fiecărui serviciu sau a mai multor procese paralele.

Diagrama de activitate este o diagramă de tip flowchart (organigramă) care evidențiază fluxul controlului de la o activitate la alta. O activitate rezultă dintr-o anumită acțiune (apelul altei operații, trimiterea unui semnal, crearea sau distrugerea unui obiect) sau evaluarea unei expresii care schimbă starea sistemului sau întoarce o valoare. Diagrama de activitate este o variantă a unei maşini de stări în care stările reprezintă performanța acțiunilor sau subactivităților.

Diagrama de activitate conține: stări ale activităților şi stări ale acțiunilor, tranziții, obiecte.

Stările activităților pot fi descompuse şi necesită un interval de timp pentru a fi îndeplinite, comparativ cu stările acțiunilor care necesită timp de execuție nesemnificativ şi nu se pot descompune. Pentru

obiectele implicate în fluxul controlului se poate evidenția rolul, starea şi schimbarea valorilor atributelor.

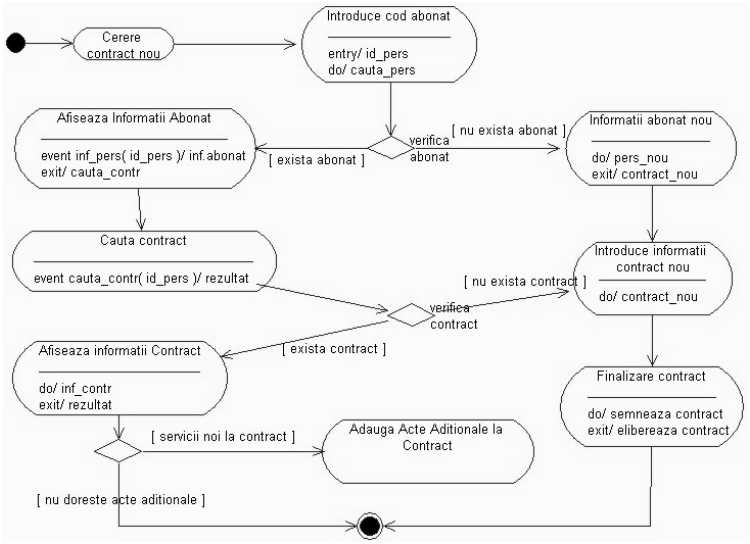


Fig. 5.22. – Diagrama de activitate

Diagrama de activitate poate fi utilizată pentru:

- Modelarea *workflow* (activități aşa cum sunt văzute de actorii care colaborează cu sistemul). Această modelare presupune:
  - selectarea obiectelor care au responsabilități de nivel înalt pentru workflow;
  - identificarea precondițiilor stărilor inițiale şi postcondițiilor stărilor finale;
  - specificarea activităților şi acțiunilor începând cu starea inițială;
  - evidențierea tranzițiilor care conectează aceste activități şi acțiuni;
  - eventual, evidențierea obiectelor importante implicate în workflow, cu evidențierea schimbării valorilor.
- Modelarea operațiilor – *flowchart*. Această modelare presupune:
  - colectarea abstracțiilor implicate în operații (parametrii, atribute ale claselor);
  - identificarea precondițiilor stării inițiale şi postcondițiilor stării finale;
  - specificarea activităților şi acțiunilor începând cu starea inițială;
  - folosirea ramificării dacă este necesar;
  - folosirea bifurcării şi reunirii pentru specificarea fluxurilor de control paralele.

Diagramele de Implementare

Diagramele de implementare arată aspecte ale implementării, incluzând cod sursă şi implementarea run-time. Aceste diagrame se prezintă sub două forme:

- Diagrama de componente. Diagrama de componente arată: structura codului, structura fizică a codului bazat pe componenta de cod actuală. Diagrama de componente defineşte care clase vor fi incluse în mai multe fişiere şi care fişiere vor fi incluse în mai multe module. Totodată ea arată dependențele componentelor software, incluzând codul sursă al componentelor, codul binar al componentelor, şi componentele executabile.
- Diagrama de aplicație. Arată structura fizică a sistemului hardware şi software. Această diagramă arată configurarea elementelor de procese run-time şi componente software, procese şi obiecte.

Utilizarea diagramelor UML depinde de perspectiva din care este analizat sistemul software. Deoarece în dezvoltarea sistemului sunt implicate persoane cu calificări diferite (utilizatori finali, analişti, proiectanți, programatori, etc.), arhitectura unui sistem poate fi descrisă utilizând: viziunea cazurilor de utilizare, viziunea proiectării, viziunea proceselor, viziunea implementării. Pentru fiecare din aceste

viziuni se pot folosi anumite diagrame pentru evidențierea aspectelor statice şi anumite diagrame pentru evidențierea aspectelor dinamice, la nivelul de detaliu cerut de fiecare caz în parte.

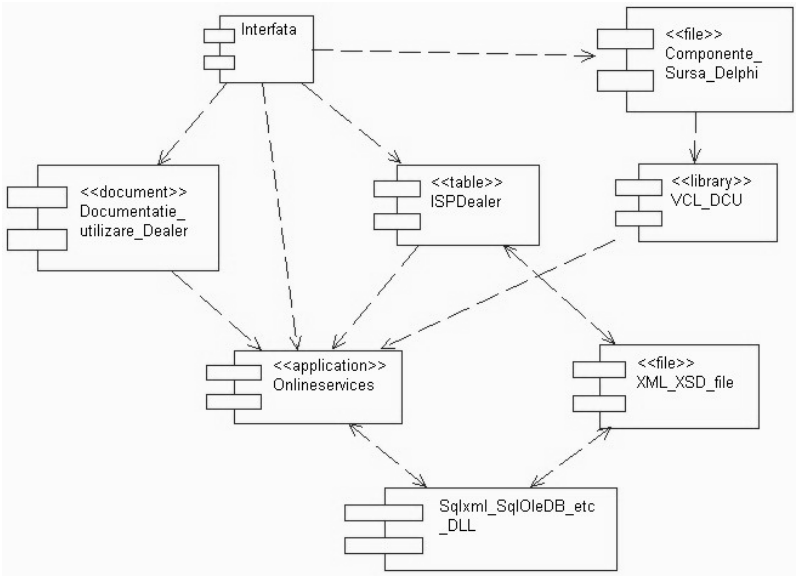


Fig. 5.23. – Diagrama de componente

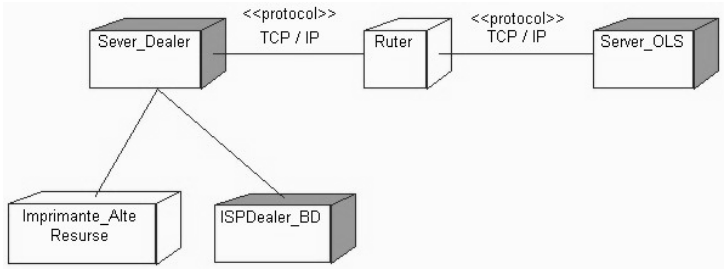


Fig. 5.24. – Diagrama de aplicație

Perspectivele din care este analizat sistemul şi numărul de diagrame utilizate depind de gradul de complexitate al sistemului şi de natura acestuia.

De exemplu, pentru modelarea unei aplicații simple, implementabile pe un singur calculator, sunt necesare numai primele două viziuni. [Vătui, 2000]

Tabelul nr.1 – Diagramele utilizate în cazul unei aplicații simple

Aspectele statice	Aspectele dinamice	Viziunea
Diagrama cazurilor de utilizare		Cazurilor de utilizare
Diagrama claselor	Diagrama interacțiunilor	Proiectării

Pentru sisteme complexe, implementările în mediu distribuit, se pot utiliza toate tipurile de diagrame.

Principalele îmbunătățiri aduse de UML față de celelalte modele prezentate sunt următoarele:

- Separarea relațiilor structurale (asociere, generalizare, agregare) de relațiile semantice (relații de dependență, de realizare).
- Simplificarea notațiilor pentru relațiile semantice utilizând aceeaşi simbolizare cu etichete diferite în funcție de tipul relației.
- Extinderea noțiunii de clasificatori (folosită de toate metodele analizate pentru conceptul de obiect) pentru următoarele concepte: interfață, tip de date, semnal, componentă, nod, caz de utilizare, subsistem.

- Utilizarea aceluiaşi tip de diagramă în etape diferite ale dezvoltării sistemului.
- Oferirea de recomandări pentru întocmirea diagramelor.
- Oferirea de criterii de verificare a corectitudinii modelării.

Pentru a uşura munca de analiză şi proiectare a sistemelor, precum şi pentru o analiză şi o planificare mai corectă a procesului de dezvoltare a produsului software, diferite firme, precum Microsoft, Rational sau OMG au început să investească tot mai mult în activitățile legate de metodologiile de analiză şi proiectare de software.

Aceste instrumente, numite **instrumente de modelare vizuale orientate obiect** (**OOVMT** – **O**bject **O**riented **V**isual **M**odeling **T**ool), sunt utilizate pentru automatizarea elaborării diagramelor din cadrul diferitelor metodologii utilizate în proiectarea sistemelor informatice.

În momentul de față majoritatea firmelor mari utilizează UML drept limbaj de modelare pentru sistemele software. Din acest motiv s-a simțit necesitatea ca acesta să fie înglobat într-un proces unificat care să furnizeze o modalitate de elaborare riguroasă a sarcinilor, ținând cont de faptul că se pune un mare accent pe lucru în echipă, mai ales pentru proiectele mari. Deoarece UML nu este complet (reprezintă o notație de modelare standard industrială), mai sunt utilizate şi alte tehnici suplimentare, care includ:

- Modelarea interfeței utilizator (diagrame de control a interfeței utilizator).
- Modelarea datelor.
- Reguli de afaceri.
- Constrângeri etc.

Capitolul 6
Metode de proiectare a bazelor de date

Avuția cea mai importantă pentru o organizație o reprezintă informația. Fără existența informațiilor o organizație nu ar mai putea exista, iar pentru a supraveți aceste informații trebuiesc utilizate corect, pentru a se putea obține rezultate care să conducă la alegerea celei mai bune soluții pentru prosperitatea acesteia. Pentru ca acest lucru să fie posibil de realizat, aceste informații trebuie stocate, triate (filtrate) şi analizate în funcție de necesitățile apărute într-un anumit moment dat. Cea mai bună alegerea pentru ca aceste informații să fie coerente, şi să nu fie pierdute, sau distruse, în mod intenționat sau nu, este stocarea acestora în diverse baze de date. De reținut este citatul lui Alvin Tofler „…, iar informația acumulată, o parte a avuției naționale".

Se ştie că un sistem informațional cuprinde totalitatea resurselor care permit colectarea, administrarea, controlul şi propagarea informațiilor într-o organizație. Atunci când prelucrarea datelor se efectuează prin intermediul calculatoarelor, acest sistem va cuprinde următoarele componente:

- baza de date;
- elementele software ale bazei de date;
- software de aplicație;
- elementele hardware ale calculatorului;
- personalul care utilizează şi dezvoltă sistemul.

Dintre aceste componente *baza de date constituie componenta fundamentală a sistemului informațional*, iar dezvoltarea şi utilizarea să trebuie privite din perspectiva cerințelor mai largi ale organizației. Prin urmare, ciclul de viață al unui sistem informațional aferent unei organizații este inerent legat de ciclul de viață al sistemului de baze de date care îl susține, şi de asemenea, ciclul de viață al aplicației de tip bază de date este inerent legat de ciclul de viață al sistemului informațional.

Ciclul de viață al unui sistem informațional implică existența mai multor faze:

- studiu de fezabilitate;
- analiza cerințelor;
- proiectarea conceptuală a datelor şi operațiilor;
- proiectarea logică;
- proiectarea externă;
- realizarea de prototipuri;
- proiectarea internă şi implementarea;
- testarea şi validarea;
- întreținerea (mentenanța);

iar etapele principale ale ciclului de viață ale unei aplicații de tip bază de date sunt [Conn]:

- planificarea bazei de date;
- definirea sistemului;
- colectarea şi analiza cerințelor;
- proiectarea bazei de date;
- alegerea sistemului SGBD (opțional);
- proiectarea aplicației;
- realizarea prototipului (opțional);
- implementarea;
- conversia şi implementarea datelor;
- întreținerea operațională.

Etapa de proiectare a bazei de date reprezintă procesul de realizare a unui proiect pentru o bază de date, care va trata toate operațiile şi obiectivele organizației. *Calitatea unei aplicații de baze de date depinde de proiectarea sa.*

Scopurile etapei de proiectare a bazei de date sunt:

- reprezentarea datelor şi a relațiilor dintre ele, necesare tuturor domeniilor de aplicație şi grupurilor de utilizatori principali;
- furnizarea unui model de date care să accepte efectuarea oricărei tranzacții necesare asupra datelor;
- specificarea unui proiect minimal, structurat în mod adecvat pentru realizarea cerințelor stabilite privind performanțele sistemului, cum ar fi timpul de răspuns.

În proiectarea unui sistem de baze de date se pot utiliza mai multe **strategii de abordare** dintre care cele mai cunoscute sunt:

- **Proiectare de jos în sus** (bottom-up sau ascendentă), care începe prin definirea atributelor, a asociațiilor dintre atribute, şi în final gruparea acestora în entități. Un astfel de tip de abordare *este reprezentat de procesul de normalizare*. Normalizarea implică identificarea atributelor necesare şi plasarea lor în tabele normalizate, bazate pe dependențele funcționale dintre atribute. Acest tip de proiectare a unui sistem de baze de date este indicat în proiectarea unor baze de date simple, cu un număr relativ mic de atribute.
- **Proiectare de sus în jos** (top-down sau descendentă), este indicată în proiectarea unor aplicații de tip baze de date complexe. Aceasta începe cu realizarea unor modele de date, care conțin câteva entități şi relații de nivel înalt, după care se aplică rafinări succesive de sus în jos, pentru a identifica entitățile, relațiile şi atributele asociate de nivel jos. Acest tip de abordarea *este specific modelului Entitate - Relație*, care începe cu identificarea entităților şi relațiilor dintre ele care prezintă interes pentru organizație.

De asemenea, toate metodologiile utilizate în etapa de proiectare (numită şi etapa de modelare) a bazelor de date cuprind trei etape principale, şi anume:

- Proiectarea conceptuală (elaborarea modelului conceptual) – reprezintă procesul de construire a unui model al informațiilor utilizate în cadrul unei organizații, independent de toate considerațiile fizice. Această fază este complet independentă de detaliile de implementare (elementele software ale SGBD-ului, programe de aplicații, limbaje de programare, platforma hardware, etc.).
- Proiectarea logică (elaborarea modelului logic) – reprezintă procesul de construire a unui model al informațiilor utilizate în cadrul unei organizații, bazat pe un anumit model de date, dar independent de un anumit SGBD şi de alte considerații fizice.
- Proiectarea fizică (elaborarea modelului fizic) – reprezintă procesul de realizare a unei descrieri a implementării bazei de date într-o capacitate de stocare secundară; descrie structurile de stocare şi metodele de acces utilizate pentru realizarea unui acces eficient la date. Modelul rezultat în urma acestei etape este cel care stă la baza materializării bazei de date propriu-zise.

Proiectarea bazei de date se poate realiza în două moduri:

- pornind de la o bază de date existentă, utilizând procedeul reverse engineering (proiectare inversă);
- pornind de la zero, elaborând pe rând cele trei modele enumerate mai sus utilizând procedeul forward engineering (proiectare directă).

Când un administrator de baze de date este întrebat ce instrument **CASE** (**C**omputer **A**ided **S**oftware **E**ngineering) foloseşte în proiectarea, implementarea şi analiza bazelor de date, unii vor răspunde Oracle Designer, alții CA ERWin/ERX sau Embarcadero ERStudio, iar alții vor opta pentru Rational Rose şi lista poate continua. Desigur, opțiunile de alegere a unui anumit instrument sunt dictate, în general, de preferințe de natură subiectivă – cunoaşterea respectivului instrument fiind decisivă în alegerea produsului.

6.1. Proiectarea bazelor de date utilizând regulile de business

Una din metodele propuse de Microsoft pentru proiectarea conceptuală a unei baze de date este **ORM** (Object Role Modeling). În Microsoft® Visio® for Enterprise Architects se poate utiliza o diagramă de tip ORM pentru a elabora modelul conceptual al unei baze de date. Modelul rezultat este independent de platforma pe care se va implementa baza de date rezultată.

Dar ce reprezintă ORM ?

ORM reprezintă o descriere generală a bazei de date, utilizând simboluri intuitive pentru obiecte şi atribute şi care verbalizează (exprimă) relaţiile dintre acestea, utilizând o gamă largă de constrângeri care surprinde şi forţează regulile de afaceri.

De asemenea, ORM este un mod de abordare conceptual de un nivel înalt, utilizat pentru modelarea datelor care descrie domeniul aplicaţiei (lumea este văzută în termeni ai obiectelor, şi a rolurilor pe care aceste obiecte le îndeplinesc) utilizând în acest scop simboluri intuitive şi un limbaj natural pe care atât utilizatorii cât şi proiectanţii bazei de date le poate înţelege. Fiecare tip de element al faptului care are loc între tipuri de obiecte este verbalizat şi afişat într-o diagramă de schemă conceptuală. În acelaşi timp, ORM permite folosirea unui set extins de constrângeri pentru a surprinde regulile de afaceri (business rules), precum şi încorporarea unor exemple pentru verificarea proiectului.

Procedura de proiectare a schemei conceptuale ORM se caracterizează prin analiza şi proiectarea datelor. Schema conceptuală specifică structura informaţională a aplicaţiei prin intermediul tipurilor de fapte. Un model sursă ORM poate fi folosit în loc de, sau înaintea unui alt model de proiectarea a bazei de date.

ORM permite:

- Proiectarea conceptuală a bazei de date folosind fapte şi exemple descrise într-un limbaj natural.
- Construirea automată a modelelor logice şi fizice ale bazei de date pe baza faptelor exprimate într-un limbaj natural.
- Modelul bazei de date este creat într-un limbaj care poate fi înţeles de utilizatorii non-tehnici.

Dintre caracteristicile mai importante ORM enumerăm:

- este **uşor de înţeles** – exprimă fapte şi reguli în limba engleză şi / sau alte grafice intuitive;
- este **fiabil** – validează regulile folosind limba engleză şi mostre de date;
- este **expresiv** – surprinde în mod grafic multe reguli de business şi mai complexe;
- este **stabil** – minimizează impactul modificărilor asupra modelului şi interogărilor.

BIBLIOGRAFIE

[Andronie, 2007] Andronie M., *Analiza şi proiectarea sistemelor informatice de gestiune*, Editura Fundaiei Române de Mâine, Bucureti, 2007

[Bocu, 2002] Bocu D., *Iniiere în modelarea obiect orientată a sistemelor utilizând UML*, Grupul microINFORMATICA, Cluj-Napoca, 2002

[Bodur&alt, 1982] Bodur-Lăţescu Gh., Ciobanu Gh., Băncilă I., *Analiza drumului critic*, Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1982

[Bosksenbaum, 2002] Boksenbaum L., *Informatică de gestiune*, Editura Economică, Bucureti, 2002

[Davidescu, 1998] Davidescu N.D., *Sisteme informatice financiar-bancare. Concepte fundamentale. Modelare prin metoda MERISE*, vol. I, Editura ALL BECK, Bucureşti, 1998

[Davidescu, 1999] Davidescu N.D., *Sisteme Informatice financiar-bancare - Metoda Merise*, Vol II Aplicaţii Practice, Editura ALL Beck, Bucureşti,1999

[Georgescu, 2002] Georgescu C., *Abordarea relaţională şi obiectuală în analiza sistemelor informatice*, Editura Didactică şi Pedagocică, Bucureşti 2002

[Lungu1&alt, 1995] Lungu I., Bodea C., Bădescu G., Ioniţă C., *Baze de date. Organizare, proiectare şi implementare*, Editura All Educational, Bucureşti, 1995

[Lungu&alt, 1995] Lungu I., Sabău Gh., Bodea C., Surcel Tr., *Sisteme informatice pentru conducere*, Editura Siaj, Bucureşti, 1995

[Lungu&alt, 2003] Lungu I., Sabau Gh., Velicanu M., Muntean M., Ionescu S., Posdarie E., Sandu D., *Sisteme informatice. Analiză, proiectare şi implementare*, Editura Economică, Bucureşti, 2003

[Meyer, 1997] Meyer B., *Object – Oriented Software Construction*, Second Edition, Prentice Hall, 1997

[Nistor&alt, 2003] Nistor R., Nistor C., Căpăţână Al., *Metodologii manageriale informatice*, Editura Academica, Galaţi, 2003

[Oprea, 1999] Oprea D., *Analiza şi proiectarea sistemelor informaţionale economice*, Editura Policrom, Iaşi, 1999

[Roca&alt, 1993] Roşca I., Macovei E., Davidescu M., Răileanu V., *Proiectarea sistemelor informatice financiar-contabile*, Editura Didactică şi Pedagogică, Bucureşti, 1993

[Vătui, 2000] Vătui M., *Metode de analiză orientată obiect a sistemelor informaţionale* – Teză de doctorat, A.S.E. Bucureşti, 2000

[Vîrlan, 2001] Vîrlan G., *Utilizarea limbajului de modelare UML în analiza i proiectarea sistemelor*, Editura Mongabit, Galai, 2001

[Vîrlan, 2004] Vîrlan G., *Tehnologii client/server în dezvoltarea sistemelor informatice în economie*, Editura Didactică şi Pedagogică R.A., Bucureşti, 2004