Departamentul Automatică şi Informatică Industrială
Facultatea Automatică şi Calculatoare
Universitatea POLITEHNICA din Bucureşti

Analiza si proiectarea sistemelor informatice

I. ANALIZA DE SISTEM

II. MODELARE ORIENTATA PE OBIECTE

Bibliografie admitere master, an universitar 2020-2021, pentru programele de master coordonate de Departamentul AII:

Automatica si Informatica Industriala (AII)
Managementul si Protectia Informatiei (MPI)
Robotics and Automation (RA)
Prelucrari Complexe de Semnal in Aplicatii Multimedia (PCSAM)
Sisteme Informatice in Medicina (SIM)
Service Engineering and Management (SEM)

I. ANALIZA DE SISTEM

1. CONSIDERAŢII PRIVIND SISTEMELE INFORMATICE. TERMINOLOGIE

O rezolvare rapidă, precisă şi economică a tuturor problemelor care apar la implementarea unui sistem informatic în sfera socio-economică, implica un **cadru metodologic şi normativ adecvat de realizare a acestor sisteme**.

Perfecţionarea continua a acestui cadru a devenit o cerinţă obiectivă, în condiţiile actuale şi de perspectivă a informaticii în ţara noastra. În acest sens, putem menţiona câţiva din factorii care pot influenţa în mod benefic dezvoltarea informaticii:

- **extinderea continuă a domeniilor de utilizare a tehnicii de calcul ca urmare a diversificării cerintelor sociale în raport cu informatica (e-learning; ebusiness; e-governing, etc.)**;
- **evoluţia tehnologiilor informatice exprimată prin diversificarea echipamentelor de tehnica de calcul, apariţia unor noi concepte privind arhitectura şi structura sistemelor informatice**.

Aceste observaţii se înscriu, în mod evident, şi în tendinţele generale semnalate pe plan mondial pentru actualul deceniu, tendinţe ce pot fi exprimate succint în urmatoarele: informatizarea într-un ritm înalt a tuturor domeniilor activităţii umane, ceea ce implică o diversificare a domeniilor de utilizare a tehnicii de calcul, utilizarea tehnicii de calcul de către utilizatori de diferite profesii şi domenii de activitate, utilizarea de sisteme de conducere distribuita a proceselor şi orientarea rapidă spre utilizarea reţelelor de calculatoare, intervenţii minime în desfăşurarea proceselor specifice tehnicii de calcul şi integrarea largă a diferitelor aplicaţii si sisteme informatice sub aspect tehnico-functional .

Realizarea de sisteme informatice complexe este materializată într-o multitudine de aplicaţii ale tehnicii de calcul: birotică, proiectarea asistată de calculator, robotică, supravegherea şi conducerea proceselor tehnologice, prelucrarea informaţiilor economice, informatizarea învăţământului, etc.

Menţionăm ca o trăsătură definitorie pentru toate aplicaţiile şi sistemele informatice tendinţa de integrare a acestora, tendinţă ce trebuie vazută sub doua aspecte:

- conducere generală : integrarea se realizează între diferite niveluri de conducere şi orizonturi de planificare-urmărire, elementele principale ale integrării fiind modelele de planificare şi bazele de date.
- conducerea proceselor de producţie : integrarea se realizeaza conform stadiilor ciclului de viaţa a unui produs (cercetare, proiectare, pregătirea fabricaţiei, procesele tehnologice şi de control a calităţii).

În ambele cazuri, integrarea conduce la arhitecturi de sisteme informatice de prelucrare, manipulare şi stocare distribuită a datelor, cu regim de prelucrare adesea în timp real. Aceste arhitecturi vor trebui sa fie concepute într-o astfel de maniera încat uzura morală a componentelor acestora să nu conducă la perimarea structurii informatice.

În acest context, analiza şi sinteza sistemelor informatice îşi propune ca plecând de la studierea procesului tehnologic şi a cerinţelor acestuia pentru o funcţionare conform unor indici de calitate (analiză), să se stabilească arhitectura sistemului

informatic atât din punct de vedere hardware şi software, dar inclusiv să dea soluţii pentru problema transmiterii informaţiei între diferitele subsisteme (sinteză).

Pentru a întelege mai bine consideraţiile pe care le vom face în cele ce urmează, este necesar să prezentăm câteva noţiuni de bază utilizate în diverse domenii ale informaticii.

1.1. Terminologie

Sistem (tehnic): un ansamblu de elemente componente fizico-tehnice, care actioneaza unele asupra altora intr-un mod bine determinat.

Proces industrial: ansamblu de fenomene de natura complexa, conceput, de regula, de om, cu o destinatie functionala precisa ce expliciteaza transformarile masice si/sau de energie.

Model: reprezentare matematica a dependentei dintre mai multe marimi. Daca dependenta corespunde unui proces fizic realizabil se spune ca avem un **model sistemic**. La un model sistemic exista o relatie de cauzalitate intre marimi. Aceasta relatie imparte marimile ce caracterizeaza un model in doua clase: marimi de intrare (cauza) si marimi de iesire (efect).

Se asociaza unui proces industrial o reprezentare de tipul celei din fig. 1.1, unde prin Ei si Ee am notat fluxurile de energie, materii prime sau materiale transmise procesului, respectiv fluxurile de energie, materiale sau produse finite extrase din proces. Marimile fizice ce caracterizeaza fenomenele din sistemul tehnic le vom numi **semnale**. Variatia lor in timp contine pentru beneficiarul instalatiei o anumita semnificatie, deci ele sunt purtatoare de informatie. In aplicatiile industriale semnalele servesc pentru influentarea lui Ei si Ee. In fig. 1.1 s-au mai facut urmatoarele notatii: u - marimea de comanda, y - marimea masurata (de iesire, parametru reglat), z - marimea de calitate, x - marimea de stare, v - marimea perturbatoare.

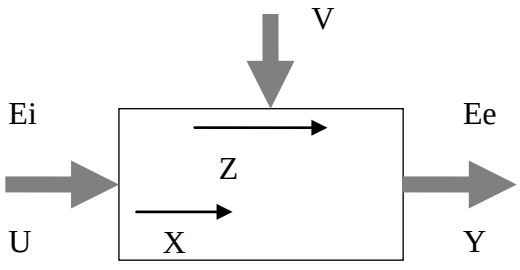

Fig. 1.1. Reprezentarea sistemica a proceselor

Privit prin prisma teoriei sistemelor, componentele **analizei de sistem** dupa Daenzer, sunt prezentate in fig. 1.2.

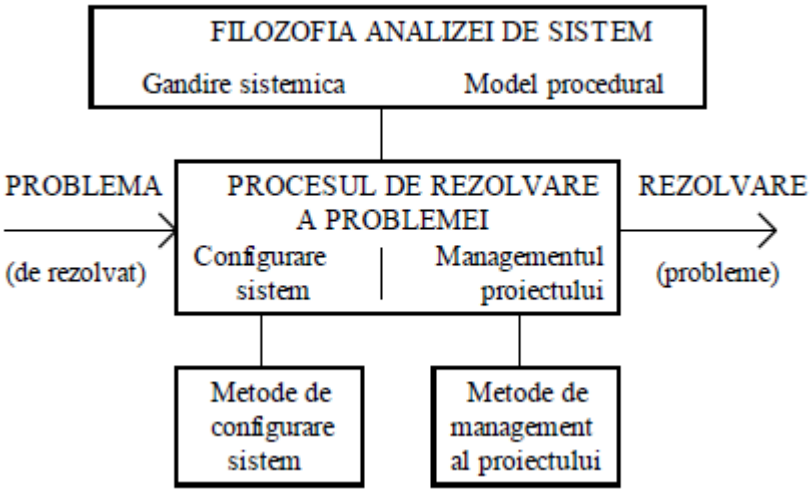

Fig. 1.2. Componentele analizei de sistem [Daenzer]

Se poate constata ca **analiza de sistem** **este o constructie de metode, tehnici si instrumente destinate tratarii problemelor de complexitate ridicata**. Analiza de sistem este implicata in ciclul de viata al sistemelor, in particular al celor informatice, deci in proiectarea, planificarea si realizarea lor, inclusiv organizarea si exploatarea acestora. In cadrul acestor activitati sunt cuprinse si relatiile sau serviciile sistemului cu mediul exterior.

Produs informatic: denumire generica prin care se refera sistemul informatic, o aplicatie informatica sau produsul program.

Aplicatie informatica: utilizarea calculatorului in rezolvarea unui grup omogen de probleme ale unui utilizator individual printre care distingem:
- aplicatii de gestiune;
- aplicatii stiintifice;
- aplicatii de birotica.

Produs program: sistem complet si documentat de programe, livrabil unui grup de utilizatori care reprezinta:
- implementarea uneia sau mai multor aplicatii informatice la utilizatorii din grup;
- suportul de realizare si exploatare a produselor program aplicative de uz general sau dedicat.

Produs program dedicat: sistem complet si documentat de programe care rezolva un grup omogen de probleme ale unui utilizator individual (implementeaza o aplicatie informatica).

Sistem informatic: ansamblu constituit din urmatoarele tipuri de elemente:
- *echipamente*, care pot fi: unul sau mai multe calculatoare, memorii, periferice;
- *software* compus din: soft de baza, soft de gestionare a bazelor de date, soft de aplicatie;
- *personal de exploatare*, utilizatori de specialitate pentru intretinere;

- *organizarea activitatilor* de pregatire a mediului de achizitiea datelor, de supraveghere a exploatarii si intretinerii sistemului.

Ciclul de realizare al unui produs informatic: partea din ciclul de viata al unui produs informatic in cadrul careia se realizeaza respectivul produs.

Ciclul de viata al unui produs informatic: perioada de timp intre momentul aparitiei cererii prin care se solicita realizarea unui produs informatic si momentul scoaterii lui din exploatare.

Calculator de proces: echipament numeric de calcul care poate realiza conducerea unui proces in timp real si care pe langa interfetele clasice dispune de o interfata industriala prin care se realizeaza o legatura bilaterala cu procesul condus, precum si de o consola a operatorului de proces ce permite dialogul operatorcalculator.

Timp real: mod de organizare a prelucrarilor prin care calculatorul trebuie sa furnizeze, pe baza datelor primite de la un proces, informatii necesare de comanda si control intr-un timp compatibil cu regimul tranzitoriu din proces.

Hardware (echipament): echipament fizic folosit pentru prelucrarea datelor, spre deosebire de programe, proceduri, reguli si documentatia asociata acestora.

Software: produs intelectual constand din programe, reguli si documentatia asociata pentru functionarea unui sistem de prelucrare a datelor (notiunea de software nu include si suportul fizic utilizat pentru a-l manipula).

Software critic: componenta software a unui produs informatic a carui functionare eronata produce efecte negative foarte mari in mediul utilizator.

Software de operare (sistem de operare): software care controleaza executia programelor si poate sa asigure functiuni ca: alocarea resurselor, planificarea resurselor, controlul intrarilor, al iesirilor si gestiunea datelor.

Software suport: software care constituie suport pentru realizarea si exploatarea produsului informatic si care va fi incorporat in componentele software ale produsului final; cuprinde: software-ul de baza, software-ul de gestionare a colectiilor de date, software-ul de comunicatie.

In stabilirea cadrului metodologic si normativ de realizare a produselor informatice, mai precis a sistemelor informatice, nu este lipsita de importanta o privire generala asupra *clasificarii* acestor sisteme.
O astfel de clasificare trebuie sa corespunda unui scop, sa defineasca multimea obiectelor supuse clasificarii si sa stabileasca criteriile de clasificare in raport de care sint stabilite clasele de sisteme informatice. Multimea obiectelor supuse clasificarii este constituita din multimea sistemelor informatice, sisteme ce nu trebuie privite de sine statator, ci impreuna cu sistemul obiect ce reprezinta o anumita structura socioeconomica sau tehnica.

Scopul clasificarii este, in special, de domeniu conceptual, fiind insa un mijloc eficient de planificare si urmarire a realizarii sistemelor informatice.

In ceea ce priveste criteriile, **sistemele informatice se pot clasifica dupa**:

- nivelul sistemului obiect pe care se grefeaza sistemul informatic: sisteme destinate unitatilor socio-economice de baza, companii, teritoriale sau functionale generale;
- domeniul de utilizare;
- stadiul atins in evolutia sa de sistemul informatic.

Legat de cel de-al treilea criteriu de clasificare este de mentionat faptul ca se impune o abordare evolutiva a unui sistem informatic, tranzitia de la un stadiu inferior catre unul superior marcand atat aspecte cantitative, cat si calitative.

O abordare evolutiva in realizarea unui sistem informatic prezinta o serie de avantaje, cum ar fi: dezvoltare treptata a sistemului, familiarizarea utilizatorului, alocarea generala a resurselor umane si materiale, etc.

Trecerea de la un studiu la altul superior nu se poate face decit avand in vedere o serie de **criterii**: extinderea ariei sistemului obiect, justificare economica, grad de integrare si distribuire, modul de organizare a datelor.

2. PRINCIPII GENERALE DE REALIZARE A PRODUSELOR INFORMATICE

2.1 Etapele de realizare a sistemelor (aplicatiilor) informatice

Ciclul de viata a unui sistem informatic este constituit din urmatoarele etape:

1. **Elaborarea temei de realizare** avand drept obiective:
   - identificarea cerintelor si restrictiilor globale pentru realizarea sistemului;
   - delimitarea ariei de aplicabilitate;
   - justificarea necesitatii, oportunitatii si fezabilitatii modelului functional global al noului sistem adoptat in comparatie cu alte solutii;
   - stabilirea cadrului tehnologic de realizare si de control al calitatii.

Rezultatele acestor activitati se vor materializa sub forma **temei de realizare (temei de proiectare).**

Aceasta tema va cuprinde baze de elaborare a temei de proiectare cerintele si restrictiile globale pentru realizarea sistemului informatic, precum si o justificare a necesitatii si oportunitatii realizarii acestui sistem.

2. **Proiectarea de ansamblu**

Obiectivele principale ale acestei etape sunt urmatoarele:

- specificarea cerintelor si restrictiilor pentru proiectarea noului sistem;
- elaborarea modelului de ansamblu a noului sistem informatic;
- stabilirea grafului de ordonantare a exploatarii componentelor functionale si a cerintelor privind asigurarea informationala;
- estimarea necesarului de testare pentru realizarea si punerea in functiune a noului sistem si a eficientei economice;

- planificarea realizarii si punerii in functiune a noului sistem; - planificarea testarii.

Rezultatul acestei etape este reprezentat de **proiectul de ansamblu**. Totodata se intocmeste si o **specificatie de testare** pentru: testul de integrare, testul de sistem si testul de acceptanta (receptie/ omologare).
Aceasta etapa se constituie, de regula, ca etapa distincta numai pentru sistemele informatice de mare complexitate care necesita o realizare esalonata in timp. Pentru produsele program proiectul de ansamblu si de detaliu - etapa urmatoare - se executa in faza unica.

3. Proiectarea de detaliu are ca obiective:
- analiza si specificarea cerintelor de detaliu;
- elaborarea modelului de detaliu (integral sau pe parti componente) - proiectarea arhitecturii compnentei functionale; - stabilirea solutiilor tehnice de realizare;
- planificarea realizarii si punerii in functiune a componentei functionale; - planificartea testarii.

Aceasta etapa se finalizeaza prin:
* **proiectul de detaliu al componentei functionale**;
* **specificatia de testare**;
* **raportul de evaluare a etapei**;
* **planul de punere in functiune**.

4. Elaborare programe, principalele obiective fiind:
- proiectarea, realizarea, testarea programelor;
- elaborarea documentatiei de intretinere (programe si date); - pregatirea testarii.

Rezultatele etapei sint materializate in urmatoarele documentatii:
* **specificatia de realizare a programelor**;
* **specificatia de testare**;
* **raport de testare si listinguri martor**;
* **documentatia de intretinere**;
* **raportul de evaluare a etapei**.

5. Integrare si testare
- a componentei functionale;
- a sistemului;
- elaborarea documentatiei;
- planificarea punerii in functiune a sistemului sau a componentelor lui functionale;
- pregatirea receptiei.

Documentele recomandate a fi elaborate la sfarsitul acestei etape sunt urmatoarele:
* **documentatia de utilizare-exploatare**;
* **documentatia de intretinere**;
* **biblioteci cu componente software**;

* **specificatii de testare** ;
* **metodica si programa de receptie**.

6. **Punerea in functiune/experimentare si acceptare sistem** care consta, in principal, din urmatoarele activitati:

- actiuni pregatitoare punerii in functiune: instruire personal, masuri organizatorice si tehnice;
- punerea in functiune/experimentare la unitatea beneficiara;
- test de acceptanta/receptie, receptie sistem;
- actualizarea documentatiei/componentelor functionale.

Vor fi elaborate urmatoarele documente:

* in forma finala, **documentatia de utilizare-exploatare si documentatia de intretinere**;
* **biblioteci** sau **fisiere cu componentele software**;
* **procesul verbal de receptie**.

7. **Exploatare si intretinere** in care se urmareste:

- functionarea sistem la parametri proiectati;

- intretinerea sistemului;
- actualizare documentatie.

Modificarile care apar in timpul exploatarii vor fi mentionate in registrele de exploatare, actualizandu-se in documentatia componentelor functionale.

2.2 Etapele de realizare a produselor program

In realizarea produselor program vor trebui sa se respecte urmatoarele principii:

- fundamentarea realizarii si dezvoltarii produselor program pe considerente tehnico-economice, ceea ce presupune o analiza comparativa a produselor program, evaluarea numarului de beneficiari potentiali si o estimare a raportului cost/ performanta; trebuie sa se realizeze si o analiza a solutiilor tehnice din diferitele etape ale ciclului de viata a produsului program in consens cu evolutia mijloacelor hardware si software pe plan mondial;
- asigurarea calitatii programelor si a documentatiilor de utilizare, exploatare si intretinere. Trebuie definite caracteristicile de calitate pe care aceste produse trebuie sa le indeplineasca, cat si metodele, tehniciile, instrumentele si procedurile selectate pentru asigurarea si controlul calitatii; se vor specifica punctele obligatorii de control al calitatii.
- cresterea productivitatii muncii in elaborarea programelor prin: reutilizarea de solutii, pachete de programe, rutine si module destinate ridicarii nivelului de calificare a personalului, aplicarea principiilor de management, etc.

Etapele de realizare a produselor program sint in mare similare cu cele corespunzatoare realizarii aplicatiilor (sistemelor) informatice.

1. **Elaborarea temei de realizare** avand drept obiective:
- analiza si identificarea cerintelor;
- justificarea necesitatii si oportunitatii realizarii produselor program;
- specificarea cerintelor globale cu privire la produsul program;
-stabilirea cadrului tehnologic de realizare si de control al calitatii;

Etapa se finalizeaza prin **tema de realizare**.

2. **Proiectarea preliminara** cuprinzand:
- analiza si specificarea cerintelor software;
- proiectarea functionala a produsului program;
- planificarea testarii.

Documentatia elaborata dupa parcurgerea acestei etape va fi constituita din **specificatia de definire a produsului program (proiectul logic)**.

3. **Proiectarea detaliata** cu urmatoarele obiective:
- proiectare tehnica: structura fizica a produsului program si definirea programelor, proiectare tehnica a interfetelor si a bazelor de date, procedurile de executie, fluxul de control;
- specificare cerinte si restrictii tehnice de realizare;
- testare.

Etapa se poate finaliza prin:
* **specificatia de realizare a produsului program (proiect tehnic)**;
* **specificatia de testare**.

4. **Elaborarea programelor** cuprinde:
- proiectare, elaborare si testare programe;
- elaborare documentatia de intretinere (programe, date);
- - testare.

Vor fi elaborate urmatoarele documentatii:
* **specificatii de realizare programe**;
* **specificatii de testare**;
* **raport de testare si listinguri martor**;
* **documentatia de intretinere**.

5. **Integrare si testare**: integrarea si testarea progresiva a componentelor software ale produsului program.

Etapa se poate finaliza cu urmatoarele documente:
* **documentatia de utilizare-exploatare** (forma preliminara);
* **biblioteci cu componentele software** (in format executabil);
* **specificatii de testare**;
* **programa si metodica de omologare**.

6. **Experimentarea** cuprinde urmatoarele activitati:
   - activitati pregatitoare in vederea experimentarii: instruirea personalului, masuri organizatorice si tehnice;
   - instalarea produsului la utilizator;
   - aducerea documentatiei la conditiile corecte de utilizare.

Finalizarea experimentarii se materializeaza prin:
* **documentatia de intretinere si cea de utilizare-exploatare** (in forma definitiva);
* **biblioteci cu componente software**.

7. **Omologarea**

Observatie:
* Realizarea succesiunii de etape de mai sus se poate face, ca si in paragraful anterior, prin strategia clasica sau a cea a prototipizarii (aceste strategii vor fi prezentate in capitolul urmator).
* Documentatia ce se elaboreaza dupa fiecare etapa este orientativa, cu exceptia documentatiei aferente temei de realizare si a celei destinate utilizarii, exploatarii si intretinerii produselor program, care are un caracter obligatoriu.

2.3 Aspecte practice privind realizarea produselor informatice

In realizarea produselor informatice, in conditiile unei eficiente ridicate a activitatilor legate de acest proces, este necesar sa se respecte urmatoarele principii:

1. **Fundamentarea pe criterii de eficienta economica a proiectarii si realizarii sistemelor informatice sau a produselor program**. Trebuie, in permanenta, sa se aiba in vedere cat se cheltuieste pentru proiectarea, construirea si implementarea unui produs informatic si cat se cheltuieste cu intretinerea acestuia in raport cu beneficiile obtinute pentru ca investitiile antrenate in procesul de realizare sa se amortizeze intr-un timp cat mai scurt. In acest fel, exista posibilitatea introducerii la un moment dat, a unui nou sistem, mai performant decit cel anterior si care sa fie "la moda" (sa nu fie uzat moral).

2. **Participarea beneficiarului la realizarea sistemului informatic sau produsului program**. Aceasta participare trebuie sa se faca, in special, in faza de punere in functiune. Nu este recomandabil ca beneficiarul sa participe la etapele de proiect de detaliu, elaborare, integrare si testare. Participarea beneficiarului este benefica la momentele mai sus amintite deoarece acesta se obisnuieste cu noul sistem, se adapteaza cu el si din punct de vedere psihologic efectul de respingere este mult redus.

3. **Adaptarea sistemului si strategia prin care se face acest lucru se face dupa o estimare cat se poate de riguroasa a resurselor tehnice (echipamenete si produse-program generalizabile), umane** (personalul de specialitate de la beneficiar) **si financiare, in conditiile existentei unor restrictii impuse**. Aceste restrictii pot fi:

a) de tip informational: o serie de acte normative, legi sau reglementari, care se repercuteaza asupra datelor prelucrate, a algoritmilor si chiar a bazelor de date aferente sistemului informatic propus;
b) de ordin temporal: durata in care o serie de echipamente pot fi procurate sau durata de realizare a unor programe.
c) cu caracter evolutiv: situatia sistemului informatic aflat intr-o tranzitie de la un stadiu informatic la un stadiu superior celui precedent din punct de vedere al performantelor si al obiectivelor.

Restrictiile mentionate mai sus impun o tratare adecvata a etapelor de realizare a unui produs informatic.

4. **Adaptarea unor metode, tehnici si instrumente, care sa asigure o inalta productivitate a etapelor de proiectare si implementare din ciclul de viata al unui produs informatic**. In conditii in care se doreste realizarea unui sistem informatic care sa asigure anumite cerinte, este necesar ca atat proiectarea, cat si implementarea lui sa se faca intr-un timp cat mai scurt, deci un ciclu de realizare mic, cu o economie insemnata de resurse umane si financiare. Realizarea acestui deziderent impune, pe linga o serie de masuri organizatorice, folosirea cu precadere a solutiilor tipizate. Practic, se pot aplica urmatoarele trei solutii de tipizare in realizarea unor sisteme informatice:

* proiecte de sisteme informatice si de produse program aferente pentru grupe tipologice de unitati economico-sociale (de exemplu: gestionarea unei magazii, modul de afisare a unor marimi, etc.);
* produse program generalizabile (proiecte de programe aplicabile unui evantai larg de aplicatii);
* realizarea de sisteme informatice din elemente tipizate.

Cerintele ce trebuie acoperite de noul sistem informatic pot fi catalogate in:

- cerinte de performanta si calitate: fiabilitate, realizarea unor viteze mari de prelucrare a informatiilor sau in comunicatia proces - calculator;
- cerinte informationale: rolul sistemelor informatice, (achizitie, prelucrare, editare date, etc.), facilitati oferite de sistemul nou propus (interpretarea autorizata si in timp real a rezultatelor, editare on-line sau off-line a unor protocoale, interfata conventionala cu utilizatorul, protectia la erori, etc.), cerinte de interfata ale sistemului de calcul in raport cu mediul (procesul) sau cu utilizatorul;
- cerinte si restrictii cu caracter tehnic si organizatoric in instalare si exploatare: respectarea unor standarde pentru echipamente, asigurarea unei continuitati in alimentarea cu energie electrica si a unei frecvente constante, etc.

In activitatile legate de realizarea unui sistem informatic se pot adopta diverse **strategii** functie de complexitatea proiectului, de resursele antrenate cu acest prilej si de obiectivele urmarite (de exemplu: scurtarea ciclului de realizare optimizarea utilizarii resurselor, grad inalt de integrare, furnizarea rapida a unor prime rezultate la utilizator), obiective care in majoritatea solutiilor sunt contradictorii.

Practica realizarii acestor sisteme arata ca dezvoltarea in etape sau stadii (versiuni) succesive a unui sistem, cu alte cuvinte, **o dezvoltare evolutiva a sistemului informatic**, este posibila si chiar de dorit, ea permitind o "maturizare" gradata pe masura ce sint identificate cerintele si restrictiile reale si se dezvolta si baza materiala.

In situatia in care se pune problema unei tranzitii de la un stadiu inferior la un stadiu superior a unui sistem informatic, este necesar sa se adopte o modalitate aparte de parcurgere a etapelor de realizare. Se fac, in acest sens, urmatoarele precizari si recomandari:

a) este necesar ca inca din faza de proiect de ansamblu sa se defineasca stadiile de evolutie a sistemului, delimitindu-se aria de cuprindere, resursele si duratele necesare realizarii, precum si modalitatea de conversie. Trecerea de la un stadiu la alt stadiu se face in aceste conditii in reluarea ciclului de realizare de la etape de proiectare de detaliu, urmata de elaborare programe, etc. Reluarea ciclului presupune si o reevaluare a proiectului de ansamblu, ceea ce implica corective asupra stadiilor urmatoare de evolutie si a resurselor implicate;

b) daca tranzitia de la un stadiu la altul superior se face cu modificarea conceptiei intregului sistem - se modifica legaturile intre componentele functionale, echipamente, modul de organizare si administrare a datelor - ciclul trebuie reluat din etapa de tema de realizare, avandu-se in vedere procedurile de conversie necesare in special in ceea ce priveste colectiile de date;

c) in toate situatiile in care apare problema acestei tranzitii este necesar ca esalonarea in timp a stadiilor si etapelor sa se faca tinand cont de termenele finale stabilite initial.

3. TEHNOLOGIA DE REALIZARE A UNUI PRODUS INFORMATIC

3.1. Consideratii generale

Finalizarea unui produs informatic implica parcurgerea a doua activitati majore: o activitate de **conceptie** sau concepere (sinonima, in parte, cu notiunea de **proiectare**) si o activitate de **realizare**.

Inainte de a examina aspectele legate de continutul procesului de conceptie si realizare a unui produs tehnic, in particular a unui produs informatic, vom face cateva precizari absolut necesare legate de partea procedurala (formala) a produsului de concepere.

Inca din antichitate s-au centralizat doua modalitati logice de gandire, doua dialectici: o dialectica ascendenta (modalitate **bottom-up**), prin care se formeaza conceptele, si o dialectica descendenta (modalitate **top-down**) sau operatia de diviziune logica a conceptelor. Unirea celor doua modalitati intr-un singur complex, complexul **top-down-bottom-up,** poate fi reprezentat schematic in fig. 3.1.

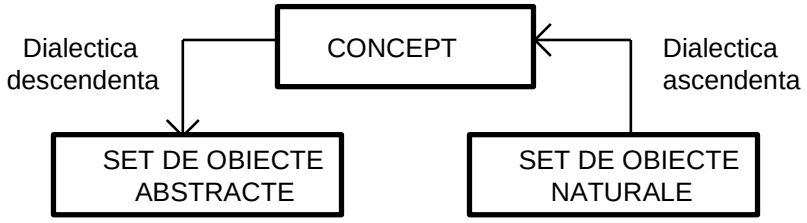

Fig. 3.1

Se poate constata ca intr-o prima etapa are loc un proces de abstractizare; punctul de plecare este reprezentat de un set de obiecte naturale, punctul terminus fiind un obiect abstract, conceptul. Urmeaza o a doua etapa in care pornindu-se de la concept, printr-un proces de divizare se ajunge la identificarea unui set de obiecte abstracte care reprezinta "oglinda" setului de obiecte naturale. Aceasta a doua etapa nu are o finalitate practica imediata, fiind doar un proces de adincire a cunoasterii.

Aplicarea complexului top-down-bottom-up la obtinerea produselor informatice, a produselor tehnice in general, este schitata in fig. 3.2, cu precizarea ca obiectul abstract, conceptul, se considera ca existent apriori, nefacandu-se precizari asupra modului in care s-a ajuns la obtinerea acestui obiect.

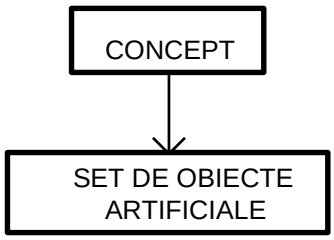

Fig. 3.2

Modul de identificare a setului de obiecte artificiale este un proces top-down, care in situatia de fata nu mai este o etapa fara finalitate, un proces contemplativ, din contra, este un proces activ prin care sunt create obiecte artificiale analoge cu cele naturale.

In capitolul anterior mentionam o etapizare care ne conducea de fapt de la concept la produsul informatic, etapizare pe care o admiteam ca o ipoteza de lucru: * tema de proiectare (realizare, determinarea utilizarii) careia ii corespunde in practica curenta: studiul preliminar, studiul de fezabilitate, studiul tehnico-economic, etc.

* proiectarea logica de ansamblu si detaliu - proiectare logica - prin care se determina cerintele si restrictiile logic-functionale globale, respectiv de detaliu, actiune finalizata in modelul logic.
* proiectare tehnica de ansamblu si detaliu - proiectare tehnica sau de executie - determina cerintele si restrictiile tehnice globale, respectiv de realizare de detaliu, finalizare modelul tehnic.
* realizare produs.

Nu s-au retinut in contextul acestui capitol etapele de testare omologare, exploatare, intretinere, etc.

Problema care se ridica intr-o analiza de sistem, in general in domeniul creativitatii tehnice, consta in faptul ca desi literatura de specialitate consemneaza o serie de metode si tehnici pentru obtinerea mai intii sub forma abstracta a modelului logic, apoi a modelului tehnic si in cele din urma pentru realizarea fizica a acestui produs, totusi aceste metode si tehnici nu pot fi general valabile. Ele sunt particularizate in functie de experienta dobandita de catre analistul de sistem in functie de calitatile acestuia, cuma ar fi: puterea de analiza si sinteza, spiritul de abstractizare, creativitatea, etc.

Avind in vedere afirmatiile de mai sus metodele si tehnicile, precum si etapizarile in procesul de elaborare o unui produs informatic, le consideram orientative, proiectantul urmand a le adopta in functia de complexitatea problemei abordate, de cerintele beneficiarului, resursele disponibile, etc.

3.2 Modelarea sistemelor informatice sau a produselor program

O notiune esentiala in definirea unui produs informatic (sistem sau aplicatia informatica, produs program) este notiunea de **sistemul obiect**. Pe acest sistem obiect se grefeaza produsul informatic.

Sistemul obiect reprezinta o parte a realitatii care genereaza date si care poseda calitati ce permit prelucrarea si atribuirea de semnificatie acestor date, transformindu-le in acest fel in informatii.

Plecand de la aceasta notiune vom prezenta cateva aspecte legate de continutul procesului de concepere, proces ce pune in evidenta citeva tipuri elementare de activitati de modelare (fig 3.3.). Le vom prezenta in ordinea de abordare a lor in conformitate cu logica procesului de cunoastere.

Modelul infologic este rezultatul obtinut prin modelarea structurii si dinamicii sistemului obiect in scopul punerii in evidenta a corelatiei dintre obiectivele, cerintele informationale si unor invarianti ai sistemului obiect.

Invariantii sistemului obiect au un rol deosebit in conceperea unui produs informatic si provin din:

- partea informationala a procesului de productie (graful procesului, graful care reprezinta structura produselor rezultate din proces).
- procesele decizionale (graful de structura organizatorica a unitatii de productie).
- activitatea de gestiune (relatiile furnizor-unitate sau unitate-beneficiari, relatiile financiare plati-incasari).

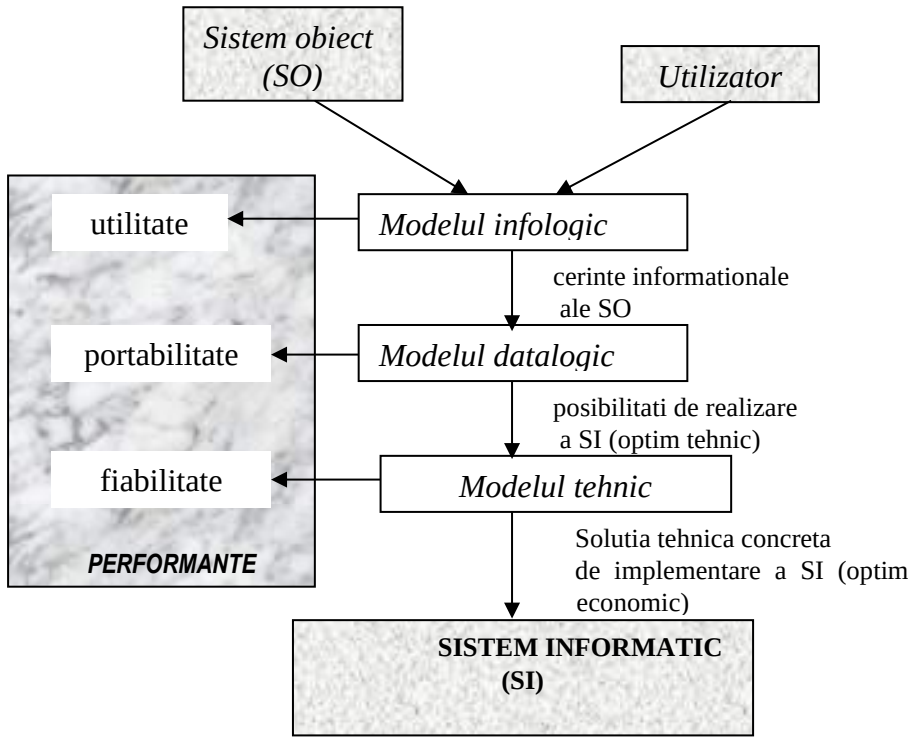

Fig.3.3. Tipurile elementare de activitati de modelare

In definitia modelului infologic termenul de dinamica are sensul de functionare a sistemului ca un sens de evolutie in timp.

Modelarea infologica are drept obiectiv principal punerea in evidenta a cerintelor informationale ale sistemului obiect, cerinte care trebuiesc indeplinite de viitorul produs informatic. In felul acesta se pune in evidenta utilitatea produsului informatic.

Modelarea datalogica reprezinta urmatorul pas firesc in logica procesului de concepere a unui produs informatic, determinata de stabilirea unor mijloace adecvate, mai performante, in raport cu cele existente, pentru satisfacerea cerintelor informationale ale sistemului obiect.

Modelul datalogic este rezultatul obtinut prin modelarea structurii si dinamicii unui sistem informatic sau produs program cu scopul atingerii, cu rezultate mai performante, a cerintelor informationale ale sistemului obiect, fara a se lua in considerare insa mijloacele si conditiile practice de construire a sistemului sau produs program.

Deci, la conceperea modelului datalogic, nu sunt luate in considerare tipurile de echipamente, prelucrare si afisarea datelor, resursele financiare, umane si de timp, etc. Acest model este insa necesar pentru a se crea posibilitatile de implementare a produsului informatic pe diferite tipuri de echipamente, deci este o problema de portabilitate a produsului realizat.

Avind in vedere evolutiile pe care le pot avea unele sisteme informatice sau produse-program trebuie sa se asigure o portabilitate atat pentru sisteme informatice unicat sau produse-program independente cat si pentru sisteme informatice tipizate sau produse-program generalizabile.

Spre deosebire de modelul infologic care este orientat spre utilizator, modelul datalogic este orientat spre o masina abstracta care sa satisfaca cerintele utilizatorului. Modelul datalogic permite obtinerea unei imagini a sistemului informatic sau a produsului program foarte apropiata de imaginea fizica.

Modelul tehnic (fizic) este rezultatul obtinut prin modelarea structurii si dinamicii unui sistem informatic sau produs-program in scopul atingerii, cu rezultate mai performante, a cerintelor informationale ale sistemului obiect, tinand seama de mijloacele si conditiile practice de construire a sistemului informatic sau produsului program.

Se poate constata ca modelarea datalogica, in principiu, poate fi omisa, dar acest lucru poate afecta portabilitatea. Acest procedeu se poate aplica, desi nu este recomandabil, pentru produse-program simple, de tipul aplicatiilor independente, destinate unor cicluri de viata scurta. In acest fel se poate reduce ciclul de realizare a unor astfel de produse-program.

Trebuie sa subliniem **diferenta** dintre modelul datologic si cel tehnic. Modelul datalogic indica mijloacele potentiale de satisfacere optima a cerintelor utilizatorului, dand contur imaginii unei masini abstracte. Modelul tehnic nominalizeaeza mijloacele concrete de satisfacere a cerintelor utilizatorului, punand in evidenta masina reala.

Un sistem informatic (SI) este caracterizat prin utilitate si prin performanta sa. **Utilitatea** se concretizeaza prin ce realizeaza sistemul, iar **performanta** prin modul in care utilitatea este indeplinita.

Pe un plan concret, utilitatea este un ansamblu de cerinte si restrictii impuse functionarii sistemului de utilizator in scopul realizarii unor rezultate bine

determinate. In acelasi sens performanta este un ansamblu de cerinte si restrictii impuse de o functionare optima a sistemului, in conditiile satisfacerii utilitatii.

Aceste doua aspecte, utilitate si performanta, pot fi corelate prin **functia de eficienta-cost** a unui SI, functie ce poate fi maximizata in anumite conditii concrete, de exemplu: resurse limitate, resurse si costuri limitate, sistem suport impus sau limitat, etc. Aceasta functie este un raport intre resursele exprimate valoric necesare pentru a realiza o utilitate si performanta SI si respectiv, costurile totale ale SI (inclusiv cele de exploatare-intretinere).

In cadrul dualismului complex-individualitate sistemul informatic trebuie sa indeplineasca anumite caracteristici, performante de baza:

- **flexibilitatea** definita prin capacitatea sistemului informatic de a se adapta la orice modificare impusa de beneficiarul sistemului in sensul exploatarii acestuia; - **fiabilitatea** este capacitatea sistemului informatic de a indeplini anumite functii sau cereri fara erori;
- **timpul de raspuns** este definit ca intervalul de timp real, din momentul unei excitatii exogene) si momentul realizarii complete a functiilor utilitare ale sistemului informatic pentru setul respectiv de date; el poate fi echivalent, in majoritatea situatiilor, cu durata regimului tranzitoriu;
- **portabilitatea** este o proprietate ce permite ca ansamblul aplicatiilor informatice sa fie implementat pe diverse tipuri de sisteme suport (sistemul hardware si o parte a sistemului software de baza (sistemul de operare)).

* O problema deosebit de importanta in realizarea sistemelor informatice este cea a integrarii acestora in sistemul obiect. Aceasta problema reflecta si se reduce, in ultima instanta, la realizarea acesteia.

In paragraful anterior se facuse o delimitare clara intre utilitate si performanta. O analiza mai fina ne arata ca aceasta delimitare trebuie sa fie nuantata.

Intr-adevar, vom intilni cel putin patru situatii in care un parametru ce defineste performanta unui sistem informatic se transforma in utilitate:

a. Sistemele informatice destinate conducerii proceselor industriale trebuie sa lucreze in timp real. Drept urmare, timpul de raspuns devine o cerinta utilitara.

b. Pentru anumite sisteme informatice cu destinatie speciala (din domeniul militar sau cosmic, sisteme energetice nationale) se impune, pentru prevenirea avariilor ce conduc implicat la catastrofe, o inalta fiabilitate. In felul acesta fiabilitatea este si o performanta si o utilitate.

c. In cazul sistemelor informatice generalizate (sisteme cu aplicabilitate pentru o clasa de sisteme) trebuie sa remarcam ca putem avea o gama larga de echipamente si, drept urmare, portabilitatea in cazul acestor sisteme informatice e o conditie de utilitate si performanta.

d. Sistemele obiect mari necesita sisteme informatice mari deoarece acestea implica o gama larga de prelucrari de date, avind in vedere evantaiul mare de cerinte si restrictii. Realizarea acestui deziderat implica antrenarea unor importante resurse, deci investitii mari. Este absolut necesar din punct de vedere economic ca aceste investitii sa fie recuperate (amortizate in timp). Evident, acest lucru nu se poate face decit pe perioada de functionare, deci pe durata de viata a sistemului respectiv. O analiza de sistem oricat de perfecta ar fi, nu poate prevedea absolut intreaga problematica ce trebuie sa-i faca fata sistemul informatic propus.

Este necesar ca sistemul informatic sa se adapteze la cerintele ce pot apare pe parcursul duratei lui de functionare. Acest lucru face ca durata de viata a produsului informatic, a sistemului informatic, sa se mareasca si sa creasca astfel posibilitatile de recuperare a cheltuielilor ce s-au facut cu instalarea si punerea in functiune a lui. Acest lucru presupune ca sistemul informatic dispune de o flexibilitate corespunzatoare. In astfel de situatii flexibilitatea devine o cerinta utilitara.

* O problema interesanta este cea a **integrarii** interne a sistemului informatic. Integrarea interna presupune determinarea unei structuri corespunzatoare pentru cele sase componente ce constituie subsistemele unui sistem total. Dintre acestea, SP, SA si SC joaca un rol principal deoarece conceperea, proiectarea si realizarea (sau alegerea) celorlalte trei: ST, SM si SS. Este de retinut faptul ca daca ne referim la un produs program, in timp ce structura datelor pentru SA poate fi privita relativ independent, structurile SP (software de aplicatii) si SC (sistemul director) sunt intrepatrunse si formeaza un tot unitar.

3.4. Strategii de concepere si realizare a unui sistem informatic sau produsprogram

Inainte de a se constitui o tehnologie de concepere sau realizare a unui produs informatic concret este necesar sa se desfasoare o serie de activitati preliminarii:

- este necesar sa se delimiteze sistemul obiect, ceva ce creaza un cadru adecvat pentru stabilirea problemelor de solutionat;
- se va face o identificare a caracteristicilor generale ale sistemului obiect pe care se va "grefa" produsul informatic. In fond, produsul informatic este o actiune de informatizare a unui domeniu de activitate sau a unor probleme sistem obiect. Pentru ca aceasta "grefa" sa nu fie respinsa este necesar ca produsul informatic sa aiba caracteristici adecvate caracteristicilor sistemului obiect.

In tabelul 3.2 sunt date o serie de elemente relationale intre caracteristicile sistemului obiect si caracteristicile produsului informatic.
Trebuie sa se evalueze personalul disponibil pentru realizarea produsului informatic atat la cel ce elaboreaza acest produs, cat si la utilizator.
Realizarea sistemelor informatice presupune existenta si utilizarea unei tehnologii.

Tehnologia de realizare a unui sistem informatic este constituita dintr-un ansamblu de procese (activitati), metode, tehnici si instrumente, ansamblu utilizat pentru obtinerea unui astfel de sistem.

Principalele componente ale unei tehnologii de realizare sunt:

- **procesul tehnologic** cadru de realizare si intretinere este un ansamblu ordonat de activitati/ subactivitati/ operatii, desfasurate in vederea obtinerii unui sistem informatic.
- **metoda de realizare** este un ansamblu de concepte si reguli prin aplicarea carora se poate realiza si conduce un sistem informatic.
- **tehnologia de realizare** este un ansamblu de reguli, compatibil cu una sau mai multe metode care concura la desfasurarea unor activitati/ subactivitati/ operatii din cadrul unui proces de realizare.

- **instrumentul de realizare** este un produs program constituit pe baza unei/ unor metode si/ sau tehnici, prin intermediul careia unele activitati ale unui proces de realizare pot fi asistate/ efectuate de calculator.
- **principii de selectare** si asamblare de elemente furnizate de cadrul tehnologic, intr-o tehnologie concreta.

Tabelul 3.2.

RELATII INTRE CARACTERISTICI ALE SISTEMULUI OBIECT SI CARACTERISTICI ALE PRODUSULUI INFORMATIC

Caracteristici sistem obiect	Caracteristici produs informatic
Noutatea domeniului problemelor supuse informatizarii in raport cu domenii / probleme deja informatizate	Originalitatea produsului informatic, exprimata prin: - originalitatea solutiei (arhitectura, algoritm) - originalitatea facilitatilor oferite
Complexitatea domeniului / problemelor abordate	Complexitatea produsului informatic: - aria functionala: - complexitatea legaturilor logice intre functiuni; - gradul de integrare al componentelor; - dimensiunea programelor
Caracterul restrictiv al domeniului / problemelor	Caracterul critic al software-ului: - performanta si calitate software; - fiabilitate suport software

Construirea unei tehnologii concrete de realizare presupune urmatoarele elemente:
- stabilirea unei strategii de realizare a produsului informatic;
- structurarea procesului tehnologic pe etape sau faze;
- definirea obligatorie la nivelul fiecarei etape a unor elemente esentiale: obiectul etapei, conditiile de intrare in etapa, activitatile de conducere si executie, criteriile de verificare, validare si testare, graful de inlantuire a activitatilor, produsele intermediare, conditiile de iesire din etapa;
- stabilirea conditiilor de trecere de la o etapa la alta;
- evaluarea efortului de realizare pe etape;
- specificarea pe etape a metodelor, tehnicilor si instrumentului tehnologic utilizat.

Strategiile **posibile de concepere, de proiectare** a unui produs informatic au la baza tipurile de modelare prezentate in paragraful anterior.

Practica curenta impune la stabilirea unei strategii adecvate luarea in considerare si a altor factori decat cei prezentati inainte, factori ce sunt de natura practica.

II. MODELARE ORIENTATA PE OBIECTE

1. Caracteristici ale limbajului UML

Interschimbarea informatiilor de modelare intre instrumente software este posibila numai pe baza unui consens cu privire la semantica si la notatii. Standardul UML a fost creat tocmai pentru a permite interoperabilitatea dintre instrumente de modelare vizuala a obiectelor - un element esential pentru progresul industriei software.

Un prim limbaj unificat, denumit Unified Method a fost pentru prima oara lansat in octombrie 1995, dar in iulie 1997 a aparut UML versiunea 1.0, care a fost adoptata ca standard de catre OMG (Object Management Group) – organizatie formata din peste 800 de firme de software. Acest standard a evoluat de-a lungul anilor (vezi http://www.omg.org/spec/UML/).

UML este de fapt doar un element dintr-un cadru comun de dezvoltare a aplicatiilor pe care organizatia OMG si-a propus sa il creeze pentru a facilita reutilizarea, portabilitatea si interoperabilitatea in medii distribuite eterogene. El se bazeaza pe dezvoltarea unei arhitecturi dirijate de modele – MDA (Model Driven Architecture).

- Este un limbaj vizual de modelare a sistemelor (nu numai software), prin concepte orientate pe obiecte.
- Contine mecanisme de extensibilitate si specializare.
- Este independent fata de un limbaj de programare sau un proces de dezvoltare
- Contribuie la incurajarea cresterii pietei de unelte orientate pe obiecte.
- Contine concepte de dezvoltare de nivel inalt (colaborari, pattern, componente).
- Integreaza cele mai bune practici.

UML a fost creat ca un limbaj de modelare matur, care sa asigure un echilibru intre expresivitate - pentru redarea complexitatii sistemelor - si simplitate – pentru a fi utilizat cu usurinta. UML a fost adoptat ca standard in numeroase organizatii, deoarece se asteapta ca el sa fie baza multor instrumente pentru modelare vizuala, simulare si a multor medii de devoltare.

Limbajul UML este specificat in mod independent de un anumit limbaj de programare sau de un anumit proces de dezvoltare. El incurajeaza piata de instrumente software si automatizarea productiei de software, fiind cheia realizarii interoperabilitatii, care permite schimbarea de modele intre utilizatori si instrumente software, fara a pierde din informatii. Standardul este utilizat in specificarea, construirea sistemelor, vizualizarea acestora si documentare.

2. Diagrama cazurilor de utilizare

- Modelul cazurilor de utilizare
- Analiza cerintelor functionale
- Cazuri de utilizare
- Actorii si relatiile acestora

Această diagramă este utilizată în modelarea comportamentului sistemului şi a interacţiunii acestuia cu mediul exterior, şi este necesară în comunicarea cu utilizatorii finali, întrucât se folosesc foarte puţine simboluri, iar terminologia este la îndemâna tuturor.

Ea este formată dintr-un set de cazuri de utilizare, actori şi relaţiile dintre aceste elemente.

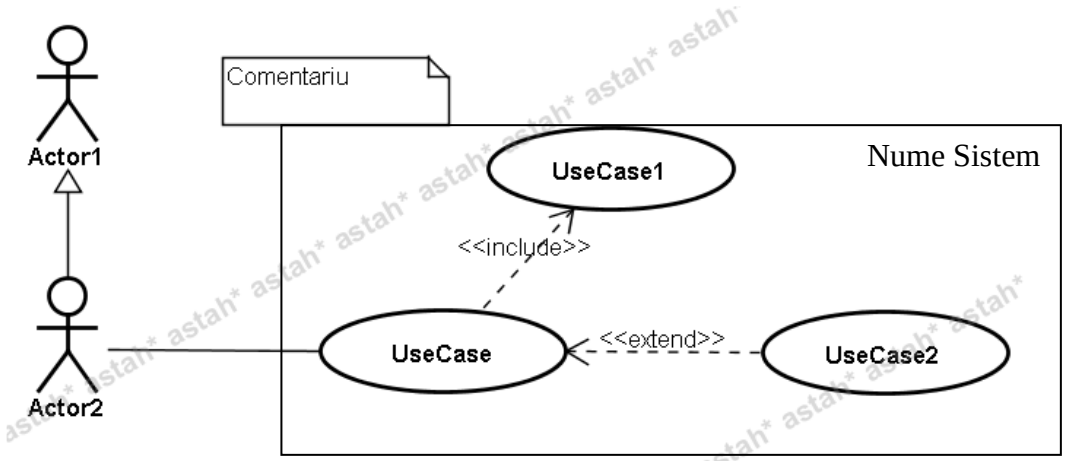

Figura 1 Diagrama cazurilor de utilizare

2.1. Cazuri de utilizare

Cazul de utilizare reprezinta o colectie de scenarii posibile referitoare la comunicarea intre sistem si actorii externi, caracterizate de anumite scopuri. Aceste scenarii sunt definite ca secvente de pasi, carora le pot corespunde cazuri de utilizare de nivel inferior. Notatia UML pentru un caz de utilizare este o elipsa, in care apare numele cazului de utilizare.

Relatia de incluziune semnifica faptul ca un caz de utilizare include comportamentul descris printr-un alt caz de utilizare, intr-un mod analog cu apelul unei subrutine.

Relatia de extindere indica faptul ca un caz de utilizare este inserat in altul, dar numai in cazul indeplinirii unor conditii

.

2.2. Actori

Actorii reprezinta elementele externe care interactioneaza cu programul; ei nu fac parte din sistem si definesc multimi de roluri jucate in comunicarea cu acesta. Notatia UML este un om stilizat, sub care se trece numele actorului.

Dialogurile posibile intre un actor si sistem sunt descrise prin intermediul cazurilor de utilizare, care grupeaza in mod coerent secvente de tranzactii care sa conduca la rezultatul dorit de catre actor. Aceasta relatie de comunicare intre un actor si un caz de utilizare se modeleaza ca o **asociatie,** si are in UML simbolul obisnuit, o linie continua intre cele doua elemente.

Intre actori poate exista un singur tip de relatie: **generalizarea**. Daca un actor mosteneste un alt actor, el poate sa comunice cu aceleasi cazuri de utilizare ale sistemului ca si parintele sau. Notatia UML este o sageata cu linie continua, avand la capat un tringhi gol care puncteaza spre actorul parinte

3. Diagrama de clase

- Reprezentarea claselor
- Relatii intre clase

Diagramele de clase sunt grafuri, având ca noduri *clasele* şi ca arce *relaţiile* între acestea. Ele modeleaza aspectul static al sistemului.

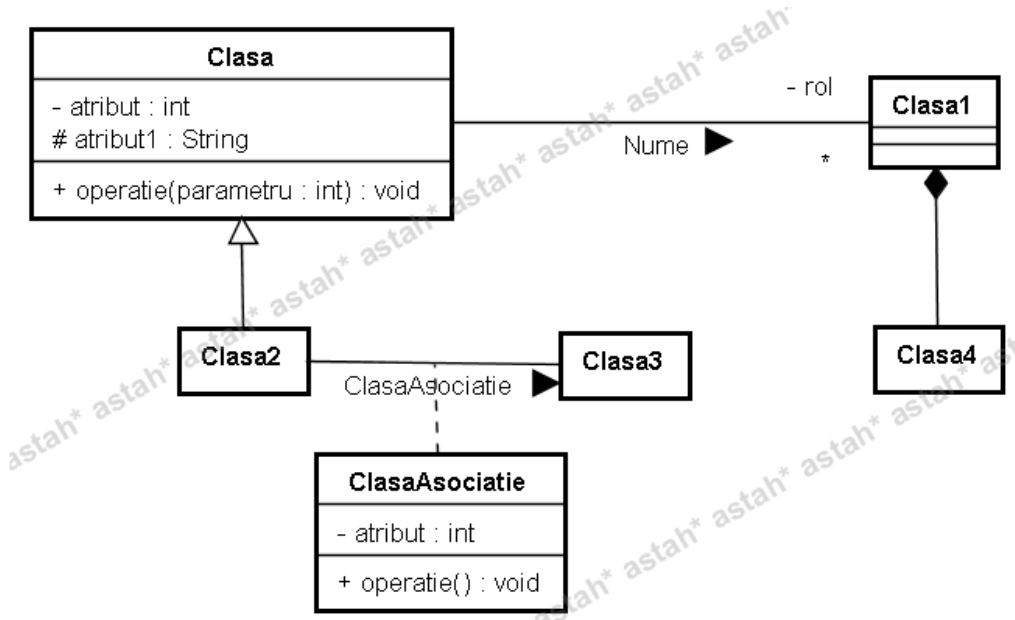

Figura 2 Diagrama de clase

3.1. Reprezentarea claselor

Numele clasei se trece intr-un dreptunghi, centrat si incepand cu litera mare. Dreptunghiul clasei poate fi impartit in 3 compartimente: (i) primul pentru **numele** acesteia; (ii) al doilea pentru **atribute**, pentru care se pot preciza tipul de date si o valoare implicita; (iii) al treilea pentru **operatiile** efectuate asupra atributelor, care sunt membre ale clasei; acestea pot avea o lista de argumente si un tip rezultat.

Se pot adauga detalii suplimentare, precum **vizibilitatea** atributelor si operatiilor clasei – modul in care acestea sunt vizibile si pot fi accesate din exteriorul clasei; se reprezinta printr-un caracter plasat inaintea denumirii membrului corespunzator, a carei prezenta este optionala; exista 3 tipuri de vizibilitate:

- **publica** - notata cu “+” – reprezentand atribute sau operatii care pot fi accesate din afara clasei;
- **privata** - notata cu “-“ - reprezentand atribute sau operatii care nu pot fi accesate decat de catre operatiile clasei respective, deci nu sunt vazute in exterior;
- **protejata** - notata cu “#” - reprezentand atribute sau operatii care pot fi accesate din clasele derivate, dar nu din alta parte din exteriorul clasei;

3.2. Relatii intre clase

Relatia de asociatie corespunde unei cooperari intre clase, in conditiile in care obiectele instantiate din aceste clase sunt create si distruse in mod independent. Cele mai multe asociatii sunt binare, fiind reprezentate printr-o linie trasata intre cele doua clase. Asociatia poate avea un **nume**, care este trecut langa linie, nu foarte aproape de capete, pentru a nu crea confuzii cu rolurile; pentru a sti in ce sens trebuie citit acest nume se poate adauga un triunghi plin, cu varful indicand sensul asociatiei. **Rolul** jucat de fiecare clasa in cadrul relatiei se poate reprezenta in imediata vecinatate a acesteia si este optional. **Multiplicitatea** unui capat al asociatiei arata cate obiecte instantiate din acea clasa pot fi asociate unui singur obiect de la celalalt capat al asociatiei. Multiplicitatea se specifica printr-o secventa de intervale de numere intregi, separate prin virgula; intervalele sunt de forma *LimitaInferioara..LimitaSuperioara*, pentru care se folosesc valori numerice sau caracterul “*” - in cazul unei limite superioare infinite.

Compozitia este relatia dintre o clasa care reprezinta un intreg si clasele din care se instantiaza partile sale componente; notatia este o linie terminate cu un romb plin in capatul dinspre intreg; intregul si componentele sale au timpi de viata egali, obiectele fiind instantiate (create), respectiv distruse simultan.

Relatia de generalizare apare atunci cand mai multe clase au proprietati commune; acestea sunt grupate intr-o superclasa, urmand ca ele sa fie incluse automat in toate subclasele derivate din ea. Notatia se efecteaza printr-o linie trasata intre clasa si subclasa, avand un tringhi gol, mare, cu varful punctand spre elementul mai general.

4. Diagrama secventiala

- Interactiunea intre obiecte
- Tipuri de mesaje si stimuli
- Secventierea in timp

Diagramele secvenţiale reprezintă interacţiunea, arătând, pentru obiectele care fac parte dintr-o anumită colaborare, secvenţa de stimuli transmişi între ele. Aceste diagrame cuprind şi o dimensiune temporală, deoarece fiecărui obiect îi corespunde o linie de viaţă, trasată vertical sub dreptunghiul ce conţine numele obiectului.

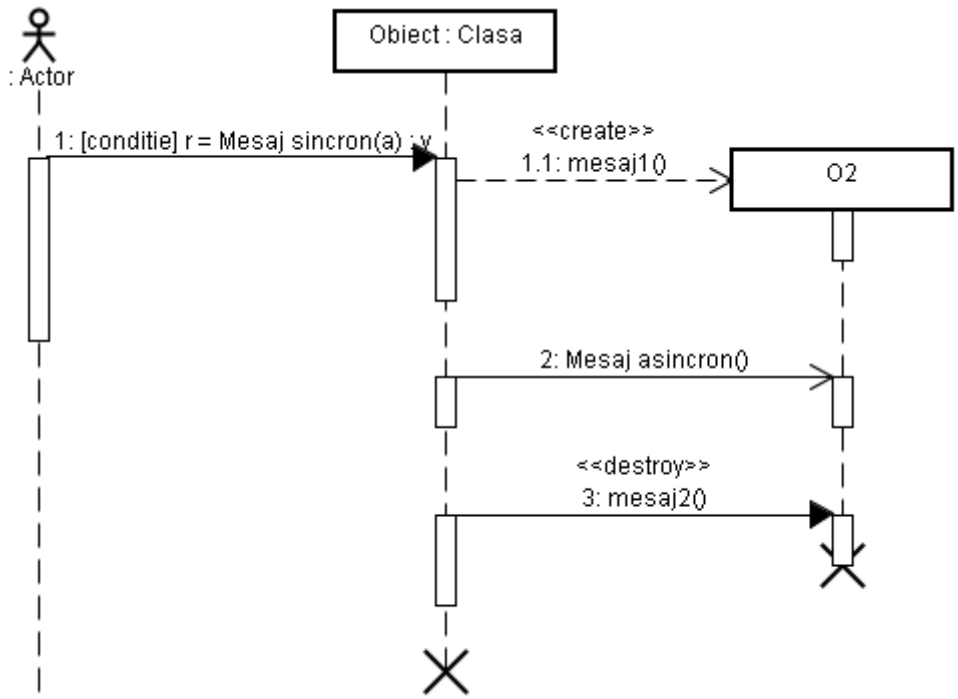

Figura 3 Diagrama secventiala

4.1. Reprezenta- rea interactiunii

Diagramele secventiale cuprind o **dimensiune temporala**; fiecarui obiect ii corespunde o linie de viata, trasata vertical sub dreptunghiul ce contine numele obiectului. Timpul se scurge de sus in jos, iar stimulii sunt reprezentati prin sageti etichetate, transmise de la linia de viata a obiectului sursa catre linia de viata a celui receptor.

Linia de viata a obiectului este reprezentata in mod obisnuit printr-o linie verticala intrerupta, ce coboara din dreptungiul obiectului. In intervalul de timp in care obiectul este activ, spre exemplu cand efectueaza o operatie, linia intrerupta este inlocuita cu un dreptunghi foarte subtire.

Un **stimul** poate constitui un semnal, un apel de operatie, crearea sau distrugerea unui obiect. Sagetile stimulilor puncteaza in sensul in care acestia sunt transmisi, fiind insotite de o eticheta. Aceasta cuprinde in general numele si, daca este cazul, argumentele sale; poate fi prezenta si o conditie pentru transmiterea stimulului. Reprezentarea acestor sageti este data in Tabelul 1, in functie de tipul de stimul.

Tabelul 1 Reprezentari ale stimulilor

Forma sageata	Semnificatie
	Apel de procedura, insemnand un stimul sincron, la care se asteapta raspuns
	Returnarea dintr-o procedura
	Stimul asincron, la care nu se asteapta raspuns

4.2. Crearea si distrugerea obiectelor

In decursul unei interactiuni pot fi create si distruse obiecte. Un obiect creat dupa inceperea interactiunii nu apare de la inceput, pe acceasi linie cu celelalte. Varful sagetii ajunge chiar la dreptunghiul care simbolizeaza noul obiect, dupa care acesta poate incepe imediat sa interactioneze cu celelalte obiecte si se traseaza linia sa de viata.

Un obiect poate fi de asemenea distrus in timpul unei interactiuni, caz in care linia sa de viata se termina si se marcheaza cu un ”X” mare; daca obiectele isi continua viata si dupa terminarea interactiunii, linia lor de viata se continua si dupa ultimul stimul transmis, adica dupa ultima sageata din diagrama.

5. Diagrama de comunicare (colaborare)

- Interactiunea intre obiecte
- Comunicarea intre obiecte

O diagramă de comunicare reprezintă de asemenea o interacţiune, printr-un graf având ca noduri obiectele care participă la colaborare şi ca arce legăturile dintre ele, însoţite de stimulii transmişi prin intermediul acestora. Numele utilizat în versiuni anterioare ale standardului este de diagramă de colaborare, el fiind încă utilizat în numeroase instrumente de editare a modelelor UML.

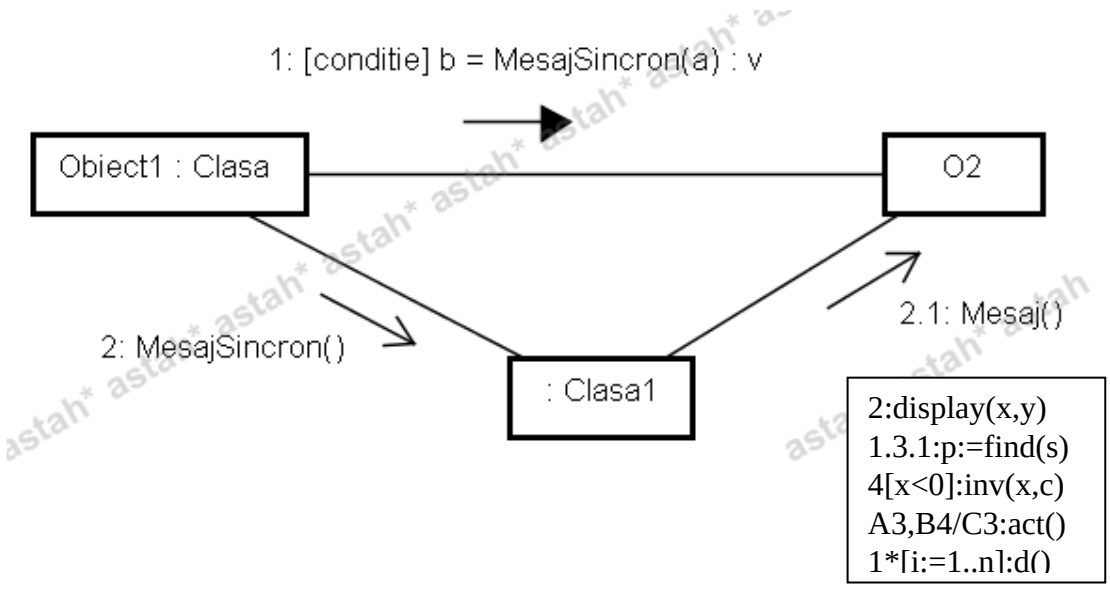

Figura 4 Diagrama de comunicare (colaborare)

5.1. Reprezentarea legaturilor intre obiecte

O diagrama de colaborare la nivelul instantelor este un graf, avand ca noduri obiectele care participa la colaborare si ca arce legaturile dintre ele, insotite de stimulii transmisi prin intermediul acestora.

Stimulii se reprezinta prin sageti mici, atasate legaturilor si indicand navigabilitatea diagramei. Standardul UML prevede 3 tipuri de sageti pentru reprezentarea stimulilor, prezentate in tabelul 1.

5.2. Caracterizarea stimulilor

Sagetile stimulilor sunt insotite de cate o eticheta avand sintaxa:

Predecesori Conditie Secventialitate ValoareReturnata := NumeStimul ListaArgumente

Predecesorul se reprezinta ca o lista de numere secventiale, separate prin ‘,’ si terminata prin ‘/’, spre exemplu *1,2 / Stimul*, pentru a arata ca mesajele 1 si 2 trebuie transmise inainte de Mesaj; aceasta notatie poate fi folosita pentru a indica sincronizarea taskurilor.

Conditionarea unui stimul se efectueaza printr-o conditie trecuta intre paranteze drepte, precum *[a>1] StimulConditionat*.

Secventialitatea consta dintr-o secventa de numere sau nume, ce pot fi urmate de indicatori de recurenta. Pentru un flux de control procedural se efectueaza o imbricare conform apelurilor imbricate de proceduri, cum ar fi *1.1.2:vr:=procedura(param)*. Pentru un control neprocedural, unde pot exista obiecte concurente, toata secventa denumirilor este pe acelasi nivel, fara imbricare, folosindu-se nume in loc de numere, de exemplu *a:afisare(x,y)*. In plus, o executie iterativa se specifica prin sintaxa ‘*’‘[‘clauza-iterare‘]’, rezultand etichete precum *mesaj*[I:=1..n]*.

6. Diagrama de stare

- Stari si actiuni
- Tranzitii si evenimente
- Stari compozite

Diagramele de stare sunt grafuri, ale căror noduri sunt stări şi ale căror arce direcţionate sunt *tranziţii*, având ca etichete numele evenimentelor care le-au provocat. Se precizează de asemenea *acţiunile* rezultate din aceste schimbări. Diagramele de stare prezintă aspectul dinamic al sistemului şi au la bază două concepte importante:

- **stările**, ce caracterizează valori sau seturi de valori ale obiectelor şi
- **evenimentele**, ce constau din stimuli externi ce acţionează asupra acestora şi determină o tranziţie de la o stare la alta.

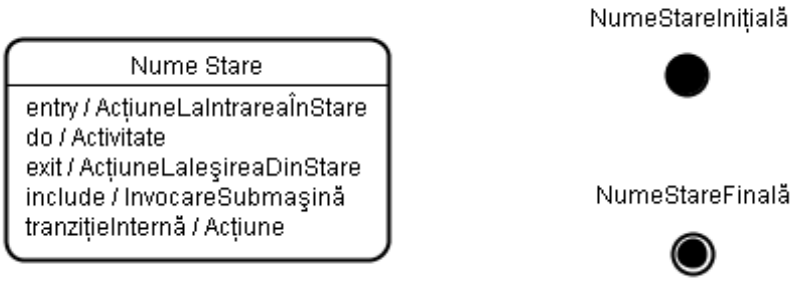

Figura 5 Reprezentarea starilor

6.1. Stari

Starea corespunde unui interval de timp in care obiectul satisface anumite conditii, efectueaza anumite actiuni sau asteapta un eveniment. Starea se reprezinta in UML printr-un dreptunghi cu colturile rotunjite, ce poate avea un compartiment pentru nume si altul pentru tranzitiile interne, care pot fi actiuni sau activitati efectuate in timp ce elementul se afla in acea stare. Actiunile sunt precedate de o eticheta urmata de caracterul '/', astfel:

- **entry** - pentru o actiune considerata instantanee, efectuata la intrarea in stare;
- **do** - pentru o activitate ce dureaza un interval de timp, fie pana ce este finalizata, fie pana se iese din starea corespunzatoare;
- **include** – pentru a identifica invocarea unei submasini;

- **tranzitieInterna** - pentru a reprezenta o actiune instantanee, efectuata la sosirea unui eveniment care nu determina tranzitia in alta stare;
- **exit** - pentru o actiune considerata instantanee, efectuata la iesirea din stare.

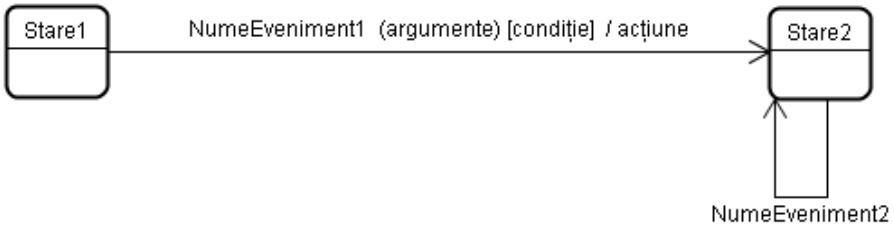

Figura 6 Diagrama de stare

6.2. Tranzitii intre stari

Trecerea dintr-o stare in alta poarta denumirea de tranzitie de stare; ea este considerata instantanee si nu poate fi intrerupta, spre deosebire de stari, care corespund unor intervale de timp; in diagramele de stare UML tranzitiile sunt reprezentate prin sageti etichetate cu urmatoarele elemente, ilustrate in Figura 6:

- *numele evenimentului* care a provocat tranzitia, care este provenit de la un alt obiect decat cel caruia ii corespunde diagrama, sau din exteriorul sistemului;
- *atributele elenimentului*, trecute intre paranteze rotunde;
- *expresia de conditionare a tranzitiei*, denumita pe scurt *conditie*, care trebuie indeplinita la sosirea evenimentului pentru ca tranzitia sa se poata produce; ea este trecuta intre paranteze drepte; expresia este scrisa in functie de parametrii evenimentului, atributele si legaturile obiectului caruia ii corespunde diagrama si starile concurente;
- *actiunea* sau *lista de actiuni* instantanee, care se petrec odata cu efectuarea tranzitiei, precedate in reprezentarea grafica de caracterul ‘/’.

.

7. Diagrama de activitati

- Stari actiuni
- Reprezentarea fluxului de actiuni ale obiectelor

Diagramele de activităţi sunt cazuri particulare ale diagramelor de stare, care nu descriu însă un flux de control bazat pe evenimente, ci unul procedural, în care toate sau majoritatea tranziţiilor se efectuează automat, la terminarea acţiunilor efectuate în interiorul stărilor. Aceste diagrame reprezintă un aspect dinamic al sistemului şi au importanţă în modelarea funcţiilor sistemului. În plus, oferă posibilitatea reprezentării paralelismului acţiunilor.

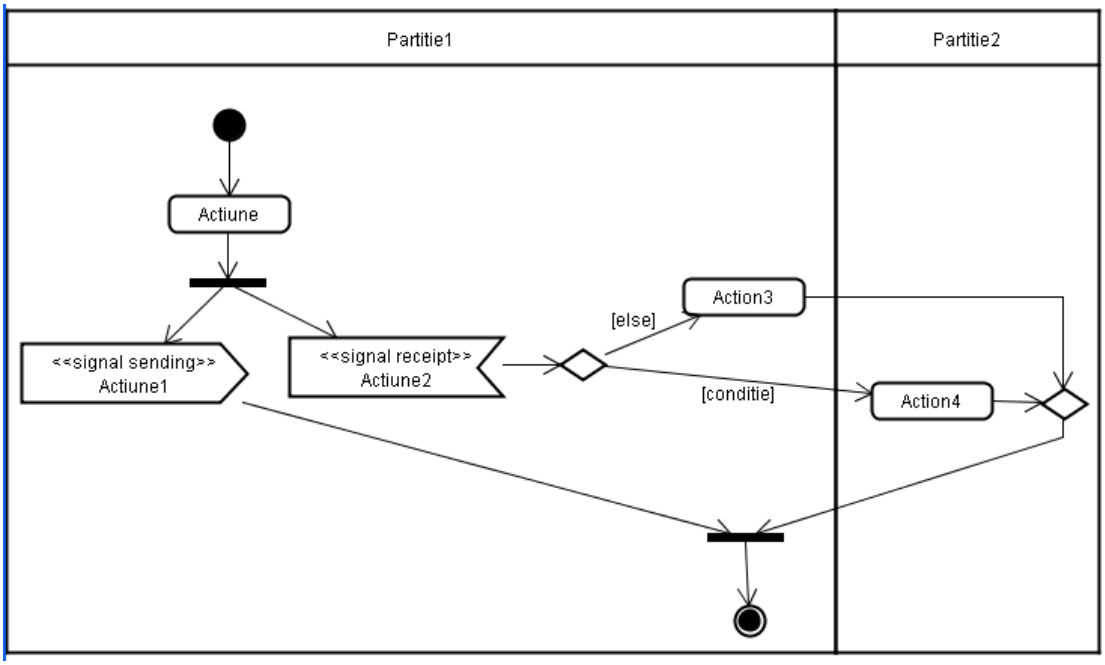

Figura 7 Diagrama de activitati

7.1. Tipuri de activitati

O **stare actiune**, spre deosebire de starile obisnuite, nu poate avea tranzitii interne sau tranzitii de iesire bazate pe evenimente; ea reprezinta o etapa din executia unui algoritm sau a unui flux de lucru, avand asociata o actiune de intrare, la terminarea careia se genereaza automat o tranzitie de iesire din stare. In locul dreptunghiului cu colturi rotunjite, standardul UML foloseste pentru starea actiune arce convexe in ambele parti.

Desi nu apar obisnuit, in diagrama de activitati pot fi prezente si **semnale** trimise sau primite de la diferite obiecte exterioare. Receptionarea unui semnal se reprezinta ca un dreptunghi din care se decupeaza un triunghi in partea sa dreapta; semnalul este trecut in interior. Trimiterea unui semnal se noteaza printr-un dreptunghi la care se adauga un triunghi in partea sa dreapta.

7.2. Fluxul de control

Fluxul de control se reprezinta in general prin **tranzitii automate** de la o stare la alta, care nu au asociate evenimente, pentru ca sunt provocate de terminarea actiunilor din interiorul starilor - activitate; ele pot avea insa expresii de conditionare trecute intre paranteze drepte. Reprezentarea se face prin sageti, la fel ca la diagramele de stare.

La fel ca si diagramele de stare, diagramele de activitati pot avea elemente care sa exprime conditionalitatea si concurenta. O posibilitate este introducerea **deciziilor**, pentru a indica mai multe tranzitii posibile care depind de o conditie Booleana asupra obiectului. Notatia este un romb, avand o singura tranzitie de intrare, care poate fi provocata de un eveniment, si doua sau mai multe iesiri insotite de cate o conditie; se poate folosi si conditia predefinita “else”. Pentru jonctiunea ramurilor decizionale se foloseste aceeasi notatie.

Concurenta impune, pe de alta parte, efectuarea unei sincronizari intre activitati paralele, dar starile de sincronizare pot fi omise, pentru ca ar avea intotdeauna o tranzitie de intrare, una de iesire si o limita superioara infinita. Astfel, la ramificare si la jonctiune apare doar segmentul de unde pleaca tranzitiile de iesire, respectiv la care ajung tranzitiile de intrare.

7.3. Asignarea responsbilitatii

Actiunile pentru care se introduc stari sunt efectuate de catre anumite obiecte si initiate de catre altele. Pentru a preciza responsabilitatile pe care acestea le au in cadrul controlului, diagrama se organizeaza pe benzi orizontale sau verticale, asignate activitatilor care sunt efectuate de catre obiectele unei clase. Benzile sunt separate la stanga si la dreapta prin linii verticale continue si au trecut in partea de sus numele clasei corespunzatoare. Activitatile de pe o banda pot fi executate de mai multe obiecte ale aceleiasi clase, iar tranzitiile pot trece de la o banda la alta.

8. Diagrama de componente

- Descompunerea sistemului in componente
- Asamblarea componentelor prin interfete

Diagramele de componente oferă o reprezentare statică a implementării sistemului şi arată organizarea şi legăturile unui set de componente. O *componentă* reprezintă un element fizic al unui sistem de programe, care grupează anumite elemente logice, realizând şi/sau solicitând anumite *interfeţe*.

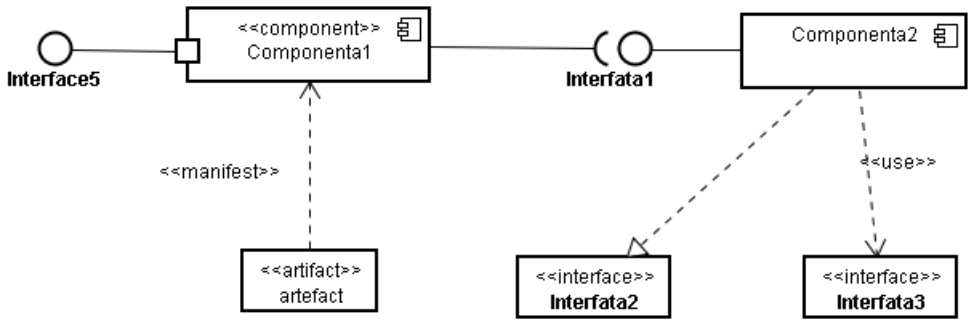

Figura 8 Diagrama de componente

8.1. Notatii UML

O componenta este reprezentata printr-un dreptunghi, avand stereotipul <<component>> si o icoana specifica. Ea poate avea doua tipuri de interfete:

- **interfete necesare** – notate printr-un semicerc conectat la componenta;
- **interfete furnizate** – notate printr-un cerc conectat la componenta.

Alternativ, interfetele pot fi reprezentate prin dreptunghiuri avand stereotipul <<interface>>.

Unei componente ii poate corespunde un **artefact** software, notat printr-un dreptunghi cu stereotipul <<artifact>>, care e conectat cu aceasta printr-o relatie de manifestare.

9. Diagrama de deployment

- Resurse fizice si conectarea acestora
- Repartizarea componentelor pe noduri de prelucrare

În general, se realizează diagrame de deployment atunci când sistemul este distribuit fizic. Ele sunt utile pentru arhitectul sistemului şi pentru inginerul de sistem, pentru a cunoaşte modul în care componentele sunt repartizate şi executate pe diferite noduri de prelucrare, permiţând planificarea unei topologii a procesoarelor şi dispozitivelor pe care se instalează sistemul.

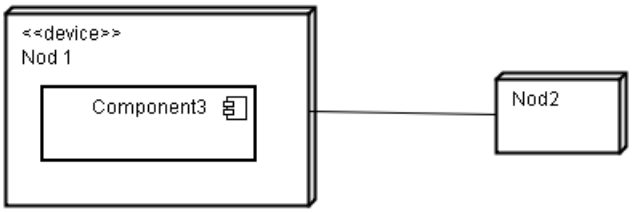

Figura 9 Diagrama de deployment

9.1. Notatii UML

Diagramele de deployment reprezintă o vedere statică asupra sistemului, arătând :

- resursele fizice disponibile – prin **noduri**, reprezentate ca paralelipipede dreptunghice ce corespund unor unităţi de calcul sau unor dispozitive inteligente;
- conectarea acestora – prin **conectori**, care sunt linii simple asemănătoare asociaţiilor şi
- repartizarea componentelor pe noduri.

10. Exemple de editoare UML

PRODUS	COMPANIE	ADRESA WEB
Astah	Change Vision	http://astah.net/
Enterprise Architect Professional 3.1	Sparx Systems	http://www.sparxsystems.com.au/ea.htm
MasterCraft Component Modeler 5	MasterCraft	http://www.tata-mastercraf.com/overview.asp
mDesPro 5.5.2	Mountfield Computers	http://www.mfcomputers.com
Metamill 2	Metamill Software	http://www.metamill.com/product.html
ModelMaker 6.2	ModelMaker Tools	www.modelmakertools.com/mm-features.htm
Object Domain Standard 3	Object Domain Systems	http://www.objactdomain.com/domain/index.html
StarUML	GPL (GNU Public License)	http://staruml.sourceforge.net/en/about.php
Structure Builder Enterprise 4.5.4.	WebGain	http://www.webgain.com
Visual UML Standard Edition 2.9.2	Visual Object Modelers	www.visualuml.com/Products.htm