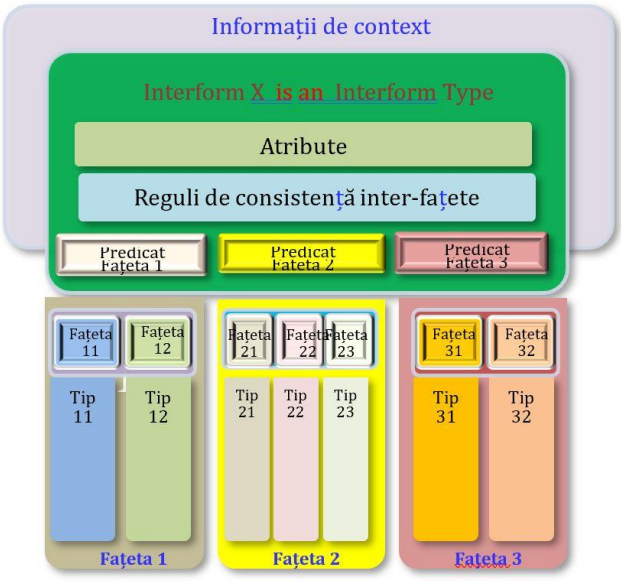
Despre interformă si maşină abstractă

1. Ce este o interforma: definiţie şi poziţionare conceptuală

Interforma este o construcţie inginerească originală, introdusă ca răspuns la complexitatea proiectării software în cadrul sistemelor mari. Ea reprezintă o **reprezentare stratificată, coerentă şi evolutivă a unei entităţi software**, aflate în proces de dezvoltare. Scopul său este să permită înţelegerea, proiectarea, comunicarea şi validarea acelei entităţi printr-o **pluralitate de faţete**, fără a sacrifica unitatea semantică a entităţii descrise.
Spre deosebire de modelele fragmentare (diagrame UML, specificaţii izolate, documentaţii ad-hoc), interforma funcţionează ca **o reţea de perspective (faţete) corelate**, fiecare reflectând un tip de cunoştinţe asupra entităţii modelate necesare pentru a construi, controla şi înţelege produsul software. Aceste faţete pot fi clasificate conform criteriilor inginereşti de modelare în:

- **structurale** – pentru a înţelege compoziţia şi organizarea internă a entităţii;
- **comportamentale** – pentru a urmări evoluţia în timp;
- **funcţionale** – pentru a descrie relaţia intrare–ieşire;
- **dinamice** – pentru a urmări dinamica entităţii în reacţie la stimuli şi sincronizarea cu alte sisteme;
- **semantice** – pentru a exprima semnificaţia entităţii în contextul aplicaţiei şi în raport cu lumea reală;

Articolul meu din 2008 „Către o proiectare dirijată de concerne” introduce explicit o idee fundamentală care în carte este doar implicită**:** fiecare faţetă nu este doar o perspectivă inginerească asupra sistemului, ci reflectă un concern explicit sau implicit al unui stakeholder (utilizator al sistemului sau dezvoltator). In felul acesta, fiecare stakeholder este interesat de o anume faţetă sau de un ansamblu coerent de faţete ale unei interforme. Dacă aceste faţete nu sunt identificate explicit, stakeholder-ul se regăseşte pierdut într-o reprezentare inutil de complexă sau, dimpotrivă, înşelător de simplificată.
Cu alte cuvinte, interforma mapează spaţiul cerinţelor stakeholderilor pe spaţiul deciziilor inginereşti: pentru fiecare aspect al entităţii respective important pentru un stakeholder (de exemplu, securitatea datelor) putem avea în interformă faţete structurale, comportamentale sau de alt fel care descriu asigurarea acelui aspect în entitatea considerată. Această aliniere între modelare şi preocupările celor interesaţi este ceea ce distinge net interforma de modelele clasice.



Reprezentarea grafica a unei interforme

Deci în structurarea faţetelor impusă de precedentele criterii inginereşti de modelare a entităţii de proiectat, se adaugă o noua dimensiune, cea a faţetelor datorate stakeholder-ilor.
Această **cartografiere între faţete şi concerne** permite:

- distribuirea clară a responsabilităţilor în echipă,
- trasabilitate decizională,
- posibilitatea unei analize de conflict între cerinţe contradictorii,

- integrarea timpurie a aspectelor non-funcţionale (calitate, securitate, ergonomie etc.),
- susţinerea ingineriei cerinţelor orientate pe viewpoint-uri.

Poziţionare conceptuală a interformei:

Criteriu	**Interformă**	**UML / MDA etc.**
Tip de reprezentare	Multifaţetată, structurată, coerentă	Diagrame izolate
Unitate semantică	Entitate unică, centrală	Fragmente de model
Relaţii între faţete	Explicite, coerente, verificabile	Implicite, separate prin standard
Legătură cu stakeholderii	Directă: fiecare fatetă corespunde unui concern	Indirectă, prin artefacte auxiliare
Evoluţie	Recursivă, incrementală, trasabilă	Adesea resetată între niveluri
Instrument formal	Abstract Machine (AM)	Lipsă sau constrângeri OCL

Astfel, **interforma nu este doar o unealtă inginerească**, ci şi **o infrastructură de comunicare între stakeholderi**, adaptată proiectării colaborative a sistemelor complexe.

2. Cum se construieşte şi evoluează o interformă

Interforma nu este un artefact static şi nici un document fix, ci un **obiect inginereşte viu**, care însoţeşte proiectarea pe întreaga durată a procesului software. Evoluţia sa reflectă **gândirea proiectantului**, **dialogul cu stakeholderii**, **testele de fezabilitate**, **alegerile tehnologice** şi **restricţiile externe**. Din acest motiv, **construcţia interformei este iterativă, recursivă şi adesea distribuită** în cadrul unei echipe.

2.1 Etapele naturale ale construcţiei

Construirea unei interforme parcurge, de regulă, următoarele stadii, nu neapărat în ordine strictă, ci într-o dinamică adaptivă:

Etapă	**Descriere**	**Instrumente posibile**
Iniţializare	Definirea entităţii de modelat şi a scopului proiectării	Limbaj natural, schiţe, meta-descrieri
Identificarea faţetelor	Alegerea perspectivelor relevante în funcţie de contextul inginerească şi de concerns	Liste de cerinţe, clasificări după stakeholderi
Popularea faţetelor	Completarea fiecărei faţete cu conţinut iniţial (adesea informal)	Note, scheme logice, tabele, diagrame parţiale
Rafinare formală	Introducerea progresivă a structurilor formale în descrierea faţetelor	grafuri, diagrame de stare, expresii logice
Integrare şi coerenţă	Corelarea între faţete, verificarea consistenţei şi a acoperirii cerinţelor	Abstract Machine, modele de coerenţă
Validare	Verificarea faţă de cerinţele stakeholderilor, prin simulare, recenzie sau prototip	AM, testare conceptuală, walkthrough-uri
Evoluţie	Îmbogăţirea, corectarea şi adaptarea interformei la noi informaţii şi decizii	Istoric al versiunilor, trasabilitate internă

Fiecare interformă este, astfel, o **unitate de lucru în evoluţie**, iar starea ei la un moment dat poate fi interpretată ca un „snapshot" al gândirii inginereşti asupra entităţii respective.

2.2 Relaţia cu versiuni şi istoric decizional

Într-o lume ideală, fiecare versiune a unei interforme ar trebui să reflecte un **stadiu clar al procesului decizional**: un punct în care o anumită faţetă a fost reformulată, o cerinţă a fost reinterpretată, sau o constrângere a fost modificată. Această **evoluţie este, în esenţă, diacronică**, şi poate fi urmărită cu o relaţie de tipul:
Interforma_t+1 = rafinarea/intervenţia asupra Interformei_t în faţeta X, justificată de Y
Această relaţie este nu doar istorică, ci **epistemică** – reflectă cum se acumulează cunoaştere în timp, şi cum aceasta este cristalizată în construcţia software.
Din acest motiv, **o interformă este mai apropiată de un concept viu decât de o fişă tehnică**: ea trăieşte în mintea proiectantului, în dialogul cu echipa, şi doar parţial în descrierile formale.

2.3 Exemple de instrumente pentru dezvoltarea interformei

Interforma, prin natura ei, **integrează instrumente de diverse tipuri**: unele formale, altele euristice; unele textuale, altele vizuale. Această flexibilitate o face **adaptabilă la nivelul de maturitate al proiectului**, la stilul echipei şi la nevoile concrete ale dezvoltării.

Componentă a interformei	Instrument tipic	Nivel de formalizare
Faţeta structurală	Diagrama de componente, de clase, graf de entităţi	mediu – înalt
Faţeta comportamentală	Maşini abstracte, diagrame de stare, modele de evenimente	înalt
Faţeta funcţională	Specificaţii I/O, diagrame de flux, scenarii de utilizare	mediu
Faţeta semantică	Ontologii, modele de business, descrieri logice	mediu – înalt
Faţeta socială / de utilizare	Personas, use-cases, evaluări de risc	scăzut – mediu

3. Recursivitatea interformei şi fundamentele modularităţii

Una dintre trăsăturile definitorii ale interformei este recursivitatea. O interformă nu este o construcţie atomică şi plată, ci un sistem de descriere stratificat, în care fiecare componentă — structurală, comportamentală sau procedurală — poate fi, la rândul ei, o interformă completă. Acest principiu reflectă o realitate profundă a proiectării inginereşti: niciun sistem complex nu poate fi înţeles sau construit în întregime deodată, iar proiectarea presupune o ierarhie de obiecte, fiecare cu autonomia sa semantică şi operaţională.
Această recursivitate derivă din însăşi natura complexităţii software-ului. În dezvoltarea de mari dimensiuni — programming in the large —, capacitatea de a împărţi raţional complexitatea în unităţi autonome şi de a le descrie coerent este o condiţie de supravieţuire a proiectului. Interforma acţionează aici ca o unitate logică de reprezentare, dar şi ca un model extensibil, ce poate încorpora alte interforme în mod explicit, păstrând consistenţa semantică la toate nivelurile.

a) Pe dimensiunea structurală

În plan structural, o interformă care descrie o entitate (sistem, subsistem, modul, clasă de obiecte) poate integra alte interforme aferente subcomponentelor sale. Această relaţie este analogă cu o compoziţie în arhitectura software sau cu agregarea de clase în limbajele orientate pe obiecte. Totuşi, interforma propune o generalizare semantică a acestui mecanism: orice parte a unei entităţi care are suficientă autonomie semantică poate deveni sub-interformă, cu propriile faţete şi propriul ciclu de dezvoltare.
Astfel, interforma oferă un cadru modular natural, în care:

- substructurile funcţionale sunt identificate ca unităţi semantice distincte;
- fiecare dintre ele este descrisă prin propriile sale faţete;
- întreaga construcţie menţine o coerenţă globală prin relaţii între faţete şi între interforme.

Într-un limbaj orientat pe clase, acest model ar putea fi exprimat ca o clasă abstractă extinsă, cu atribute faţetate şi metode conceptuale. Dar interforma nu urmăreşte doar organizarea codului — ci organizarea gândirii proiectante.

b) Pe dimensiunea comportamentală

În plan comportamental, o interformă poate include o descriere a funcţionării entităţii printr-o maşină abstractă. Comportamentele complexe sunt astfel descompuse în sub-comportamente, fiecare modelabil printr-o sub-maşină, care poate la rândul ei deveni o interformă. În acest sens, se poate construi o ierarhie de maşini abstracte care reflectă:

- niveluri de comportament (iniţializare, execuţie, eroare, finalizare);
- controlul tranziţiilor între stări;
- interacţiuni între sub-maşini (coordonare, paralelism, sincronizare).

Această abordare permite o modularizare a comportamentului în termeni de agregate dinamice şi oferă o cale de rafinare progresivă a logicii de funcţionare, până la nivel de cod executabil.

c) Recursivitate semantică şi trasabilitate

Interforma permite nu doar includerea altor interforme, ci şi urmărirea istoricului propriei evoluţii. Aceeaşi interformă poate trece prin mai multe versiuni — schiţă, prototip, versiune finală — fiecare reprezentând o etapă în rafinarea conceptuală. Relaţia „este_versiune_a” face posibilă urmărirea deciziilor inginereşti, justificarea lor şi eventuala reversibilitate.

Această recursivitate semantică este esențială pentru menținerea coerenței într-un proces evolutiv și adaptativ. Prin integrarea ei într-o bază de date de interforme — analogă unei baze de management al configurației —, întregul proces de proiectare devine trasabil, evaluabil și reconfigurabil.

5. Limbajul interformei, colaborarea și trasabilitatea deciziilor
A proiecta nu înseamnă doar a raționa, ci și a comunica. Iar interforma, deși născută ca instrument cognitiv individual, devine cu adevărat eficientă atunci când este împărtășită — în cadrul unei echipe, al unui ecosistem inginerească sau al unui lanț de decizie tehnică. Pentru aceasta, ea are nevoie de un **limbaj comun**, de o **formă de expresie convenită**, dar și de o **infrastructură de păstrare și urmărire a versiunilor**, pentru a sprijini colaborarea și coordonarea.

a) Limbajul interformei – între pseudocod și modelare fațetată
În sistemul INTERFORM, am propus un **limbaj de descriere bazat pe pseudocod**, extins cu o structură de tip CASE adaptată. Fiecare fațetă a unei interforme era exprimată în acest cadru printr-o „ramură" semantică:

- fațeta structurală putea conține definiții de structuri, liste de componente;
- fațeta funcțională – relația input/output;
- fațeta comportamentală – o descriere logică sau automată a tranzițiilor;
- fațeta semantică – corespondențe cu realitatea modelată;
- fațeta socială – utilizatori, scopuri, riscuri.

Aceste ramuri **nu erau disjuncte** — dimpotrivă, erau redundant complementare. Folosirea conceptului de CASE era o metaforă inginerească: nu pentru a selecta între alternative, ci pentru a descrie aceleași obiecte din unghiuri diferite. Această **redundanță controlată** devenea o strategie de înțelegere, nu un pericol de inconsistență. Fațetele nu erau doar viziuni diferite,

4. Limbajul interformei, colaborarea și trasabilitatea deciziilor
Pentru ca interforma să își îndeplinească rolul de pivot inginerească în cadrul proiectării software, ea trebuie să fie mai mult decât o abstracțiune conceptuală. Trebuie să fie exprimabilă, comunicabilă și manipulabilă într-un limbaj accesibil și convenit de membrii echipei de dezvoltare. Din acest motiv, interforma presupune nu doar un model de reprezentare, ci și un **limbaj de exprimare**, o **infrastructură colaborativă** și un **mecanism de trasabilitate decizională**.

a) Un limbaj inginerească de descriere: între pseudocod și modele formale
În contextul sistemului INTERFORM și al lucrării mele din 1992, am propus un limbaj de descriere a interformelor inspirat de pseudocodul procedural, dar extins semnificativ pentru a putea reda mai multe fațete simultan. Am utilizat, ca metaforă expresivă, mecanismul instrucțiunii CASE, însă nu pentru selecții exclusive — ca în limbajele clasice de programare — ci pentru **proiecții redundante și complementare** ale aceleiași entități software. Astfel, fiecare „caz" descria o fațetă diferită: structurală, comportamentală, funcțională, semantică. În acest limbaj:

- fațetele pot fi exprimate în limbaje specializate (UML, logică de tranziții, algebra de procese etc.);
- descrierile pot coexista chiar dacă sunt parțial redundante, atâta vreme cât nu sunt contradictorii;
- formalismul este dezirabil, dar nu obligatoriu — limbajul natural clar și riguros are un rol legitim în stadiile incipiente.

Acest limbaj a fost gândit ca un **mediu tranzitoriu**, capabil să acomodeze treptat niveluri crescânde de formalizare, pe măsură ce interforma avansează spre implementare.

b) Interforma ca unitate de lucru colaborativă
În proiectele software de mari dimensiuni, interformele devin **unități naturale de organizare a muncii în echipă**. Fiecare interformă corespunde unui obiect de interes inginerească — un modul, un subsistem, un serviciu — care poate fi proiectat, validat și întreținut în mod autonom. Prin urmare:

- interformele pot fi distribuite către echipe distincte;
- fiecare echipă lucrează pe baza unui set coerent de fațete;
- relațiile dintre interforme asigură coerența arhitecturii globale.

Această abordare susține **programarea în echipă pe subarbori semantici**. Se pot aloca interforme diferitelor echipe în funcție de specializare, și se pot integra ulterior rezultatele prin mecanisme standardizate de compunere a fațetelor.
În acest sens, interformele devin **unități de dezvoltare semantică**, depășind rolul de simplă documentație. Ele sunt instrumente active în comunicare, sincronizare și validare a construcției software.

c) Trasabilitatea deciziilor și gestiunea istorică

Una dintre marile promisiuni ale interformei este capacitatea sa de a păstra **memoria deciziilor inginereşti**. Fiecare versiune a unei interforme reflectă o etapă a raţionamentului proiectantului. Prin marcarea acestor versiuni şi legarea lor prin relaţii de tip este_o_versiune_a, putem construi o **reţea a evoluţiei proiectului**:

- fiecare decizie (structurală, algoritmică, funcţională) este înregistrată ca transformare a unei interforme;
- motivele deciziilor, constrângerile şi implicaţiile pot fi documentate contextual;
- traseul de dezvoltare este astfel **auditabil, justificabil şi reversibil**.

Un astfel de mecanism de trasabilitate este esenţial pentru proiectele critice sau de lungă durată, unde înţelegerea motivaţiilor trecutului este la fel de importantă ca implementarea prezentului. În cadrul unei infrastructuri inginereşti moderne, această reţea de interforme şi transformări poate fi integrată într-o **bază de date inginerească**, analogă sistemelor de gestiune a configuraţiilor (Configuration Management)

5. Maşina abstractă: controlul logic al deciziilor de proiectare

Dacă interforma este instrumentul de **descriere plurală** a unei entităţi software, maşina abstractă (Abstract Machine, AM) este instrumentul de **control al procesului de construcţie**. Ea reprezintă modul în care gândirea inginerească poate fi modelată operaţional, printr-un set de stări, tranziţii şi mecanisme de decizie care structurează evoluţia interformelor în timp.

Am introdus conceptul de **maşină abstractă** nu ca o simplă metaforă pentru un program de execuţie, ci ca o **maşină de proiectare**: un operator logic care guvernează paşii prin care un sistem software este construit, rafinat, evaluat şi validat.

a) De la formalism teoretic la aplicaţie inginerească

În literatura clasică, maşina abstractă desemnează un model formal care mapează datele de intrare în rezultate, printr-o succesiune de operaţii conforme cu o semnătură prestabilită. Aceste modele sunt esenţiale în teoria limbajelor formale, în semantica operaţională şi în definirea compilatoarelor.

În abordarea mea, am extins această noţiune pentru a descrie **comportamentul unui proiectant** în raport cu procesul de dezvoltare:

- AM devine un **agent de decizie** capabil să selecteze, să rafineze, să valideze interforme;
- fiecare stare a AM corespunde unei **etape inginereşti** (de exemplu, explorare, alternativă, selecţie, rafinare, validare);
- tranziţiile dintre stări sunt declanşate de condiţii verificabile sau de inputuri externe (de la stakeholderi, de la prototipuri, de la simulări).

În acest sens, maşina abstractă devine o **infrastructură logică** pentru desfăşurarea proiectării într-un spaţiu controlat, reproductibil şi inteligibil.

b) AM ca motor de proces decizional

Utilizată împreună cu interforma, maşina abstractă permite modelarea:

- **fluxului decizional** care guvernează tranziţiile între versiuni ale unei interforme;
- **raţionamentului inginereasc** ce conduce de la o soluţie parţială la o construcţie completă;
- **stărilor de validare** (prin simulare, testare conceptuală, comparare între faţete);
- **controlului calităţii** în timpul proiectării.

Maşina abstractă nu este un mecanism închis — ea este extensibilă. Proiectantul defineşte propriile operaţii aplicabile asupra interformei (de exemplu, „adaugă o componentă", „rafinează o faţetă", „combină două comportamente") şi stabileşte regulile de tranziţie. În acest mod, AM poate fi văzută ca un **agent semi-automat de proiectare**, un precursor al instrumentelor moderne de Design Automation şi, potenţial, al unei inteligenţe artificiale inginereşti.

c) Componibilitate şi ierarhizare

În proiectele de mari dimensiuni, nu o singură maşină abstractă guvernează întregul sistem, ci un **ansamblu ierarhic de maşini abstracte**:

- unele operează la nivel înalt, coordonând tranziţii între interforme de subsistem;
- altele operează local, guvernând rafinări fine asupra unei faţete sau componente;
- între maşini se stabilesc relaţii de **invocare, rafinare, substituţie sau paralelism**.

Această ierarhie de AM reflectă, în fond, stratificarea arhitecturală a unui proiect complex. În acelaşi timp, ea permite **distribuirea controlului** către diferite roluri inginereşti: arhitectul de sistem, inginerul de detaliu, analistul de cerinţe etc.

d) O punte către formalizarea procesului inginereasc

Rolul maşinii abstracte este acela de a transforma un proces adesea informal — proiectarea — într-un **spaţiu raţional de explorare şi validare**. Ea oferă:

- o **memorie operaţională** a paşilor parcurşi;
- un cadru pentru **evaluarea variantelor**;
- un mecanism pentru **simularea scenariilor** de proiectare.

Această orientare spre formalizare nu elimină creativitatea, ci o încadrează într-o disciplină inginerească. Proiectarea devine astfel o **activitate cu proprietăţi inginereşti**: predictibilitate, reproductibilitate, transparenţă decizională.

6. Interforma şi Abstract Machine: o arhitectură inginerească pentru complexitatea software

Interforma şi Maşina Abstractă nu sunt doar două concepte izolate – ele formează împreună un **cadru inginereasc coerent** pentru modelarea, controlul şi comunicarea în procesul de proiectare a produsului software. Relaţia lor este una de complementaritate profundă: dacă interforma este **obiectul gândirii inginereşti**, maşina abstractă este **mecanismul acestei gândiri** – ceea ce face posibilă transformarea viziunii în construcţie.

Această arhitectură dual poate fi descrisă în termeni simpli:

- **Interforma** organizează **ce ştim** despre o entitate software, prin faţete multiple, formalizate şi coerente;
- **Maşina abstractă** controlează **ce facem** cu această cunoaştere, cum o rafinăm, cum o testăm, cum decidem şi cum comunicăm.

În esenţă, interforma este modelul, iar maşina abstractă este procesul care operează pe model.

a) Coevoluţia între interformă şi AM

Cele două entităţi evoluează împreună:

- o **nouă faţetă** adăugată unei interforme presupune extinderea comportamentului AM care o gestionează;
- o **decizie inginerească** luată de AM (ex. alegerea unei arhitecturi) se reflectă într-o **modificare a unei faţete**;
- **validarea** unei versiuni a interformei este o tranziţie într-o **stare terminală** a unei maşini abstracte de proiectare.

Această coevoluţie garantează trasabilitate, consistenţă şi transparenţă: orice modificare are un motiv, orice decizie are un loc în istoria proiectului.

b) O infrastructură inginerească pentru "programarea în mare"

Modelul dual interform–AM a fost conceput în mod explicit pentru a răspunde provocărilor software-ului complex, acolo unde:

- produsul este rezultatul colaborării între echipe numeroase şi distribuite;
- cerinţele sunt dinamice, contradictorii, emergente;
- livrarea se face în etape, cu verificări intermediare şi adaptări frecvente;
- mentenanţa şi extensibilitatea sunt esenţiale pentru succesul pe termen lung.

În acest context, **interforma oferă un mecanism de reprezentare plurală şi structurată a produsului**, iar **maşina abstractă furnizează un mecanism de guvernanţă operaţională a construcţiei**. Împreună, ele permit o proiectare distribuită dar coerentă, formalizabilă dar adaptabilă, riguroasă dar deschisă dialogului.

c) Precursorul unei epistemologii inginereşti

Dincolo de aplicabilitatea lor practică, aceste două concepte propun şi o **filozofie inginerească**: ideea că **înţelegerea profundă precede implementarea eficientă**. Într-un moment istoric în care software-ul a devenit infrastructură critică pentru societate, o astfel de filozofie este nu doar utilă, ci necesară.

Interforma ne obligă să ne gândim în termeni de **concepte clar exprimate, multiperspectivice**, înainte de a desena diagrame sau de a scrie cod. Maşina abstractă ne invită să **raţionăm procedural**, să articulăm etapele prin care trecem, să păstrăm logica internă a parcursului proiectant.

Această epistemologie nu se bazează pe intuiţie, ci pe **construirea deliberată de reprezentări şi mecanisme** care fac raţionamentul inginerească vizibil, controlabil şi transferabil.

Concluzie – O moştenire neexplorată, un orizont redeschis

Privind în urmă la interformă şi maşina abstractă, nu pot să nu resimt o dublă realitate: pe de o parte, **aceste concepte au fost formulate cu o maturitate inginerească timpurie**, în avans faţă de nevoile şi capacităţile comunităţii profesionale din vremea lor; pe de altă parte, ele **nu au fost receptate, citate sau continuate**, rămânând izolate în paginile unei cărţi prea puţin cunoscute şi într-un sistem de proiectare – INTERFORM – care nu a fost adoptat pe scară largă.

Și totuși, această izolare nu le-a anulat valabilitatea. Dimpotrivă: odată cu trecerea timpului și creșterea complexității software-ului contemporan, **nevoia de structuri conceptuale clare, formale, integrative și navigabile** este mai acută ca oricând. Marile ecosisteme software din ziua de azi – sisteme de sănătate digitală, infrastructuri de mobilitate autonomă, platforme integrate de date și servicii – sunt atât de vaste și dinamice încât nu pot fi gestionate doar prin instrumente fragmentare și modele vizuale dispersate.
Aici, interforma – în calitate de **agregator semantic și formal al tuturor fațetelor unui produs software** – își regăsește relevanța. Iar mașina abstractă – ca **mecanism de raționament procedural, validare și guvernanță a deciziilor** – poate deveni parte din infrastructura logică a unei noi generații de sisteme inteligente de asistență în proiectare.
Mai mult, interforma prefigurează o cale de legătură între **modelarea inginerească tradițională** și **viitoarele forme de design colaborativ susținut de inteligență artificială**. Un digital twin de proiectare, așa cum este azi imaginat în industrie, ar putea fi văzut ca o interformă vie: capabilă să evolueze, să memoreze deciziile, să genereze versiuni alternative și să comunice între roluri și straturi organizaționale.
Această posibilă redescoperire nu ar trebui însă să vină fără rigoare. Ea cere un efort de **reinterpretare, formalizare și implementare** a ideilor într-un context nou, dar păstrând **esența lor inginerească**: aceea de a structura complexitatea, nu de a o ascunde; de a clarifica procesul decizional, nu de a-l înlocui cu opacitatea algoritmică.
Închei acest capitol nu cu un epilog nostalgic, ci cu o **chemare către o generație viitoare de arhitecți software**: să redescopere aceste instrumente nu ca relicve, ci ca **repere posibile într-o reconstrucție conceptuală a designului inginereasc**. Poate că ideile de aici au fost prea devreme. Poate că nu au avut ocazia de a intra într-o tradiție. Dar **în ele trăiește o ambiție care rămâne vie**: aceea de a transforma proiectarea software-ului dintr-un act artizanal, fragmentar și obscur, într-o **disciplină vizibilă, formală, umană și reproductibilă**.