2. INGINERIA CA TIP DE CUNOAŞTERE

Ingineria este mai mult decât calcul. Soluţia numerică nu are nici o valoare dacă modelul unei probleme este neadecvat sau dacă problema în sine este greşit formulată. Vom acorda importanţa cuvenită etapelor premergătoare calculului numeric propriuzis, prin expunerea unor aspecte esenţiale în rezolvarea problemelor inginereşti.

2.1. Gândirea sistemică

Realitatea este o măsură perceptibilă a universului accesibilă cunoaşterii umane. Oricare lucru sau fapt din lumea reală este perceput într-un ansamblu de alte lucruri şi fapte între care există relaţii de interdependenţă. Indiferent de scara spaţio-temporală la care percepem realitatea nu există nimic izolat şi independent, adică nimic în afara unui sistem. Definiţia sistemului este următoarea:

> Un ansamblu de *elemente* distincte, denumite *unităţi*, între care există *legături* denumite *relaţii de interacţiune şi interdependenţă*, care funcţionează ca un întreg organizat în sensul atingerii unui obiectiv.

Pornind de la definiţia generală, sistemele se pot clasifica după mai multe criterii. Fără a avea pretenţia de a epuiza acest subiect vom da câteva exemple de clase de sisteme. În funcţie de *natura elementelor* componente există două clase mari de sisteme: fizico-tehnice şi economico-sociale. În funcţie de *geneza* sistemelor acestea sunt fie naturale, fie artificiale. În funcţie de *tipul dominant al elementelor* componente există sisteme omogene şi sisteme eterogene. După *criteriul topologic* pot exista sisteme locale şi sisteme distribuite. Modul de manifestare a relaţiilor (legăturilor) dintre elementele unui sistem ţine de *organizarea* acestuia, în funcţie de care există sisteme ierarhice sau sisteme heterarhice.

Gândirea sistemică presupune adoptarea unui tip integrator de înţelegere asupra tuturor elementelor şi relaţiilor dintre ele din perspectiva legăturilor funcţionale între intrări şi ieşiri. Înţelegerea, ca proces cognitiv

prin care se formează gândirea sistemică are un rol esențial, atît în obținerea cunoştintelor despre sisteme, cît şi pentru formarea conceptelor. Gândirea sistemică se poate forma pe baza unor cunoştințe dobândite prin patru moduri distincte: *rațional*, *empiric*, *teoretic* şi *intuitiv*. Cunoştințele dobândite prin toate aceste metode conduc la o înțelegere integratoare asupra realității, în care *perspectiva holistică* are un rol important. Practic, întregul edificiu al cunoaşterii umane se bazează pe gândirea sistemică, ale cărei componente cognitive sunt schematizate în figura 2.1. Pe de altă parte, principiile gândirii sistemice stau la baza metodologiei de structurare a cunoştințelor în orice domeniu.

În cadrul unui sistem, elementele constituente şi legăturile fizice alcătuiesc **structura** sistemului. Interacțiunile şi interdependențele dintre elemente definesc **funcționalitatea** sistemului. Componența şi dispunerea elementelor (inclusiv legăturile fizice), în sensul organizării, definesc **arhitectura** sistemului.

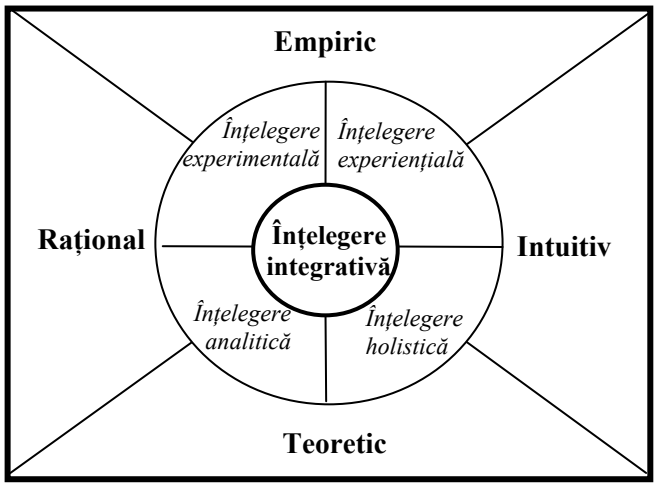

Fig. 2.1. Tipuri de înțelegere

Gândirea sistemică se bazează pe două metode generale de abordare fundamental diferite, dar complementare: *analiza*, respectiv *sinteza*.

Ca metodă generală în cercetarea realității, **analiza** presupune descompunerea sistemelor în unitățile componente şi tratarea fiecărei componente în contextul legăturilor sale cu celelalte.

Sinteza reprezintă metoda ştiințifică generală prin care se reunesc în întregul din care au făcut parte sau într-unul nou elemente obținute printr-o analiză prealabilă. Sinteza este specifică activității de proiectare a sistemelor. În figura 2.2 este prezentată sugestiv acțiunea complemetară şi reversibilă a celor două metode fundamentale de abordare a sistemelor. Se remarcă efectul de descompunere, respectiv de unificare a componentelor.

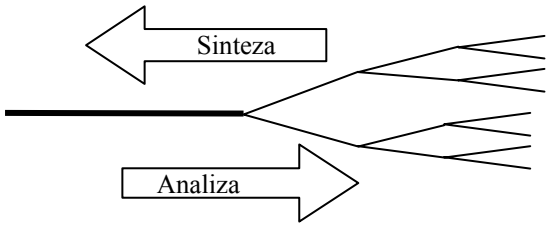

Fig. 2.2. Efectul comparativ al analizei şi sintezei.

În funcție de perspectiva asupra sistemului, analiza permite următoarele abordări:

ANALIZA { *Structurală*; *Funcțională*; *Arhitecturală*

Analiza structurală vizează înțelegerea modului în care sunt interconectate şi corelate lucrurile, pornind de la cunoaşterea fiecărei entități a sistemului supus analizei.

Analiza funcțională se face în scopul înțelegerii modului în care lucrează fiecare element al sistemului considerat în conexiune cu celelalte.

Analiza la nivel de arhitectură vizează investigarea comportamentului sistemului ca întreg în contextul atingerii obiectivului pentru care a fost creat.

2.2. Conceptul de proces

Din punct de vedere formal, sistemele sunt caracterizate de **parametri** *structurali*, *funcționali* şi de *arhitectură*. Parametrii sunt mărimi fizice variabile şi/sau constante ce definesc **starea** sistemului. În general, starea unui sistem depinde de timp, astfel că aceasta este caracterizată de o anumită **dinamică**. Deoarece, potrivit definiției, scopul funcțional al oricărui sistem este atingerea unui obiectiv, dinamica sistemelor este considerată **evoluție**. Se spune că un sistem evoluează, trecând succesiv printr-o mulțime de stări, într-un spațiu definit de variabilele sale de stare. Evoluția este cel mai general proces în cadrul sistemelor, având la bază o multitudine de fenomene care concură la dinamica acestora.

Definiție. Prin proces se înțelege un ansamblu de fenomene care evoluează realizând o transformare de stare a sistemului. Alte noțiuni asociate conceptului de proces sunt:

- succesiune de operații, stări sau fenomene prin care se realizează o lucrare (de exemplu un proces tehnologic);
- evoluție, dezvoltare, desfăşurare, acțiune;
- evoluție cu început şi sfârşit având un rezultat perceptibil/utilizabil.

Entitatea de bază a procesului este **fenomenul**. Fenomenul este manifestarea exterioară, perceptibilă a esenței unui lucru, care este asociată unui eveniment. Aşa cum se sugerează în figura 2.3, orice proces este determinat de o mulțime a fenomenelor asociate, dar nu reprezintă o simplă însumare a acestora. Orice proces se manifestă în cadrul unui sistem şi nu poate fi detaşat de conceptul sistemic.

Tipologia proceselor se poate clasifica în funcție de natura fenomenelor asociate şi de tipul sistemului cadru care le generează:

- procese naturale sau artificiale,
- procese fizice: mecanice, electrice, chimice, termodinamice, cuantice;
- procese fiziologice, psihice;
- procese economice;
- procese tehnologice;
- procese de decizie.

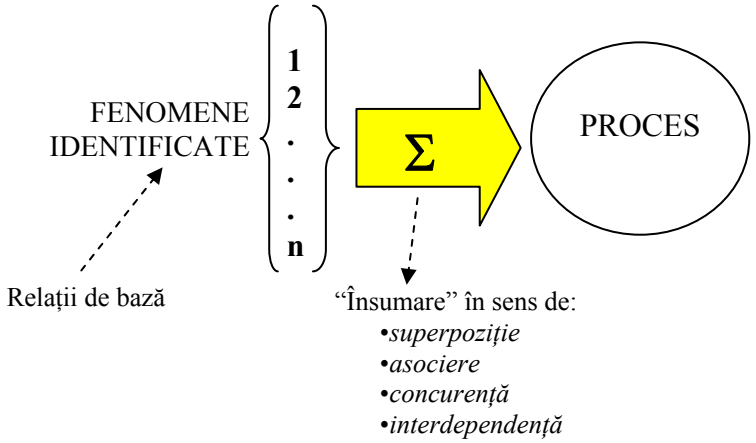

Fig. 2.3. O reprezentare intuitivă a conceptului de proces

Deşi, în sine nu este o noţiune abstractă, **procesul** poate fi *abstractizat*. Orice **proces** se manifestă în cadrul unui sistem care poate fi *modelat*. Prin urmare, *modelarea* este activitatea prin care reprezentăm sistemele în vederea înțelegerii proceselor care le guvernează dinamica.

Există două viziuni fundamentale în descrierea proceselor şi în general asupra percepției fenomenelor din natură: abordarea *continuă*, respectiv abordarea *discretă*.

Reprezentarea continuă a sistemelor este legată de percepția continuă a proceselor şi descrierea fenomenelor asociate prin funcții continue de timp ce descriu evoluția variabilelor sistemelor. Acest mod de abordare este cât se poate de natural deoarece timpul este variabila independentă continuă

fundamentală, iar toate sistemele prin variabilele lor de stare y evoluează după anumite legi în timp exprimate prin funcţii de forma $f(t) = y$.

Abordarea discretă nu schimbă esenţa continuă a proceselor şi nici natura sistemelor. Reprezentarea discretă presupune extragerea informaţiei despre starea unui sistem numai la anumite momente distincte de timp t_k, $k=1,2,...,n$. Astfel, sistemele şi procesele pot fi modelate prin funcţii discrete de forma $f(t_k) = y_k$.

Valorile variabilelor din sisteme influenţează decisiv procesele care au loc şi de aceea disponibilitatea datelor şi precizia acestora determină tipul de proces în sine. Astfel, dacă datele sunt disponibile şi se cunosc cu precizie avem de a face cu procese, respectiv sisteme *deterministe*. Dimpotrivă, dacă datele sunt obţinute printr-un număr finit de măsurători şi există un anumit grad de imprecizie asupra rezultatelor, atunci avem de a face cu procese *probabiliste*. Tipul datelor ce definesc parametrii unui sistem şi variabilele ce intervin într-un proces determină diferite clase sau tipuri de modele ale acestora.

Prin urmare, în funcţie de tipul de model utilizat procesele pot fi clasificate în următoarele clase:

- procese discrete;
- procese continue;
- procese deterministe;
- procese aleatoare sau probabiliste (stochastice);

Orice produs tehnic pe care îl concepem şi îl realizăm fizic constituie un sistem, care va funcţiona în simbioză cu un mediu specific. Prin urmare, este necesar să se studieze întregul *ansamblu de fenomene* ce stau la baza funcţionării sistemelor tehnice şi a *interacţiunii* acestora cu mediul.

Ca metodă generală de investigare a realităţii, analiza se aplică la nivelul proceselor prin descompunerea acestora în fenomene asociate şi studierea lor separată. Se caută să se stabilească ponderea tipurilor de fenomene ce contribuie la evoluţia unui sistem. Analiza proceselor constituie un caz particular de analiză funcţională, care necesită cunoştinţe multidisciplinare, inclusiv de programare.

Esenţa oricărui proces este **principiul cauzalităţii.** Potrivit acestui principiu fundamental al naturii, fiecare fenomen, indiferent de natura sa, decurge în baza unor relaţii cauză-efect. Prin urmare întreg spectrul fenomenologic al lumii reale se sprijină pe un edificiu de relaţii de

cauzalitate mai mult sau mai puțin complex, mai mult sau mai puțin cunoscut şi explicat, aşa cum se sugerează în figura 2.4.

Din punctul de vedere al puterii de investigație a proceselor, eficiența analizei ca metodă generală se loveşte de *gradul de cunoaştere asupra relațiilor cauză-efect* de la baza fenomenelor. Astfel, în realitate se întâlnesc următoarele situații:

- cauza şi efectul sunt cunoscute (fenomene evidențiate);
- cauza necunoscută-efectul cunoscut (fenomen neexplicat/paranormal);
- cauză predictibilă şi efect aşteptat (fenomen potențial/anticipat);
- cauze şi efecte încă necunoscute (fenomene nemanifestate).

Metodologii tipice pentru analiza proceselor:

- discriminarea (deosebirea);
- deducția (în opoziție cu inducția, care este specifică sintezei);
- analogia;
- descompunerea;
- divizarea;
- caracterizarea;

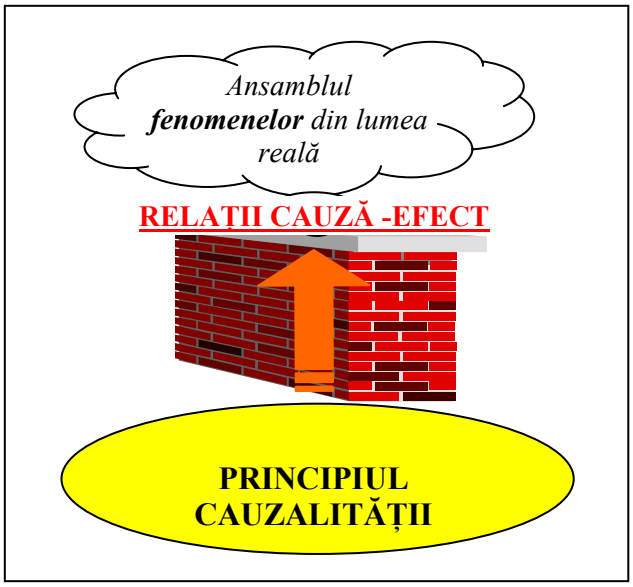

Fig. 2.4. Edificiul relațiilor cauză-efect ca bază a fenomenelor lumii reală

Strategii de analiză:

- *analiza elementară*: printr-o transformare a unității în dualitate, astfel: o noțiune în două abordări, un sistem în două părți, un proces în două fenomene;
- pornind de la o noțiune se dezvoltă mai multe puncte de vedere;
- plecând de la un aspect se dezvoltă multiple varietăți succesive în ordine genealogică (model arborescent, vezi figura 2.2).

Analiza proceselor, ca şi analiza de sistem este strâns legată de conceptul de modelare. Modelarea se efectuează prin abstractizare pe bază de raționamente de tip formal şi analogic şi utilizarea unor reprezentări simbolice. Modelarea matematică este cea mai utilizată în analiza sistemelor şi proceselor şi se bazează pe *relații analitice*. De regulă, se folosesc şi tehnici complementare de modelare bazate pe instrumente grafice sau combinate numite *metode grafo-analitice*. Conceptul "*black box*" (cutie neagră) este un instrument de modelare în teoria sistemelor fiind foarte popular în analiza şi simularea de sisteme şi mai ales în analiza proceselor complexe.

Instrumentele specifice pentru reprezentarea (modelarea) proceselor folosesc în general un puternic suport grafic:

- teoria grafurilor (grafuri de stare, grafuri de caz, grafuri de fluență);
- rețele Petri;
- diagrame (specifice metodologiei Unified Modeling Language-UML, diagrame de tip Stateflow).
- scheme structurale cu blocuri funcționale (organigrame).

2.3. De la problemă la soluție

Modul în care este formulată o problemă practică are un rol esențial în inginerie şi aceasta trebuie făcută din perspectiva cunoaşterii sistemice.

Rezolvarea multitudinii de probleme legate de diverse sisteme cu care omul operează presupune o metodologie bazată atât pe *analiză*, cât şi pe *sinteză*. Analiza cât şi sinteza sistemelor ridică o problematică diversificată care se clasifică în domenii distincte. Astfel, prin analiză se abordează probleme de tipul: *verificare*, determinarea limitelor de *capabilitate funcțională*, studiul *senzitivității* la diverşi factori, *răspunsul* sistemelor în diferite condiții, *identificarea* sistemelor, etc.. Sinteza ridică probleme specifice proiectării sistemelor: *dimensionarea*, proiectarea *robustă*, *optimizarea*, etc.. Problema devine astfel entitatea de bază în orice abordare sistemică, iar de formularea ei depinde întregul demers de analiză sau sinteză pentru sisteme şi procesele asociate lor.

În continuare, prezentăm trei categorii reprezentative de probleme care se întâlnesc în mod curent în activitatea inginerească.

a) *Dimensionarea* reprezintă categoria de probleme cea mai întâlnită în inginerie fiind specifică în activitatea de proiectare. Se dau anumite date de intrare şi pentru ieşiri impuse sistemului se determină prin calcul parametrii dimensionali necesari ai elementelor din sistem.

Un exemplu elementar: determinarea rezistenței unui dispozitiv de încălzire care să disipe o putere impusă, pentru o tensiune de alimentare dată. Fie P_{nec} puterea necesară impusă la proiectare şi U_n tensiunea nominală de alimentare. Dimensionarea rezistorului se face cu relația: $R_{nec} = \frac{U_n^2}{P_{nec}}$.

b) *Verificarea* reprezintă o categorie de probleme în care se cunosc date reale despre un sistem şi se verifică prin calcul dacă anumiți parametri sunt în limitele fizice admisibile. De exemplu, pentru un sistem format dintr-o sursă de tensiune conectată la o sarcină rezistivă să se verifice dacă în circuitul electric astfel format

nu se depăşeşte puterea admisibilă a rezistorului. Fie U tensiunea sursei de alimentare, R_n rezistența nominală a sarcinii şi P_{dn} puterea electrică maximă ce poate fi dezvoltată de un rezistor. Verificarea constă în testarea condiției: $P = \left(\frac{U^2}{R_n} \right) \le P_{dn}$.

c) *Determinarea sarcinii admisibile* constituie categoria de probleme inginereşti prin care se calculează solicitarea maximă (sarcina admisibilă sau capabilitatea) pe care o poate suporta un element cunoscut în condiții de funcționare date. De exemplu, curentul maxim suportat de un rezistor în condiții de funcționare îndelungată se calculează cu relația: $I_{\max} = \sqrt{\frac{P_{dn}}{R_n}}$.

În general, entitățile conceptuale, respectiv etapele fundamentale în rezolvarea unei probleme se pot schematiza astfel:

Cunoştințe → **Model** → **Algoritm** → **Rezolvare** → Soluție

Aşadar, *modelul* problemei este prima formă în care este structurată cunoşterea în vederea rezolvării acesteia. Modelul este o componentă necesară dar nu şi suficientă pentru rezolvarea unei probleme. Pentru găsirea soluției (sau soluțiilor) unei probleme este necesar să se elaboreze un *algoritm*. Algoritmul reprezintă modul în care se operează cu elementele modelului în prezența datelor, în sensul obținerii soluției pentru problema abordată. Cu alte cuvinte, algoritmul cuprinde paşii efectuați pe calea rezolvării problemei. De regulă, se pot gasi mai multe căi de rezolvare a problemei, deci în general, există algoritmi distincți ce vor conduce la aceeaşi soluție a problemei date sau la soluții sensibil apropiate. În practică se pune problema *eficienței* algoritmilor, la fel cum pentru modele se ridică problema *adecvanței* sau *acurateții (fidelității)* acestora. În general, algoritmul depinde în mod esențial de modelul problemei.

Majoritatea covârşitoare a abordărilor în descrierea naturii sunt dominate de concepția tradițională, în virtutea căreia cunoştințele noastre

despre sisteme şi procesele din cadrul acestora sunt formalizate matematic prin *relaţii exprimate analitic*. Descrierile aritmomorfe admit rezolvări pur numerice, astfel că aceste tipuri de modele devin în final puternic dependente de date. În această suită de modele analitice, cele bazate pe calculul diferenţial sunt dominante. De regulă, algoritmii de rezolvare a modelelor matematice diferenţiale se bazează pe scheme de calcul iterativ, implementate prin diferite metode numerice de rezolvare.

În acest context, rolul instrumentelor software şi a mediilor de calcul specializate pentru analiza sistemelor şi proceselor precum şi în proiectarea de sisteme este indiscutabil. Cvasitotalitatea problemelor de analiză a sistemelor şi proceselor conduc către abordări software concretizate prin programe de aplicaţie. Astfel a devenit posibilă rezolvarea unor probleme complicate, care au la bază modele tot mai complexe şi date mult mai precise.

Există criterii de bază în rezolvarea problemelor, care constituie în acelaşi timp şi parametrii de apreciere a adecvanţei modelelor şi eficienţei algoritmilor. Acestea sunt:

- *precizia* soluţiei (soluţiilor);
- *completitudinea* (dacă toate soluţiile posibile au fost găsite);
- *timpul* de rezolvare a problemei.

2.4. Implementarea software

Prin implementare software se înțelege scrierea unui program într-un limbaj de programare prin execuția căruia pe un sistem de calcul să se se obțină soluția problemei puse. Implementarea software este o etapă aproape indispensabilă în ingineria contemporană. Marea majoritate a problemelor în ştiinţă şi inginerie se finalizează prin aplicații software şi/sau utilizează medii de programare pentru calcul şi proiectare, astfel că modul de prelucrare a datelor devine dependent de algoritm.

Reamintim aici ecuația formală prin care Niklaus Wirth[1] a rezumat idea de program software:

Algoritmi+ Structuri de date=Programe

În figura 2.5 este prezentată o diagramă ce descrie fluxul etapelor ce se parcurg pe drumul de la problemă la soluție, cu utilizarea instrumentelor software pentru programarea aplicațiilor.

Universul problemei (UP) constituie mulțimea tipurilor de obiecte, obiectele în sine, relații între obiecte, precum şi restricțiile de rezolvare proprii problemei.

Metoda de rezolvare reprezintă mulțimea de *reguli* ce desemnează acțiunile ce trebuie efectuate asupra universului problemei, în vederea transformării acestuia conform scopului propus. Metoda indică doar *ce trebuie* făcut nu şi succesiunea de paşi pentru rezolvare.

Algoritm. Definirea generală a algoritmului se referă la un ansamblu de specificații precise (reguli) despre *cum trebuie* procedat pentru a rezolva o problemă. În particular, algoritmul poate fi definit sub două forme:

a) *proces dinamic* de calcul a unei funcții de rezolvare a unei probleme;
b) *secvență* (finită) de transformări asupra obiectelor problemei şi de directive de control aferente execuției secvenței.

[1] Profesor elvețian , creatorul limbajelor Pascal, Modula-2 şi contribuții deosebite în ştiința calculatoarelor.

Strategia de implementare software se adoptă din perspectiva conceptelor de programare. Scopul acesteia este de a obține *programul de aplicație* pentru problema în cauză adecvat analizei sau sintezei sub formă de cod scris de regulă într-un limbaj de programare de *nivel înalt*.

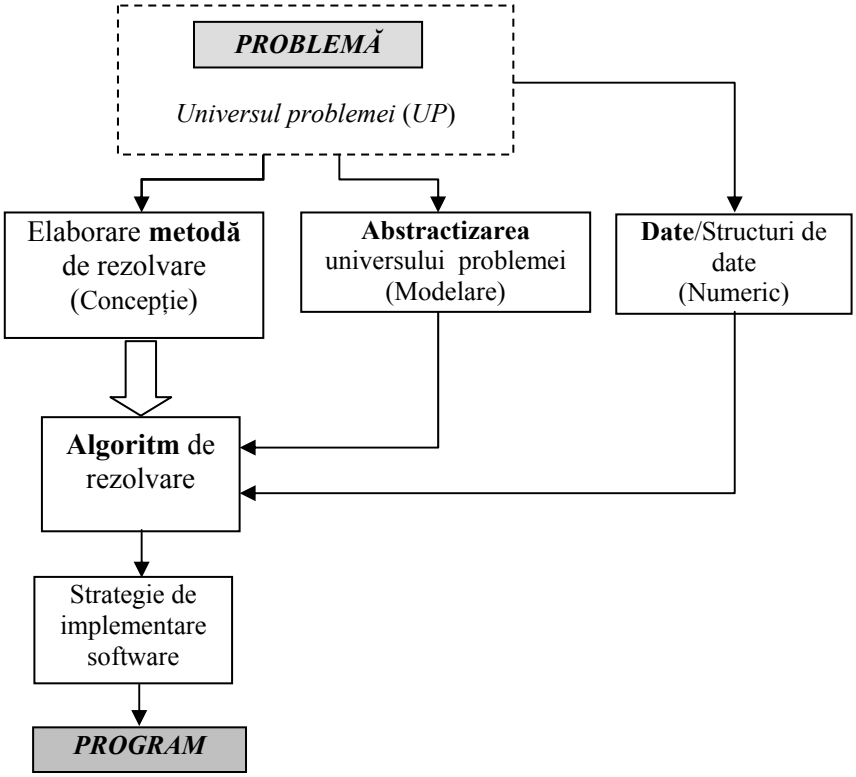

Fig. 2.5. De la problemă la program

În continuare sunt prezentate structurat, în evoluția lor, conceptele majore de programare. Acestea reprezintă paradigme de programare software care se adoptă în funcție de scopul şi destinația programului de aplicație final. Evoluția acestora este prezentată în tabelul 2.1.

Tab. 2.1. Evoluția conceptului de programare

	Programarea monolitică	*Programarea modulară*	*Programarea orientată pe obiect*	*Programarea orientată pe agent*
Unitatea de bază	Nemodulară	Modulară	Modulară	Modulară
Starea unității	Externă	Externă	Internă	Internă
Invocarea unității	Externă	Externă (apelata)	Externă (mesaj)	Internă (reguli, scopuri)

Prima etapă
Programarea monolitică

Unitatea de bază = întregul program
Starea unității de bază (controlul programului) = responsabilitatea programatorului
Invocarea unității de bază = determinată de operatorul de sistem

(vezi Tab.2.1)

A doua etapă
Programarea modulară

Pe masură ce programele au devenit mai *complexe* şi *spațiul de memorie a crescut* a apărut necesitatea de a introduce anumite *grade de organizare a codului*. Aceasta a constituit epoca *programării procedurale*.

Unitatea de bază = module mai mici reutilizabile în diferite situații particulare - subrutine proiectate să aibă un grad înalt de integrare locală.
Starea unității (controlul) = determinată de argumentele furnizate din exteriorul subrutinei "încapsulate".
Invocarea unității = prin instrucțiuni de apel din exterior.

(vezi Tab.2.1)

A treia etapă
Programarea orientată pe obiecte

Unitatea de baza = obiectul ca segment de cod sau metodă.
Starea unității (controlul) = prin controlul intern asupra variabilelor manipulate de către metode.
Invocarea unității = prin mesaje trimise metodelor de alte entități externe.

Observație: obiectele sunt considerate *pasive* deoarece metodele lor pot fi invocate numai din exterior.

(vezi Tab.2.1)

A patra etapă
Programarea orientată pe agent

Unitatea de baza = agentul software
Starea unității (controlul) = maniera proprie de control intern care îi conferă *autonomie*.
Invocarea unității = manieră interactivă bazată pe reguli şi scopuri interne prin care efectuează schimbul informațional sub forma de mesaje de complexitate mai mare.

Observații:
- Agenții sunt considerati *activi* - ei pot răspunde la un mesaj care invocă o metodă a sa sau pot inția o interacțiune cu exteriorul. Se spune că agenții au *inițiativă*.
- Agenții au propriul lor set de *responsabilități* definite intern în conformitate cu funcționalitatea lor.
- Agenții sunt priviți ca entități software cu următoarele proprietăți: *autonomi, activi şi reactivi, cu inițiativă, responsabili, comunicativi.*

(vezi Tab.2.1)